

**RADA NAUKOWA DYSCYPLINY  
INFORMATYKA TECHNICZNA I TELEKOMUNIKACJA POLITECHNIKI WARSZAWSKIEJ**

zaprasza na

OBRONĘ ROZPRAWY DOKTORSKIEJ

**mgr inż. Justyny Sylwii Stypułkowskiej**

która odbędzie się w dniu **18 czerwca 2025 roku**, o godzinie **13:00** w trybie hybrydowym

Temat rozprawy:

**” Analiza obrazów wielospektralnych i RGB z zastosowaniem głębokich sieci neuronowych dla potrzeb automatycznej oceny parametrów kondycji kukurydzy przez robota polowego”**

Promotor: prof. dr hab. inż. Przemysław Rokita – Politechnika Warszawska

Recenzenci: dr hab. inż. Leszek Chmielewski – Szkoła Główna Gospodarstwa Wiejskiego w Warszawie

prof. dr hab. inż. Rafał Scherer – Politechnika Częstochowska

prof. dr hab. inż. Adam Wojciechowski – Politechnika Łódzka

Obrona odbędzie się zdalnie na platformie MS Teams oraz stacjonarnie w Audytorium Centralnym Wydziału Technik Informatycznych Politechniki Warszawskiej w Warszawie. Osoby zainteresowane uczestnictwem w obronie proszone są o zgłoszenie chęci uczestnictwa w formie elektronicznej na adres sekretarza komisji: dr hab. inż. Tomasz Martyn, email: [tomasz.martyn@pw.edu.pl](mailto:tomasz.martyn@pw.edu.pl), do dnia 16 czerwca 2025r. godz. 12:00.

Z rozprawą doktorską i recenzjami można zapoznać się w Czytelni Biblioteki Głównej Politechniki Warszawskiej, Warszawa, Plac Politechniki 1.

Streszczenie rozprawy doktorskiej i recenzje są zamieszczone na stronie internetowej: <https://www.bip.pw.edu.pl/Postepowania-w-sprawie-nadania-stopnia-naukowego/Doktoraty/Wszczete-po-30-kwietnia-2019-r/Rada-Naukowa-Dyscypliny-Informatyka-Techniczna-i-Telekomunikacja/mgr-inz.-Justyna-Sylwia-Stypulkowska>

Przewodniczący Rady Naukowej Dyscypliny  
Informatyka Techniczna i Telekomunikacja  
Politechniki Warszawskiej  
**prof. dr hab. inż. Jarosław Arabas**



**Politechnika Łódzka**

**Instytut Informatyki**

Rada Naukowa Dyscypliny  
**INFORMATYKA TECHNICZNA  
I TELEKOMUNIKACJA**

Sekretariat

Data wpływu... **7.04.25r.**

Numer.....

Łódź, 28 marca 2025 roku

prof. dr hab. inż. Adam Wojciechowski

Instytut Informatyki

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Politechnika Łódzka

Al. Politechniki 8, 93-590 Łódź

### **RECENZJA ROZPRAWY DOKTORSKIEJ**

Tytuł rozprawy: **Analiza obrazów wielospektralnych i RGB z zastosowaniem głębokich sieci neuronowych dla potrzeb automatycznej oceny parametrów kondycji kukurydzy przez robota polowego**

Autor rozprawy: **mgr inż. Justyna Stypułkowksa**

Promotor rozprawy: **prof. dr hab. inż. Przemysław Rokita**

**Jakie zagadnienie naukowe jest rozpatrywane w pracy (teza rozprawy) i czy zostało ono dostatecznie jasno sformułowane przez Autorkę? Czy tematyka rozprawy jest aktualna lub dostatecznie ważna? Jaki charakter ma rozprawa (teoretyczny, doświadczalny, inny)?**

Pani mgr inż. Justyna Stypułkowksa podjęła w swojej pracy doktorskiej, realizowanej w ramach doktoratu wdrożeniowego, zagadnienie wizyjnej oceny upraw kukurydzy. Tematyka rozprawy dotyczyła zastosowania technik i metod uczenia maszynowego w klasyfikacji obrazów RGB i obrazów wielospektralnych. W ramach zagadnienia badawczego sformułowano cele pracy. Obejmują one zarówno opracowanie dedykowanych zbiorów danych jak i przeprowadzenie badań z wykorzystaniem wielozadaniowych głębokich sieci neuronowych w celu analizy obrazów wielospektralnych i obrazów RGB na potrzeby automatycznej klasyfikacji faz rozwoju kukurydzy w skali BBCH, poziomów nawodnienia kukurydzy oraz porażenia kukurydzy wybranymi patogenami. Rozwiązanie jest dedykowane do systemu robota polowego, którego funkcjonalność wynika z rozwoju tzw. rolnictwa precyzyjnego, któremu poświęcony był doktorat wdrożeniowy i towarzyszący jemu projekt naukowo-badawczy, finansowany z NCBR, w którym Doktorantka brała udział. Zagadnienie badawcze zostało zatem precyzyjnie sformułowane i wyjaśnione.

Tematyka rozprawy jest aktualna i ważna w kontekście automatyzacji rolnictwa oraz precyzyjnego wizyjnego nadzoru upraw. W literaturze można spotkać zastosowanie sieci neuronowych w procesie monitorowania upraw, ale konkretnie dla upraw kukurydzy prac jest bardzo niewiele. Tym samym podjęcie tematyki dla upraw kukurydzy jest istotne i wymaga opracowania autorskich rozwiązań informatycznych.

Rozprawa ma charakter eksperymentalny i nosi w sobie istotny potencjał praktyczny. Doktorantka przeprowadziła szereg prób uczenia głębokich modeli sieci neuronowych, autorsko przygotowanym zbiorem danych oraz ocenę ich przydatności w zadaniu klasyfikacji cech roślin kukurydzy, w trzech przedmiotowych kontekstach. Przedstawione analizy są ciekawe i rzeczowe. Praca z racji podjętej problematyki, zakresu oraz metodologii badawczej jest pełnoprawnym osiągnięciem o charakterze naukowo-badawczym w dziedzinie nauk inżynierjno-technicznych, w dyscyplinie Informatyka Techniczna i Telekomunikacja.



**Czy w rozprawie przeprowadzono w sposób właściwy analizę źródeł (w tym literatury światowej i stanu zagadnień w przemyśle) świadczącą o dostatecznej wiedzy Autorki? Czy wnioski z przeglądu źródeł sformułowano w sposób jasny i przekonujący?**

W przedstawionej do oceny rozprawie doktorskiej, przegląd literatury rozpoczyna się w rozdziale drugim i podzielony jest na trzy logiczne części. Pierwsza poświęcona jest przybliżeniu wybranych architektur sieci neuronowych w zadaniu klasyfikacji. Trudno określić jaki związek z podjętym problemem ma przedstawienie formalnego modelu perceptronu, procesu uczenia oraz propagacji wstecznej błędu, czy sieci splotowej, szczególnie tak synergicznego z pracą [Rashka 2018], jeśli formalny opis i oznaczenia nie występują w dalszej części pracy. Po wprowadzeniu formalnym, Doktorantka przywołała uznane głębokie architektury wielozadaniowych sieci neuronowych: *ResNet50*, *ResNet101*, *ResNeXt101*, *Mask2Former*, *ConvNeXt*, *EfficientNet*, *Vision Transformer*, *Swin Transformer*, in., ale ich dyskusja była już znacznie bardziej ogólna. Niemniej lektura tej części rozdziału była przyjemna i przydatna dla zbudowania kontekstu badawczego. Jednocześnie uzasadnienie wyboru dyskutowanych architektur, szczególnie w zakresie predyspozycji do klasyfikowania zadanych cech roślin kukurydzy, na pewno ubogaciłoby rozprawę. Druga część rozdziału drugiego została poświęcona analizie charakterystyk obrazów wielospektralnych i RGB oraz ocenie jakości modeli i technik walidacyjnych. Szkoda, że pomimo deklaracji podjęcia problemów detekcji i klasyfikacji, Autorka skoncentrowała się na miarach klasyfikacji, a nie omówiła szerzej miar detekcji, do której odnosi się w rozprawie. Ostatnia część przeglądu dotyczy dziedzinowej analizy faz wzrostu kukurydzy, nawodnienia roślin, identyfikacji patogenów oraz analizy nielicznych rozwiązań referencyjnych. Ciekawe dlaczego spośród rozwiązań literaturowych żadne nie zostało uznane za referencyjne i zestawione z rozwiązaniami Autorki, szczególnie, że ponoć „opracowane modele pokazały dużą skuteczność” w podobnym zadaniu - str. 69 [Zhao i in., 2020][Bera i in., 2024]?

Rozdział trzeci to analiza materiałów i metod badawczych. Rozdział jest o tyle ważny, że stanowi przegląd narzędzi do akwizycji danych, w tym m.in. opisano specjalistyczne kamery wielospektralne *MicaSense RedEdge MX Dual* i *MapIR Survey3 Red+Green+NIR*. Opis jest umiarkowanie kompletny. Szkoda, że na wzór kamery *MicaSense* nie przeanalizowano pasm spektralnych kamery *MapIR* i kamery RGB, gdyż wtedy można by przedyskutować praktyczne różnice pomiędzy poszczególnymi urządzeniami – wszystkie bowiem urządzenia rejestrują, w pewnym sensie, spektrum z okolic światła czerwonego i zielonego. Bardzo przydatna byłaby też dyskusja poszczególnych klas obrazów (cech kukurydzy) w świetle poszczególnych kanałów spektralnych. Czy istnieją dziedzinowe przesłanki sugerujące, że w którymkolwiek spektrum fal świetlnych którakolwiek klasa (np. porażenia patogenami, czy nawodnienia) jest lepiej widoczna? Czy poczyniono w tym zakresie jakiekolwiek spostrzeżenia?

W zakresie metodologii badawczej opisy są bardzo obszerne, ale miejscami nieklarowne. Przykładowa wątpliwość dotyczy deklarowanego zadania jednoczesnej/sekwencyjnej detekcji i klasyfikacji. Jeśli dla klasyfikacji faz rozwojowych oznaczano rośliny na obrazach wielokątami to jaką formalnie rolę odgrywała detekcja w procesie klasyfikacji i na ile obciążająca, dla finalnej klasyfikacji, była ewentualna niedokładna detekcja? Czy monitorowano dokładność detekcji roślin na obrazach? Jak wyglądała detekcja w przypadku nawodnienia? Czy do analizy nawodnienia brano pod uwagę tylko największą roślinę kukurydzy na obrazie? Szkoda, że przy okazji sygnalizowania wskaźnika *bbox\_mAP\_50* nie podano dokładnej jego definicji i nie przedyskutowano wpływu detekcji na klasyfikację obrazów.

Wartościowa jest analiza zbiorów danych, szczególnie liczności i zbalansowania klas, która uzasadnia poprawność wyboru miar klasyfikacji. Trudno natomiast domyśleć się jaka była intencja rejestrowania osobnych zbiorów danych RGB we wczesnych fazach rozwoju roślin i w późniejszych fazach. Czy Doktorantce zależało jedynie na rozdzieleniu zbiorów RGB odpowiadających kamerze *MicaSense*



i *MapIR*? Czy skoro wiadomo, że wielkość zbioru danych uczących zwykle pozytywnie wpływa na jakość uczonego nim modelu, nie warto rozważyć połączenia zbiorów? Chociaż Doktorantka wspomina o podziale zbiorów danych na podzbiór treningowy, walidacyjny i testowy to czy przy, jakby nie patrzeć, ograniczonym i unikalnym zbiorze obrazów nie warto było rozważyć kilku różnych podziałów danych i krzyżowej walidacji algorytmów?

Przegląd należy uznać za rzeczowy i umiarkowanie aktualny – większość prac jest sprzed 3 lat lub starsza. Bibliografia jest dość obszerna i liczy 98 pozycji, w tym 2 prace z udziałem Doktorantki oraz zgłoszenie patentowe. W dorobku Autorki znajduje się praca w czasopiśmie *Machine Graphics and Vision* oraz publikacja na międzynarodowej konferencji *FedCSIS*. Całokształt dokonań publikacyjnych Doktorantki jest wystarczający w świetle obowiązujących wymagań.

**Czy Autorka rozwiązała postawione zagadnienie, czy użyła właściwej do tego metody i czy przyjęte założenia są uzasadnione?**

W obszarze podjętej problematyki, Doktorantka rzetelnie przeprowadziła analizę problemu, określiła kluczowe etapy prac badawczych, przygotowała zbiory danych, zaproponowała głębokie architektury sieci neuronowych, po czym przeprowadziła eksperymenty weryfikujące ich przydatność w klasyfikacji charakterystyk roślin kukurydzy, co systematycznie wspierało kontrybucje Doktorantki.

Dobór architektur głębokich sieci neuronowych do przedmiotowego problemu był odpowiedni, chociaż prawdopodobnie intuicyjny. Doktorantka wybrała popularne modele architektur sieci głębokich, które dostosowała do postawionych celów badawczych. Zbiór obrazów rejestrujących fazy rozwoju roślin przetwarzany był przy użyciu zaadaptowanych architektur *HTC\_x101*, *HTC\_r101*, *HTC\_r50*, *Mask2Former*, zaś zbiór obrazów dokumentujących różne poziomy nawodnienia i zbiór dokumentujący porażenie patogenami był przetwarzany przy użyciu modeli *ConvNeXt Small*, *ResNet50*, *EfficientNet-B3*, *Vision Transformer*, *Swin Transformer*. Doktorantka nie doprecyzowała jednak kryteriów doboru modeli do zagadnień klasyfikacyjnych. Prawdopodobnie przeprowadzenie eksperymentów z udziałem wszystkich modeli dla wszystkich danych dałoby szerszy obraz możliwości wykorzystania sztucznej inteligencji w ocenie upraw kukurydzy?

Rozdział 4 poświęcony jest nominalnie detekcji i klasyfikacji faz rozwoju kukurydzy w skali BBCH na obrazach RGB i obrazach wielospektralnych z kamery *MicaSense*. Eksperymenty prowadzone były zgodnie ze schematem zademonstrowanym na rysunku 4.1, który nie uwypukla fazy detekcji, ale określa warianty badań. Zakres prowadzonych badań miał na celu porównanie klasyfikacji obrazów RGB, klasyfikacji poszczególnych (dziesięciu) kanałów spektralnych oraz klasyfikacji agregującej wynik z poszczególnych kanałów z użyciem klasyfikatora głosującego. Doktorantka przeprowadziła wiele wariantów eksperymentów i opracowała wyniki w formie tabel oraz wykresów reprezentujących krzywe uczenia poszczególnych modeli. Wartościowym uzupełnieniem byłoby przeanalizowanie nie tylko wyników, ale macierzy pomyłek i dyskusja ewentualnych przyczyn niedoskonałości. Ciekawy jest wniosek z badań wskazujący, że największą skuteczność można uzyskać z użyciem klasyfikatora głosującego oraz, że najlepszymi kanałami okazały się kanał 5 i 7 kamery wielospektralnej. Ze względu na fakt, że kanał 5 jest podpasmem spektrum światła czerwonego zaś kanał 7 jest podpasmem spektrum światła zielonego to nasuwa się spostrzeżenie, że warto być może sprawdzić skuteczność klasyfikacji obrazu RGB w schemacie głosowania klasyfikatorów trzech kanałów R, G, i B lub tylko dwóch R i G. Czy skoro pozostałe kanały wielospektralne obciążają wynik głosowania (są gorsze) może warto ograniczyć głosowanie tylko do kanału 5 i 7 jako najbardziej reprezentatywnych? Ciekawa byłaby też dyskusja porównująca obrazy z kanałów słabszych i lepszych, aby odpowiedzieć na pytanie dlaczego obrazy rejestrowane w konkretnych pasmach światła (kanał 5 i 7) lepiej demonstrować fazy wzrostu roślin?



Rozdział 5 poświęcony był również detekcji i klasyfikacji faz rozwoju kukurydzy w skali BBCH z kamery RGB i z kamery wielospektralnej *MapIR Survey3 Red+Green+NIR*. Kamera wielospektralna *MapIR* ma przewagę względem kamery *MicaSense* polegającą na rejestrowaniu obrazów z jednego obiektywu, przez co obrazy dla różnych pasm spektrum światła mają takie samo ujęcie i różna perspektywa rejestracji nie obciąża klasyfikacji. Wyniki są nieco gorsze niż dla poprzedniej kamery wielospektralnej, ale czy rzeczywiście jest tak jak pisze Doktorantka, że „obraz RGB dostarcza pełniejszego spektrum informacji”? Być może należało klasyfikację przeprowadzić nie na najlepszym modelu ale w schemacie głosowania klasyfikatorów poszczególnych kanałów kamery *MapIR*, który wykazał swoją przewagę na poprzednim etapie eksperymentów? Na str. 152 Doktorantka, podsumowując cechy modelu *HTC\_r50*, wspomina o zdolności modelu do generalizacji. Nasuwa się pytanie, które etapy eksperymentu pozwalają na takie stwierdzenie?

Rozdział 6 poświęcony jest detekcji i klasyfikacji poziomów nawodnienia na obrazach RGB i wielospektralnych z kamery *MapIR*. Do klasyfikacji trzyklasowej Autorka wybrała zupełnie inny garnitur modeli sieci głębokich. Przeprowadzone eksperymenty wykazały istotną zmianę jakościową wyników klasyfikacji. Nawet pojedyncze modele pozwalały uzyskać skuteczność klasyfikacji bliską 90%. Na pewno wpływ na to miało samo zadanie, mniejsza liczba klas, ale być może też same architektury sieci. Czy nie warto byłoby zastosować najlepszych modeli z danego eksperymentu (tj. *Vision Transformer*, *Swin Transformer*) do klasyfikacji faz rozwoju rośliny? Niemniej analiza wyników jest pogłębiona. Wartościowa jest analiza skuteczności modelu w czasie uczenia i odpowiadająca im analiza wartości funkcji straty. W rozdziale związanym z przygotowaniem danych Autorka deklaruje podział na zbiór treningowy, walidacyjny i testowy, podczas gdy w rozdziale szóstym prawdopodobnie zbiór walidacyjny był jednocześnie zbiorem testowym, co prawdopodobnie warto było wyraźniej zakomunikować. Można też odnieść wrażenie, że wartości liczbowe zakomunikowane w tabeli 6.2 nie odpowiadają konkluzji w sekcji 6.5.

Rozdział 7 poświęcony jest zagadnieniu detekcji i klasyfikacji porażenia kukurydzy patogenami. W tym scenariuszu badań Doktorantka posłużyła się zewnętrznym zbiorem obrazów z platformy *Kaggle*, które dostępne były tylko w formacie RGB. Do testów wykorzystano analogiczny zbiór modeli sieci głębokich co w zadaniu dotyczącym klasyfikacji nawodnienia. Wyniki okazały się rewelacyjne dla niektórych modeli (np. *Swin/Vision Transformer*) gdyż sięgały blisko 100%. Zastanawiające jest to, że bazowano wyłącznie na obrazach RGB i przy aż 5 klasach (dla nawodnienia były 3) uzyskano tak wysmienite rezultaty. Ciekawe czy zauważalna była jakościowa różnica pomiędzy obrazami RGB zaczerpniętymi z platformy *Kaggle* i zarejestrowanymi przez Doktorantkę? Czy może bardzo wysoka skuteczność klasyfikacji wynika z ewidentności samych klas?

Dysertację wieńczą sekcje podsumowujące kontrybucje Doktorantki, wkład w dyscyplinę naukową oraz perspektywa dalszych badań. Doktorantka, ze względu na reporterski styl narracji, koncentruje się mocniej na sprawozdawczym referowaniu wyników z poszczególnych eksperymentów niż na jakościowej analizie porównawczej uzyskanych wyników. Zabrakło chyba pogłębionej refleksji nad wiarygodnością uzyskanych wyników, czy generalizacją samych modeli sieci, co można było zawrzeć w sekcji poświęconej ograniczeniom/założeniom przeprowadzonych badań. Niemniej cele badawcze, które sformułowano na początku pracy należy uznać za osiągnięte.

Podsumowując, przedstawione rozwiązania, eksperymenty oraz wyniki są ciekawe i nowatorskie. Szczególnie wartościowy jest zbiór danych obrazów wielospektralnych oraz seria przeprowadzonych na nich eksperymentów, stosująca bardzo nowoczesne modele głębokich sieci neuronowych do zadania klasyfikacji charakterystyk upraw kukurydzy.



**Na czym polega oryginalność rozprawy, co stanowi samodzielny i oryginalny dorobek Autorki, jaka jest pozycja rozprawy w stosunku do stanu wiedzy, czy poziomowi techniki reprezentowanych przez literaturę światową?**

Oryginalność rozprawy doktorskiej leży przede wszystkim w szeroko pojętym zakresie eksperymentów przeprowadzonych z użyciem nowoczesnych architektur głębokich sieci spłotowych, których zadaniem była klasyfikacja faz rozwoju kukurydzy, określenie poziomu nawodnienia roślin, czy faktu porażenia roślin patogenami. W szczególności najważniejszymi, moim zdaniem, osiągnięciami są:

- opracowanie autorskiego zbioru danych obrazów z kamer RGB i kamer wielospektralnych, która stanowi bazę dla dalszych badań nad monitorowaniem upraw kukurydzy lub nawet badań generalizowalnych na inne uprawy;
- przeprowadzenie usystematyzowanej serii eksperymentów na wielu architekturach sieci neuronowych, w różnych strategiach klasyfikacji (indywidualnie, poprzez głosowanie), które zwróciły uwagę na możliwość wykorzystania obrazów wielospektralnych w monitorowaniu upraw kukurydzy;
- wdrożenie opracowanych rozwiązań do rzeczywistego robota polowego, co wiązało się z realizacją projektu naukowo-badawczego, finansowanego przez NCBR, i zakończyło się, między innymi, zgłoszeniem patentowym;

Wymienione osiągnięcia stanowią bezsprzecznie oryginalny wkład Doktorantki w rozwój dyscypliny Informatyka Techniczna i Telekomunikacja.

**Czy Autorka wykazała umiejętność poprawnego i przekonującego przedstawienia uzyskanych przez siebie wyników (zwięzłość, jasność, poprawność redakcyjna rozprawy)?**

Praca stanowi bardzo szerokie studium analizy obrazów wielospektralnych przy użyciu wielozadaniowych architektur sieci neuronowych. Oprócz przedmiotowego zagadnienia monitorowania charakterystyk upraw kukurydzy, którym Doktorantka poświęciła, w sposób naturalny, wiele miejsca, same badania dotyczące skuteczności klasyfikacji, były opisane dokładnie, czasami nawet w kilku miejscach można było dostrzec powtarzające się treści.

Miejscami Doktorantka rozluźniała podejście do terminologii badań i szczegółów eksperymentów, ale można to usprawiedliwić zakresem prowadzonych eksperymentów oraz wysoce praktycznym ujęciem badań, które narzucało konkretne wdrożenie. Niewielka, choć wystarczająca liczba publikacji z udziałem Doktorantki wynika prawdopodobnie ze zgłoszenia patentowego poczynionego w 2023 roku, co formalnie blokuje możliwość publikowania rezultatów przed samym złożeniem wniosku patentowego. Docenienia wymaga spójny i przyjemny w lekturze układ pracy, który konsekwentnie i logicznie ujmuje przedmiotowy obszar badań. Pracę czyta się bardzo dobrze.

Pod względem edycyjnym praca została zredagowana starannie. Praca zawiera wszystkie niezbędne spisy oraz bibliografię. Autorce nie udało się uniknąć drobnych błędów. Przykładowo: na str. 32 i 34 pisząc o funkcji sinusoidalnej Doktorantka miała na myśli prawdopodobnie funkcję sigmoidalną; na str. 33 pojawiają się nieprecyzyjności oznaczeń wymiarów macierzy  $W$  (np.  $m$  zamiast  $m_1$ ). Nie udało się uniknąć drobnych błędów tekstowych, np.: „za zbiorze” str. 61, 63; „po późniejszego” str. 73. W tekście pracy można też napotkać wiele powtarzających się stwierdzeń – przykładowo zawartość sekcji 3.4.3 i 3.5.1 jest mocno redundantna.

Osobną kwestią są rysunki zamieszczone w pracy, które z racji swojej szczegółowości są trudne w analizie w formie wydrukowanej – niestety na niektórych napisy są prawie nieczytelne, np. rys. 3.18,



6.3, 6.4, 8.1. Dodatkowo chyba na rysunku 6.10 kolory są niewłaściwie opisane (np. model *ConvNeXt Small* i *EfficientNet-B3*)?

### Jakie są słabe strony rozprawy i jej główne wady?

Rozprawa doktorska nie zawiera istotnych uchybień, a poniższe uwagi mają przede wszystkim charakter dyskusyjny. Przykładowe kwestie, które mogłyby zostać omówione, lecz nie ujmują pozytywnej oceny rozprawy, dotyczą:

W deklaracji opracowania zbiorów danych (str. 17) Autorka wspomina o przygotowaniu danych treningowych, walidacyjnych i testowych, podczas gdy w dalszej części pracy dane walidacyjne są mieszane z danymi testowymi. Skłania to czytelnika do refleksji jak optymalnie operować podobnymi zbiorami danych, o wystarczającej, chociaż niezbyt dużej liczności, aby uzyskać najbardziej wiarygodne wyniki eksperymentów? Czy nie warto było stosować walidacji krzyżowej i dynamicznego podziału zbioru danych dla podniesienia wiarygodności uzyskanych wyników?

Chociaż w celach pracy i w opisach zagadnień pojawia się deklaracja zarówno detekcji i klasyfikacji to jednak samej detekcji nie omówiono zbyt szczegółowo w pracy. Główne schematy w rozdziałach 4,5,6 nawet nie zaznaczają etapu detekcji, który pojawia się w temacie rozdziałów i jest immanentnym etapem wielozadaniowych modeli użytych w pracy. Proszę o przybliżenie roli detekcji w analizie obrazów oraz wpływu detekcji na finalną skuteczność klasyfikacji.

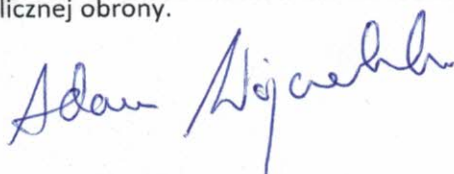
Podsumowanie rozdziału 4 nie do końca jest spójne z wynikami eksperymentów opisanych w tym rozdziale. Przykładowo na str. 142 Doktorantka pisze, że algorytm *HTC\_r101* osiągnął najlepsze wyniki m.in. dla kanału 10, czego nie potwierdza rys. 4.9. Dodatkowo Autorka pisze, że algorytm *HTC\_x101* osiągnął najgorsze wyniki m.in. dla kanału 5, czego nie potwierdzają wyniki zaprezentowane na rys. 4.10. Te prawdopodobne omyłki i koncentracja na suchych wynikach, w innym miejscach rozprawy, skłoniły mnie do pytań odnośnie ogólnej przydatności/przewagi obrazów wielospektralnych w analizie faz rozwoju kukurydzy, ocenie nawodnienia czy porażenia patogenami. Które pasma spektralne i dlaczego mają szansę zapewnić dokładniejszą analizę upraw? Które modele sieci głębokich neuronowych i dlaczego wydają się najwłaściwsze do analizy obrazów w wąskim spektrum fal? Co możemy zyskać dzieląc pasmo rejestracji na pasma spektralne skoro w dużej części eksperymentów obraz RGB uzyskiwał najlepsze wyniki i uzasadniano to szerszym spektrum dostarczanych informacji?

### Konkluzja

Uważam, że cele rozprawy doktorskiej zostały w pełni zrealizowane. Doktorantka przedstawiła w rozprawie nowe badania z zakresu wizyjnej klasyfikacji obrazów wielospektralnych przy ocenie upraw kukurydzy. Uzyskane przez Doktorantkę wyniki uważam za oryginalne, wartościowe i ciekawe poznawczo. Zakres prowadzonych badań był szeroki, analiza opracowanych rozwiązań wszechstronna, a warsztat badawczy Doktorantki był właściwy. Doktorantka może poszczycić się również aktywnością projektową i zgłoszeniem patentowym, które stanowią ważne uzupełnienie jej dorobku i dają szansę na owocne kontynuowanie ambitnych badań.

Tym samym rozprawa prezentuje wartościowe osiągnięcia naukowe w obszarze dyscypliny Informatyka Techniczna i Telekomunikacja oraz potwierdza umiejętność samodzielnego prowadzenia przez Doktorantkę pracy naukowej.

Bez najmniejszej wątpliwości stwierdzam, że recenzowana rozprawa doktorska Pani mgr inż. Justyny Stypułkowskiej spełnia wymagania stawiane rozprawom doktorskim, przez obowiązującą ustawę. Wnoszę o jej przyjęcie i dopuszczenie rozprawy do publicznej obrony.





9 maja 2025 r.

dr hab. inż. Leszek J. Chmielewski, prof. SGGW  
Instytut Informatyki Technicznej  
Szkoła Główna Gospodarstwa Wiejskiego w Warszawie  
ul. Nowoursynowska 159, 02-776 Warszawa

## **Recenzja rozprawy doktorskiej**

autor rozprawy:

**mgr inż. Justyna Stypułkowska**  
Instytut Informatyki, Politechnika Warszawska  
Instytut Sieci Badawczej Łukasiewicz – Instytut Lotnictwa

tytuł rozprawy:

**Analiza obrazów wielospektralnych i RGB  
z zastosowaniem głębokich sieci neuronowych  
dla potrzeb automatycznej oceny  
parametrów kondycji kukurydzy przez robota polowego**

Recenzja została przygotowana w odpowiedzi na powołanie mnie, uchwałą Rady Naukowej Dyscypliny Informatyka Techniczna i Telekomunikacja Politechniki Warszawskiej z dnia 10 grudnia 2024 r., na recenzenta rozprawy doktorskiej Pani mgr inż. Justyny Stypułkowskiej. Promotorem rozprawy jest prof. dr hab. inż. Przemysław Rokita. Przewód jest prowadzony w dyscyplinie *informatyka techniczna i telekomunikacja*.

Rozprawa doktorska wpłynęła do Kancelarii Głównej SGGW 10 marca 2025 r.

## **1 Zawartość rozprawy**

### **1.1 Charakterystyka ogólna**

Rozprawa składa się z dziewięciu rozdziałów, oraz zawiera spis rysunków, spis tabel, bibliografię i siedem załączników.

We wstępie, będącym rozdziałem 1, uzasadniono podjęcie tematu rozwoju i zastosowania sztucznej inteligencji w zagadnieniu związanym z rolnictwem precyzyjnym i krótko przedstawiono dwa projekty, w ramach których rozwinięte i zastosowane zostały wyniki przedstawione w rozprawie. Sformułowano główne zadanie, jakie zostało zrealizowane, oraz cele pracy. Przedstawiono także strukturę dalszej części pracy.

W rozdziale 2 umieszczono przegląd literatury dla tematyki rozprawy od strony informatycznej i na tej podstawie przedstawiono zarys metod, które będą wykorzystane w pracy. Omówiono budowę głębokich sieci neuronowych, obrazowanie barwne i wielospektralne, oraz obszernie omówiono zagadnienie walidacji i testowania i miar ich jakości. Następnie omówiono kwestie opisu spodziewanych wyników rozpoznawania, a w tym fazy rozwoju BBCH dla kukurydzy, stopnia nawodnienia z podziałem na trzy lasy, oraz obecności patogenów.

Dalej, dokonano przeglądu literatury od strony systemów monitorowania upraw kukurydzy.

Rozdział jest zakończony opisem czterech zidentyfikowanych luk w obecnym stanie wiedzy. Stwierdzono, że warto podjąć badania nad obrazami pozyskiwanymi przez urządzenie naziemne, a nie latające, należy używać obrazów wielospektralnych, a nie tylko barwnych RGB, należy zająć się tworzeniem dużych zbiorów opisanych obrazów, oraz rozszerzyć zbiór badanych architektur uczenia głębokiego, szczególnie używając architektur typu transformer.

W rozdziale 3 bliżej i bardziej szczegółowo opisano architekturę i implementację sieci. Przedstawiono metodykę przygotowywania danych obrazowych. Opisano procesy uczenia i klasyfikacji. Opisano klasyfikator wykorzystujący głosowanie wśród zbioru klasyfikatorów.





W kolejnych dwóch rozdziałach opisano, w jaki sposób przeprowadzono rozpoznawanie faz rozwoju kukurydzy w skali BBCH z wykorzystaniem obrazów RGB i wielospektralnych, jakie uzyskano wyniki, oraz jak wyglądały krzywe uczenia się. W rozdziale 4 przedmiotem opisu były badania przeprowadzone na podstawie obrazów z kamery MicaSense RedEdge MX Dual, natomiast w rozdziale 5 przedstawiono wyniki badań wykonanych z użyciem zdjęć z wielospektralnej kamery MapIR Survey3 Red+Green+NIR.

W następnych dwóch rozdziałach pokazano analogiczne wyniki dla poziomów nawodnienia (rozdział 6), oraz porażenia patogenami (rozdział 7), dla uprawy kukurydzy.

Sposób prezentacji wyników tych badań przedstawiono w rozdziale 8. Prezentacja jest realizowana za pomocą osobnej aplikacji.

W rozdziale 9 kończącym rozprawę dokonano podsumowania uzyskanych osiągnięć, wskazano elementy oryginalne, przedstawiono wnioski i zarekomendowano kierunki przyszłych badań.

## 1.2 Cele pracy

Zadanie, jakie zostało postawione, oraz cele pracy zostały krótko sformułowane w rozdziale 1.2.

Zadanie zostało scharakteryzowane następująco (dosłowny cytat):

*Niniejsza praca ma za zadanie zaproponowanie rozwiązań dla systemu robota polowego, wykorzystujących uczenie głębokie oraz analizę obrazów RGB i wielospektralnych, które pozwolą na wsparcie procesu automatycznego określania kondycji roślin uprawnych, a w szczególności kukurydzy.*

Wyróżniono cztery cele główne (dosłowne cytaty):

- *Przeprowadzenie badań z wykorzystaniem głębokich sieci neuronowych w celu analizy obrazów wielospektralnych i RGB na potrzeby opracowania rozwiązania służącego do automatycznej detekcji i klasyfikacji faz rozwoju kukurydzy w skali BBCH. Wybór najskuteczniejszego rozwiązania w oparciu o wybrany algorytm uczenia głębokiego oraz wybrany typ zobrazowania, które w połączeniu ze sobą pozwalają osiągnąć najlepsze wyniki.*
- *Przeprowadzenie badań z wykorzystaniem głębokich sieci neuronowych w celu analizy obrazów wielospektralnych i RGB na potrzeby opracowania rozwiązania służącego do automatycznej detekcji i klasyfikacji poziomów nawodnienia kukurydzy. Wybór najskuteczniejszego rozwiązania w oparciu o wybrany algorytm uczenia głębokiego oraz wybrany typ zobrazowania, które w połączeniu ze sobą pozwalają osiągnąć najlepsze wyniki.*
- *Przeprowadzenie badań z wykorzystaniem głębokich sieci neuronowych w celu analizy obrazów RGB na potrzeby opracowania rozwiązania służącego do automatycznej detekcji i klasyfikacji porażenia kukurydzy wskazanymi patogenami. Wybór najskuteczniejszego algorytmu uczenia głębokiego, który pozwala osiągnąć najlepsze wyniki.*
- *Stworzenie dedykowanych autorskich zbiorów danych (podzielonych następnie na zbiory treningowe, walidacyjne i testowe), składających się z oznaczonych obrazów RGB i oznaczonych obrazów wielospektralnych, służących do realizacji zadań związanych z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH oraz detekcją i klasyfikacją poziomów nawodnienia kukurydzy. Wykorzystanie do tego celu własnego poletka testowego.*

Nie została sformułowana teza pracy – nie ma takiej potrzeby.

## 2 Dyskusja zawartości i wyników rozprawy

### 2.1 Uwagi pozytywne

#### 2.1.1 Ważność tematyki rozprawy

Praca dotyczy twórczego zastosowania głębokich sieci neuronowych w rolnictwie precyzyjnym. Sieci zostały użyte do rozpoznawania najważniejszych parametrów dotyczących stanu upraw kukurydzy. W powszechnym przekonaniu w Polsce uprawia się raczej podstawowe zboża, niż



kukurydzą. Produkcja kukurydzy w Polsce w 2023 roku wynosiła około 6 mln ton i przekroczyła poziom zapotrzebowania krajowego o 2 mln ton. Był to drugi po pszenicy gatunek uprawiany w Polsce (Łuczak 2023). W 2024 roku powierzchnia upraw kukurydzy wynosiła około 1.3 mln ha, zaś pszenicy 2.4 mln ha, pszenżyta 1.1 mln ha, żyta i jęczmienia po 0.7 mln ha, owsa 0.5 mln ha, itd. (ziemniaka: 0.2 mln ha) (GUS 2024). Kukurydza jest trwale drugą uprawą zbożową w Polsce. Zatem, badania mające na celu optymalizację uprawy kukurydzy mają duże znaczenie gospodarcze.

Recenzowana praca mieści się zatem w kategorii badań w zakresie dyscypliny *informatyka techniczna i telekomunikacja* wspierających rolnictwo precyzyjne.

Przynależność pracy do zakresu rolnictwa precyzyjnego jest uzasadniona przez przetestowanie rozwiązań opracowanych w ramach recenzowanej rozprawy w prototypie robota polowego opracowanego w ramach projektu „Inteligentny robot spełniający wymogi rolnictwa precyzyjnego” finansowanego przez NCBiR i Komisję Europejską.

Metody głębokich sieci neuronowych, wraz z ich licznymi rodzajami, to obecnie obszar najbardziej burzliwego rozwoju metod informatycznych. Stale pojawiają się opisy nowych architektur i nowych dziedzin ich zastosowań. Jak dotąd jest regułą, że jakość uzyskiwanych wyników przewyższa tę, którą można było osiągnąć metodami stosowanymi przed powstaniem sieci głębokich. Rozpatrywana praca dobrze sytuuje się w tym trendzie.

### 2.1.2 Opis stanu wiedzy

W pracy przedstawiono obszerny opis stanu wiedzy w jej zakresie. Dobry opis architektur sieci głębokich, transformerów i ich połączeń znajdujemy w rozdziałach 2.1.4-2.1.6. Dość dobrze napisano rozdział 2.3 o ocenie jakości modeli i technikach walidacyjnych. Szczególnie zwróciłem uwagę na rozdział 2.3.2, w którym Autorka stara się rozwikłać zamęt istniejący w literaturze w związku ze zbiorem uczącym, walidacyjnym i testowym, gdzie liczni autorzy oprogramowania, a za nimi autorzy publikacji, mylą testowanie z walidacją. Autorka wyjaśnia, że mając małe zbiory danych można użyć tego samego zbioru do walidacji i testowania, bo wtedy testowanie zachodzi na zbiorze nieznanym dla procesu uczenia, ale to nie powinna być normalna praktyka.

Informatyczne metody wnioskowania o stanie upraw kukurydzy na podstawie obrazów Doktorantka analizuje w rozdziale 2.5. Analiza wprost doprowadziła do zidentyfikowania czterech luk badawczych.

1. Mało jest badań, w których korzysta się z obrazów wykonanych z ziemi, z małej odległości od roślin.
2. Korzystanie z obrazów wielospektralnych jest rzadkością.
3. Brakuje dostępnych dużych zbiorów opisanych obrazów roślin znajdujących się w określonej fazie rozwoju i określonych warunkach.
4. Wielu algorytmów i architektur jak dotąd nie zbadano w zastosowaniu do określania stanu roślin.

Nie zabrakło także opisu cech charakteryzujących rośliny, a konkretnie kukurydzą, w zakresie fazy rozwoju (skala BCCH), stanu nawodnienia i identyfikacji patogenów atakujących roślinę. Jest to opis zadania od strony nauk rolniczych wystarczający dla celów postawionych w rozprawie.

### 2.1.3 Elementy nowości

W zakres pracy weszło wykorzystanie siedmiu modeli klasyfikatorów typu sieć głęboka i transformer, wymienionych w rozdziale 3.3, s. 97. Wystąpiły one w kilku wersjach i ich połączeniach, oraz dla różnych danych – obrazów RGB i wielospektralnych, a jednocześnie dla trzech klas: faz wzrostu, poziomu nawodnienia i porażenia wybranymi patogenami. Realizacja zadania wymagała skonfigurowania złożonego środowiska obliczeniowego, odzwierciedlającego całą strukturę eksperymentów, co widać w Załączniku B do rozprawy. Wykonano także oprogramowanie do agregacji wyników. Obsługa takiej liczby modeli i wielu wersji parametrów i danych wymagała doskonałego zrozumienia procesów zachodzących w fazach uczenia i testowania i stanowiło



niemałe wyzwanie. Procesy te zorganizowano tak aby, między innymi, zapewnić dobór właściwych hiperparametrów i uniknąć przepełnienia pamięci.

Przygotowanie zbiorów danych, z adnotacjami dla faz wzrostu i poziomu nawodnienia i z oznaczonymi obszarami należącymi do klas patogenów, jest niewątpliwie bardzo znaczącym osiągnięciem Doktorantki.

Koncepcja zastosowania głosowania jako sposobu podejmowania decyzji w przypadku posiadania wielu wyników klasyfikacji jest bardzo słuszna, ponieważ odwołuje się do zasady odporności na dane odstające, a nawet częściowo błędne. W rozprawie zastosowano głosowanie ważone oceną jakości predykcji, co powinno jeszcze poprawić jego wyniki. Jest to metoda właściwa, choć zdecydowanie nie oryginalna – liczba publikacji odwołujących się do głosowania w kontekście rozpoznawania i klasyfikacji przekracza 1.5 miliona. Dlatego zdecydowałem się nie cytować tu moich własnych prac o tej tematyce, lecz tylko wspomnieć artykuł (Józwik 1983), w którym w zadaniu klasyfikacji z wieloma klasami zastosowano głosowanie między rozmytymi wynikami z klasyfikatorów dla par klas.

#### 2.1.4 Weryfikacja metod

Metody sprawdzano wyliczając klasyczne miary jakości wyników klasyfikacji – precyzja, czułość, dokładność i miara F1. Wyniki przedstawiono dla obrazów RGB, dla poszczególnych kanałów w kamerze wielospektralnej, oraz dla głosowania wszystkich kanałów. Jeśli weźmiemy pod uwagę dokładność lub miarę F1 jako miary zarówno błędów typu fałszywie pozytywny, jak i fałszywie negatywny, to miary te dla poszczególnych kanałów nie przekraczają wartości 0.5, zaś dla RGB i głosowania dochodzą do niemal do wartości 0.7, lub 0.8, dla najlepszych modeli klasyfikatorów.

Osiągnięcie bardzo dobrych wartości miar jakości klasyfikacji, z dokładnością dochodzącą niemal do 1.00, dla klasyfikacji patogenów, jest znakomitym wynikiem.

#### 2.1.5 Publikacje

Doktorantka ma cztery publikacje we współautorstwie, w tym jedną tylko wraz z promotorem (można dodać pracę w MG&V do bazy Politechniki Warszawskiej). Najwyższa punktacja to 70, sumaryczna punktacja to 150 punktów MNiSW, trzy cytowania w Scopus, cztery w Scholar. Jest to dorobek wystarczający na etapie rozprawy doktorskiej.

#### 2.1.6 Projekty

Opracowane rozwiązanie zastosowano w rzeczywistym projekcie robota polowego, w ramach projektu *Inteligentny robot spełniający wymogi rolnictwa precyzyjnego*, Program Operacyjny Inteligentny Rozwój. Beneficjent: UNIA SP. Z O.O.

Wiedzę pozyskaną w ramach rozprawy wykorzystano w projekcie *Zwiększenie konkurencyjności polskich towarów roślinnych na rynkach międzynarodowych poprzez podniesienie ich jakości i bezpieczeństwa fitosanitarnego FITOEXPORT*, program GOSPOSTRATEG, NCBiR, nr 1/385957/5/NCBR/2018.

### 2.2 Uwagi dyskusyjne i krytyczne

Uwagi są raczej wstępem do dyskusji podczas obrony, niż krytyką.

**Usuwanie obrazów wadliwych przypadków** Usuwanie obrazów nie zawierających kukurydzy wydaje się oczywiste; jednak system powinien poprawnie klasyfikować takie obrazy – właśnie jako nie zawierające obiektu. Dla systemów sztucznej inteligencji wykrycie przypadku znajdującego się poza dziedziną obrazów nauczonych jest trudne, ale takie obrazy z pewnością znajdują się wśród obrazów wykonanych przez autonomicznego robota. Co wtedy zostanie zwrócone jako wynik analizy?

Usuwanie obrazów kukurydzy przesłoniętej przez nietypowe obiekty również może dać wynik w postaci braku odporności systemu autonomicznego na nietypowe przypadki.



Proszę o komentarz na ten temat.

**Wykorzystanie obrazów zewnętrznych** Wskutek nierozwinięcia się grzybów, mających być patogenami w zadaniu, wykorzystano zbiory z serwisu Kaggle. Jaki to mogło mieć wpływ na wyniki, które powinny być charakterystyczne dla upraw realizowanych w Polsce? czy dla własnych obrazów również osiągnięto by tak dobre wyniki rozpoznawania?

**Przygotowanie upraw i obrazów** Kto przygotowywał poletka, obsługiwał uprawy, wreszcie – wykonywał zdjęcia? Czy to była Doktorantka, czy inne osoby, być może pracujące w ramach projektów wymienionych w rozprawie? Tej informacji nie znalazłem w tekście.

Dalsze uwagi podane tutaj są na pograniczu uwag merytorycznych i redakcyjnych. Zresztą, wszystkie te uwagi należy uważać raczej za podstawę do dyskusji. Nie podważają one mojej ogólnej pozytywnej oceny rozprawy doktorskiej.

**Niejednolity poziom szczegółowości i trudności** Opisy w rozdziale 2.1, z jego kolejnymi podrozdziałami, zaczynają się od tematów prostych i powszechnie znanych, które są wyjaśnione szczegółowo, natomiast kończą się na zagadnieniach bardzo złożonych, które są z kolei opisywane na wysokim poziomie ogólności. Nie jest jasne, do jakiego czytelnika jest skierowany taki tekst. Czytelnik, który skorzysta na początkowej części opisu, ale nie zrozumie końcowej, natomiast czytelnik, który pojmie końcową część, nie potrzebuje tak podstawowych wyjaśnień, jak te umieszczone w części początkowej. Moim zdaniem, rozprawa doktorska nie jest podręcznikiem, w którym trzeba wyjaśnić wszystko dokładnie od początku do końca, lecz raczej raportem z badań na zaawansowanym poziomie. Dlatego wystarczyłby opis na wysokim poziomie ogólności, w którym dokładniej byłyby opisane tylko te zagadnienia, które nie byłyby powszechnie znane, a byłyby niezbędne do zrozumienia opisu własnych badań Doktorantki.

Pewien niepokój wzbudziło we mnie to, że znaczne fragmenty rozdziału 2.1 są oparte na tylko dwóch źródłach: (Raschka 2018) – rozdziały 2.1.1-2.1.3, oraz (Weidman 2019) – rozdział 2.1.3. Jednak, źródła są podane poprawnie. Dotyczy to również rysunków, które najwyraźniej zostały wykonane samodzielnie przez Autorkę na podstawie źródeł. Tym bardziej zatem uważam, że wchodzenie w szczegóły treści powszechnie znanych jest niepotrzebne, skoro można je znaleźć w znakomitych, dostępnych źródłach.

**Okrągłe wartości miar jakości** Dlaczego uzyskano dwa razy wartości okrągłe, równe 0.800000, dla modelu mask2former i obrazów RGB, w tabelach 4.3 i 4.4? Czy liczba obiektów testowych nie była zbyt mała? Jaki ma sens podawanie miar jakości z sześcioma miejscami po przecinku?

**Brak danych liczbowych w rozdziałach *wyбір najlepszych rozwiązań*** W rozdziałach 4.6, 5.4, 6.5 i 7.4, a także 9.1.2, spodziewałbym się informacji ilościowej, jaką ostatecznie dokładność, czułość, specyficzność osiągnięto w najlepszym przypadku. Taką informację podano tylko w tabeli 7.1, s. 176, dla patogenów, i są to wyniki bardzo dobre.

**Umiarkowane dokładności osiągnięte finalnie** Dla BBCH i nawodnienia, w najlepszych przypadkach, osiągnęto dokładności w zakresie 0.6-0.7-0.8, i to tylko dla wyników zagregowanych – RGB lub głosowanie dla wielu zakresów. Są to wartości umiarkowane. Dla poszczególnych zakresów częstości wyniki były znacząco poniżej 0.5, czyli właściwie poniżej granicy wyniku losowego. Czym można tłumaczyć tak niskie wyniki, mimo tak wielkiego nakładu pracy na ich uzyskanie? Co można zrobić, aby były one wyższe? Czy takie wyniki zadowolają ewentualnych użytkowników robota polowego?

**Brak porównania wyników liczbowych z innymi publikacjami** Tematyka rozprawy jest specyficzna, ale być może istnieją inne publikacje, z którymi można by porównać wyniki osiągnięte w rozprawie? Na pewno nie byłyby to wyniki dla tych samych danych, ale jakieś częściowe porównanie byłoby pożądane, przynajmniej dla ogólnej orientacji. Prawdopodobnie byłoby to możliwe dla danych z serwisu Kaggle, dla których Doktorantka uzyskała wyniki bardzo korzystne.



## 2.3 Uwagi redakcyjne i techniczne

Pracę napisano i zredagowano nadzwyczaj staranie. Potknięcia są drobne i bardzo nieliczne. Zauważam je tylko dla porządku, nie spodziewam się omawiania ich w odpowiedzi na recenzję.

**Połączenie opisów implementacji modeli i ich charakterystyk** W rozdziale 3.3, w którym opisano implementację sieci, znalazły się krótkie fragmenty opisów ich charakterystyk. Te opisy powinny się raczej znaleźć w rozdziale 2, być może w podrozdziale 2.1.6, gdzie w ramach analizy stanu wiedzy opisywane są architektury rozważane pod kątem ich zastosowania. Struktura opisu mogłaby na tym zyskać.

### Nomenklatura

Wiele z użytych miar jakości klasyfikacji nie jest w rzeczywistości metrykami, nie definiują one funkcji spełniających warunki odległości. Zamiast słowa ~~metryka~~ należałoby użyć słowa miara, miara jakości klasyfikacji. Niestety, w literaturze słowo *metric* w tym zastosowaniu jest powszechne (na przykład, w (Raschka 2018)), zatem trudno zarzucać Autorce, że postępuje za trendem; zresztą, w kilku miejscach w pracy użyła poprawnej nazwy.

Dalsze potknięcia nomenklaturowe: ~~poligen~~wielokąt; ~~konwolucja~~splot. Moim zdaniem, można mówić o sieciach konwolucyjnych, bo ta nazwa się przyjęła, ale gdy mamy do czynienia ze zwykłym splotem, jak np. na s. 42, to trudno operację splotu traktować tak, jakby istniała wyłącznie dla tworzenia nowych cech między warstwami sieci. Jednocześnie cieszy, że Doktorantka użyła prawidłowo nazwy obrazów wielospektralnych, a nie multispektralnych.

O ile wiem, istnieje iloczyn skalarny wektorów, lub iloczyn macierzy i składowa, ale mówienie o iloczynie skalarnym *dwóch* macierzy jest zbyt daleko idącym skrótem myślowym (s. 43, góra). Formalnie, liczymy sumę iloczynów skalarnych wektorów będących wierszami (lub kolumnami) macierzy. Inaczej, można także mówić o sumie wyrazów macierzy będącej iloczynem po współrzędnych (iloczynem Hadamarda lub Shura) dwóch macierzy. Najlepiej jednak po prostu zatrzymać się na definicji splotu w przypadku cyfrowym.

Wielokrotnie użyto słowa *predykcja*, jakby chodziło o przewidywanie wartości, która pojawi się w przyszłości. Chodzi natomiast o *klasyfikację* lub po prostu *rozpoznanie*.

Doktorantka użyła wielu słów angielskich niepotrzebnie: ~~score~~ocena lub punktacja; ~~area~~powierzchnia, ~~accuracy~~dokładność, ~~precision~~precyzja, ~~recall~~, również nazywana sensitivityczułość, ~~F1-measure~~miara F1, ~~dataset~~zbiór danych. Nawet znalazły się odmienne słowa ~~datasetów~~ (s. 71<sub>11</sub>) i ~~outputów~~ (s. 95<sup>2</sup>). Jeśli już używa się słów obcojęzycznych, ale nie w roli zamienników słów polskich, tylko w ich własnej roli, to należy je pisać *czcionką wyróżnioną (kursywą)* i objaśnić – tę zasadę Doktorantka czasem stosuje.

### Błędy składania wzorów matematycznych

Str. 22, równania na funkcję aktywacji: klamra powinna być dostosowana wielkością do objętych nią wzorów. Użyty symbol funkcji przypomina symbol zbioru pustego ' $\emptyset$ ', podczas gdy w (Raschka 2018) jest  $\phi(z)$  (grecka litera *fi*). Taka litera jest użyta we wzorze na  $J(w)$  na s. 27.

Str. 24, wyrażenia na uaktualnienia  $\Delta w_0, \dots, \Delta w_2$ : lepiej byłoby oznaczyć *wyjście* symbolem, niż pisać nazwę. Jeśli nazwa, to pisana prostą czcionką, podobnie jak nazwy funkcji, np.  $\sin, \cos$ . Nie jest to iloczyn zmiennych  $w, y, j, \acute{s}, c, i, e$ .

Mimo tych uwag, należy zauważyć, że rozprawa jest złożona nadzwyczaj starannie (jak na tekst zredagowany w użytym edytorze).

**Anakolut** Niemal w każdej pracy, od inżynierskiej do habilitacyjnej, jest przynajmniej jeden *anakolut* (Pietrzak 2024). Anakolut jest jedną z plag języka polskiego. W recenzowanej pracy anakolut znalazł się na stronie 27 pod rysunkiem 2.4: *Porównując schemat perceptronu ze schematem Adaline, główna różnica polega na tym, że Adaline wykorzystuje ciągłe wartości (...)*.

### Błędy literowe i podobne, niewłaściwe znaki

Str. 5<sub>11</sub>: ~~prażenia~~porażenia kukurydzy.

Str. 28<sup>6</sup> i w wielu innych miejscach: zamiast znaków polskiego cudzysłowu użyto zna-



ków "." z podstawowego zbioru znaków ASCII. Jednak linię wyżej użyto prawidłowego cudzysłowu.

Str. 53<sup>6-8</sup>: pominięto wyraz: (...) aktualizacja zapytań w czasie treningu daje możliwość lepszego dostosowania modelu do różnych scenariuszy i przez to pozwala ulepszyć jakość generowanych masek.

### 3 Podsumowanie

W rozprawie można wyróżnić istotne elementy nowatorskie oraz inne pozytywne aspekty omówione w rozdziale 2.1, zaś uwagi dyskusyjne i krytyczne omówione w rozdziale 2.2 nie umniejszają w istotny sposób wartości pracy.

Cele pracy zostały osiągnięte, a luki badawcze zostały w znaczącym stopniu wypełnione.

W rozprawie zaproponowano i przebadano bardzo dużą liczbę modeli klasyfikacyjnych, w różnych architekturach głębokich sieci neuronowych i transformerów, dla trzech zagadnień ważnych dla upraw kukurydzy: faza rozwoju w skali BBCH, stopień nawodnienia, oraz patogeny, które mogą zaatakować uprawę. Wskazano, które modele i jakiego typu obrazy można wykorzystać dla osiągnięcia zadowalających wyników. Stworzono w tym celu obszerne zbiory opisanych obrazów i zbudowano skomplikowane oprogramowanie do przeprowadzenia licznych procesów uczenia i testowania klasyfikatorów i prezentacji wyników tych procesów.

**Wnioski** Powyższy opis, uwzględniający uwagi pozytywne jak również uwagi dyskusyjne i krytyczne, uzasadnia moje ostateczne wnioski o następującej treści.

Recenzowana rozprawa w postaci opracowania pisemnego **wykazuje i prezentuje ogólną wiedzę teoretyczną kandydatki w tej dyscyplinie naukowej oraz umiejętność samodzielnego prowadzenia pracy naukowej**. Rozprawa stanowi oryginalne rozwiązanie problemu naukowego w dyscyplinie informatyka techniczna i telekomunikacja, a także oryginalne rozwiązanie w zakresie zastosowania wyników własnych badań naukowych w sferze gospodarczej. Zagadnienie badawcze zostało prawidłowo postawione i skutecznie rozwiązane, a rozwiązanie zostało rzetelnie zweryfikowane. Tym samym rozprawa spełnia wymagania określone w art. 187.1 i 2 Ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce. **Rozprawę oceniam pozytywnie** i stawiam wniosek o skierowanie jej do dalszych etapów przewodu doktorskiego.



### Cytowana literatura

- GUS (2024, 18 grudnia). Obszary tematyczne, uprawy rolne i ogrodnictwo. wynikowy szacunek głównych ziemioplodów rolnych i ogrodnictwa w 2024 roku. Główny Urząd Statystyczny. URL <https://stat.gov.pl/obszary-tematyczne/rolnictwo-lesnictwo/uprawy-rolne-i-ogrodnictwo/wynikowy-szacunek-glownych-ziemioplodow-rolnych-i-ogrodnictwa-w-2024-roku>.
- Jóźwik, A. (1983). A learning scheme for a fuzzy k-NN rule. *Pattern Recognition Letters* 1(5-6), 287-289. doi:10.1016/0167-8655(83)90064-8.
- Pietrzak, G. (2024). Anakolut. In G. Pietrzak (Ed.), *Skarbnica Figur Retorycznych*. URL <https://figury.net.pl/slownik/anakolut>. [Dostęp: 9 listopada 2024].
- Raschka, S. (2018). *Python. Uczenie maszynowe*. Helion. URL <https://helion.pl/ksiazki/python-uczenie-maszynowe-sebastian-raschka>. Tytuł oryginału: *Python Machine Learning*. Packt Publishing, 2016. Tłum. K. Sawka.



- Weidman, S. (2019). *Uczenie głębokie od zera. Podstawy implementacji w Pythonie*. Helion. URL <https://helion.pl/ksiazki/uczenie-glebokie-od-zera-podstawy-implementacji-w-pythonie-seth-weidman>. Tytuł oryginału: *Deep Learning from Scratch: Building with Python from First Principles*. O'Reilly, 2018. Tłum. T. Walczak.
- Łuczak, P. (2023, 31 maja). Kukurydza w polsce: 40 lat minęło. AgroKonsument.pl. URL <https://agrokonsument.pl/kukurydza-z-polsce-40-lat-minelo/>.



POLITECHNIKA WARSZAWSKA

DYSCYPLINA NAUKOWA INFORMATYKA TECHNICZNA I

TELEKOMUNIKACJA

DZIEDZINA NAUK INŻYNIERYJNO-TECHNICZNYCH

## Rozprawa doktorska

mgr inż. Justyny Stypułkowskiej

**Analiza obrazów wielospektralnych i RGB z zastosowaniem głębokich  
sieci neuronowych dla potrzeb automatycznej oceny parametrów  
kondycji kukurydzy przez robota polowego**

Promotor

prof. dr hab. inż. Przemysław Rokita

Warszawa, 2024





## **Podziękowania**

Szczególne podziękowania pragnę złożyć mojemu Promotorowi Panu prof. dr hab. inż. Przemysławowi Rokicie, bez którego realizacja niniejszej pracy byłaby niemożliwa. Wyjątkowe podejście Pana Profesora oparte na życzliwości, współpracy, udzielaniu cennych wskazówek i motywowaniu okazało się ogromnym wsparciem podczas realizacji badań, prezentacji wyników oraz pisaniu rozprawy.

Podziękowania kieruję również moim przełożonym z Sieci Badawczej Łukasiewicz – Instytutu Lotnictwa, Panu dr inż. Pawłowi Stężyckiemu – Dyrektorowi Sieci Badawczej – Instytutu Lotnictwa, Panu Mariuszowi Kacprzakowi – Kierownikowi Działu Teledetekcji oraz Panu Karolowi Rotchimmelowi – Kierownikowi Sekcji Rozwoju Narzędzi Teledetekcyjnych pełniącemu rolę Opiekuna pracy, którzy zapewнили dostęp do niezbędnego sprzętu obliczeniowego i kamer oraz możliwość realizacji prac badawczych i wdrożenia uzyskanych wyników.





## Streszczenie

Analiza obrazów wielospektralnych i RGB z zastosowaniem głębokich sieci neuronowych dla potrzeb automatycznej oceny parametrów kondycji kukurydzy przez robota polowego.

*Celem niniejszej rozprawy jest przeprowadzenie badań z wykorzystaniem głębokich sieci neuronowych w celu analizy obrazów wielospektralnych i RGB na potrzeby opracowania systemu do automatycznej detekcji i klasyfikacji faz rozwoju kukurydzy w skali BBCH, poziomów nawodnienia oraz porażenia kukurydzy patogenami. Zadania te są częścią rzeczywistych prac projektowych prowadzonych w ramach doktoratu wdrożeniowego realizowanego w Politechnice Warszawskiej oraz Sieci Badawczej Łukasiewicz – Instytucie Lotnictwa. Analiza literatury pokazuje niezliczone przykłady zastosowań sztucznej inteligencji, która jest wyjątkowo szybko rozwijającym się działem informatyki mającym wpływ na rozwój wielu innych dyscyplin naukowych. Niniejsza rozprawa skupia się na wykorzystaniu głębokich sieci neuronowych do analizy obrazów wspierając tym samym rozwój rolnictwa precyzyjnego. Podczas realizacji badań skupiono się na wypełnieniu zauważonych luk badawczych, a przeprowadzone eksperymenty oparte zostały o nowoczesne architektury uczenia głębokiego i pozwoliły sprawdzić ich możliwości w konfrontacji z nowymi zastosowaniami.*

*Wszystkie przedstawione cele zostały zrealizowane, a finalne rozwiązanie zostało wypracowane na podstawie analizy porównawczej wyników uzyskanych przy zastosowaniu różnych architektur głębokich sieci neuronowych oraz wskazanych typów zobrazowań. Na podstawie uzyskanych wyników zaproponowano trzy modele odpowiadające poszczególnym podzadaniom dotyczącym detekcji i klasyfikacji faz rozwojowych, poziomów nawodnienia i porażenia kukurydzy wybranymi patogenami. Zaproponowane podejścia zostały skonfrontowane z podobnymi rozwiązaniami dostępnymi w literaturze, co podkreśliło innowacyjność opracowanych w ramach niniejszej rozprawy metod.*

*Rozwiązania zaproponowane w rozprawie zostały wdrożone w rzeczywistym projekcie Robot Polski NCBiR, którego celem jest stworzenie robota polowego spełniającego wymogi rolnictwa precyzyjnego. Ponadto, rozszerzone wyniki badań stały się również wkładem merytorycznym do innego projektu pn. Fitoexport NCBiR. Fakty te świadczą o rzeczywistej aplikowalności opracowanego rozwiązania oraz o istnieniu zapotrzebowaniu na tego typu badania.*

**Słowa kluczowe:** *głębokie sieci neuronowe, sztuczna inteligencja, analiza obrazów RGB, analiza obrazów wielospektralnych, rolnictwo precyzyjne*

## Abstract

Analysis of multispectral and RGB images using deep neural networks for automatic evaluation of maize condition parameters by a field robot.

*The aim of this dissertation is to carry out research using deep neural networks to analyse multispectral and RGB images for the development of a system for the automatic detection and classification of maize development phases on the BBCH scale, irrigation levels and maize pathogen infestation. These tasks are part of the actual project work carried out within the framework of an implementation doctorate carried out at the Warsaw University of Technology and the Łukasiewicz Research Network - Institute of Aviation. An analysis of the literature shows countless examples of applications of artificial intelligence, which is an exceptionally fast-growing branch of computer science influencing the development of many other scientific disciplines. This thesis focuses on the use of deep neural networks for image analysis supporting the development of precision agriculture. During the course of the research, the focus was on filling the research gaps noted, and the experiments carried out were based on modern deep learning architectures and allowed their capabilities to be tested when confronted with new applications.*

*All the objectives presented were met, and the final solution was developed on the basis of a comparative analysis of the results obtained using different deep neural network architectures and the indicated types of imaging. Based on the results obtained, three models were proposed corresponding to the different subtasks concerning the detection and classification of developmental phases, irrigation levels and roasting of maize with selected pathogens. The proposed approaches were contrasted with similar solutions available in the literature, which highlighted the innovation of the methods developed in this dissertation.*

*The solutions proposed in the dissertation have been implemented in the actual NCBiR Polish Robot project aimed at creating a field robot meeting the requirements of precision agriculture. In addition, the extended research results also became a substantive contribution to another project entitled Fitoexport NCBiR. These facts testify to the real-life applicability of the developed solution and the existence of a demand for this type of research.*

**Keywords:** *deep neural networks, artificial intelligence, RGB image analysis, multispectral image analysis, precision agriculture*



## Spis treści

|                                                                                                                  |    |
|------------------------------------------------------------------------------------------------------------------|----|
| <b>1. Wstęp</b>                                                                                                  | 10 |
| 1.1. Kontekst pracy i uzasadnienie podjęcia tematu                                                               | 10 |
| 1.2. Cele pracy                                                                                                  | 15 |
| 1.3. Struktura rozprawy                                                                                          | 16 |
| <b>2. Przegląd literatury i wykorzystywanych technik</b>                                                         | 18 |
| 2.1. Sztuczne sieci neuronowe i uczenie głębokie w klasyfikacji obiektów                                         | 18 |
| 2.1.1. Wprowadzenie do sztucznych sieci neuronowych                                                              | 19 |
| 2.1.2. Uczenie głębokie i propagacja wsteczna                                                                    | 32 |
| 2.1.3. Konwolucyjne sieci neuronowe (CNN)                                                                        | 37 |
| 2.1.4. Architektury ResNet50, ResNet101, ResNeXt101                                                              | 48 |
| 2.1.5. Architektura sieci typu Transformer w analizie obrazów                                                    | 50 |
| 2.1.6. Zaawansowane architektury: HTC, Mask2Former, ConvNeXt, EfficientNet, Vision Transformer, Swin Transformer | 51 |
| 2.2. Przetwarzanie obrazów RGB i wielospektralnych                                                               | 55 |
| 2.2.1. Charakterystyka danych RGB                                                                                | 55 |
| 2.2.2. Charakterystyka danych wielospektralnych                                                                  | 56 |
| 2.2.3. Techniki przetwarzania danych obrazowych                                                                  | 57 |
| 2.3. Ocena jakości modeli i techniki walidacyjne                                                                 | 58 |
| 2.3.1. Metryki oceny (Accuracy, Precision, Recall, F1-score)                                                     | 58 |
| 2.3.2. Techniki walidacji Hold-out                                                                               | 59 |
| 2.3.3. Analiza błędów i interpretacja wyników                                                                    | 61 |
| 2.4. Zastosowanie skali BBCH, nawodnienia i identyfikacji patogenów w analizie obrazów                           | 63 |
| 2.4.1. Skala BBCH i jej zastosowanie w analizie faz rozwojowych roślin                                           | 63 |
| 2.4.2. Znaczenie nawodnienia roślin w kontekście analizy obrazów                                                 | 65 |
| 2.4.3. Identyfikacja patogenów roślinnych przy użyciu analizy obrazów                                            | 65 |
| 2.5. Przegląd podobnych rozwiązań                                                                                | 66 |
| 2.5.1. Przegląd literatury i analiza istniejących rozwiązań                                                      | 67 |
| 2.5.2. Wnioski i identyfikacja luk badawczych                                                                    | 69 |
| <b>3. Materiały i metody badawcze</b>                                                                            | 71 |
| 3.1. Przyjęta metodyka przygotowywania danych oraz prowadzenia badań                                             | 71 |
| 3.2. Opis danych obrazowych                                                                                      | 74 |
| 3.2.1. Źródła danych i przygotowanie pól testowych                                                               | 74 |
| 3.2.2. Krótkie wprowadzenie dotyczące użytych kamer                                                              | 77 |
| 3.2.3. Dane RGB                                                                                                  | 77 |
| 3.2.4. Dane wielospektralne z kamery RedEdge                                                                     | 84 |

|           |                                                                                                                                           |            |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------|------------|
| 3.2.5.    | Dane wielospektralne z kamery MapIR .....                                                                                                 | 87         |
| 3.2.6.    | Proces oznaczania obrazów .....                                                                                                           | 89         |
| 3.3.      | Implementacja architektur głębokich sieci neuronowych .....                                                                               | 95         |
| 3.3.1.    | Implementacja modeli HTC_x101, HTC_r101, HTC_r50.....                                                                                     | 96         |
| 3.3.2.    | Implementacja Mask2Former.....                                                                                                            | 97         |
| 3.3.3.    | Implementacja modeli ConvNeXt small, ResNet50, EfficientNet_B3, Vision Transformer, Swin Transformer.....                                 | 98         |
| 3.4.      | Proces trenowania modeli.....                                                                                                             | 100        |
| 3.4.1.    | Trenowanie modeli HTC, Mask2Former, YOLOX .....                                                                                           | 102        |
| 3.4.2.    | Trenowanie modeli ConvNeXt, ResNet, EfficientNet, Vision Transformer, Swin Transformer .....                                              | 102        |
| 3.4.3.    | Techniczne aspekty trenowania modeli.....                                                                                                 | 103        |
| 3.5.      | Proces detekcji i klasyfikacji obiektów .....                                                                                             | 104        |
| 3.5.1.    | Zastosowanie modeli.....                                                                                                                  | 107        |
| 3.5.2.    | Klasyfikator głosujący i jego zastosowanie .....                                                                                          | 109        |
| <b>4.</b> | <b>Detekcja i klasyfikacja faz rozwoju kukurydzy w skali BBCH z wykorzystaniem obrazowania RGB i kamery wielospektralnej RedEdge.....</b> | <b>115</b> |
| 4.1.      | Schemat opracowanego rozwiązania.....                                                                                                     | 115        |
| 4.2.      | Wyniki uzyskane dla danych wielospektralnych.....                                                                                         | 118        |
| 4.3.      | Wyniki uzyskane dla danych RGB.....                                                                                                       | 122        |
| 4.4.      | Wyniki uzyskane przy zastosowaniu klasyfikatora głosującego.....                                                                          | 125        |
| 4.5.      | Krzywe uczenia dla danych wielospektralnych i RGB .....                                                                                   | 130        |
| 4.6.      | Porównanie wyników i wybór najlepszych rozwiązań.....                                                                                     | 142        |
| <b>5.</b> | <b>Detekcja i klasyfikacja faz rozwoju kukurydzy w skali BBCH przy wykorzystaniu obrazowania RGB i kamery wielospektralnej MapIR.....</b> | <b>146</b> |
| 5.1.      | Schemat opracowanego rozwiązania.....                                                                                                     | 146        |
| 5.2.      | Wyniki uzyskane dla danych wielospektralnych i RGB .....                                                                                  | 148        |
| 5.3.      | Krzywe uczenia dla danych wielospektralnych i RGB .....                                                                                   | 153        |
| 5.4.      | Porównanie wyników i wybór najlepszych rozwiązań.....                                                                                     | 159        |
| <b>6.</b> | <b>Detekcja i klasyfikacja poziomów nawodnienia kukurydzy.....</b>                                                                        | <b>161</b> |
| 6.1.      | Schemat zaproponowanego rozwiązania.....                                                                                                  | 161        |
| 6.2.      | Wyniki uzyskane dla danych wielospektralnych.....                                                                                         | 164        |
| 6.3.      | Wyniki uzyskane dla danych RGB.....                                                                                                       | 165        |
| 6.4.      | Krzywe uczenia dla danych wielospektralnych i RGB .....                                                                                   | 166        |
| 6.5.      | Porównanie wyników i wybór najlepszych rozwiązań.....                                                                                     | 175        |
| <b>7.</b> | <b>Detekcja i klasyfikacja porażenia kukurydzy.....</b>                                                                                   | <b>176</b> |
| 7.1.      | Schemat zaproponowanego rozwiązania.....                                                                                                  | 176        |



|                                                              |            |
|--------------------------------------------------------------|------------|
| 7.2. Wyniki i ich analiza.....                               | 179        |
| 7.3. Krzywe uczenia .....                                    | 180        |
| 7.4. Porównanie wyników i wybór najlepszego algorytmu.....   | 185        |
| <b>8. Prezentacja wyników .....</b>                          | <b>186</b> |
| 8.1. Aplikacja Gradio .....                                  | 186        |
| 8.2. Plik wyjściowy w formacie .csv .....                    | 187        |
| <b>9. Wnioski.....</b>                                       | <b>190</b> |
| 9.1. Podsumowanie.....                                       | 190        |
| 9.1.1. Główne osiągnięcia pracy i jej oryginalny wkład ..... | 191        |
| 9.1.2. Wnioski z przeprowadzonych badań.....                 | 194        |
| 9.2. Rekomendacje dla przyszłych badań.....                  | 197        |
| <b>Spis rysunków .....</b>                                   | <b>199</b> |
| <b>Spis tabel .....</b>                                      | <b>205</b> |
| <b>Bibliografia .....</b>                                    | <b>206</b> |
| <b>Załączniki .....</b>                                      | <b>215</b> |

# 1. Wstęp

## 1.1. Kontekst pracy i uzasadnienie podjęcia tematu

Postęp w nauce i technologii jest w głównej mierze wynikiem dążenia do poprawy istniejących rozwiązań, zwiększenia efektywności wybranych procesów oraz tworzenia nowych przełomowych narzędzi i technologii otwierających zupełnie nowe możliwości. Przykładem jednego z takich przełomowych osiągnięć jest sztuczna inteligencja będąca rezultatem wieloletniego rozwoju teorii z dziedziny matematyki i statystyki. Nierzadko nowe odkrycia muszą czekać na pełne ujawnienie swojego potencjału i tak też było w przypadku sztucznej inteligencji, której początki opierały się na teoriach matematycznych i statystycznych, istniejących początkowo jedynie w formie abstrakcyjnych modeli teoretycznych [Rosenblatt, 1958], [Rumelhart i in., 1986]. Wówczas możliwości ich praktycznego zastosowania były ograniczone brakiem dostępu do narzędzi o odpowiednio dużej mocy obliczeniowej. Dopiero postęp w elektronice i inżynierii komputerowej umożliwił rozwój zaawansowanych narzędzi obliczeniowych, które zrewolucjonizowały dziedziny związane z przetwarzaniem dużych ilości danych. Wraz z rozwojem sprzętu komputerowego nastąpił gwałtowny wzrost liczby nowych metod służących do gromadzenia, przetwarzania i analizy dużych zbiorów danych, co dla sztucznej inteligencji stanowi fundamentalny aspekt jej rozwoju.

To właśnie sztuczna inteligencja odegrała kluczową rolę w procesie doskonalenia technologii mających znaczący wpływ na różne sektory współczesnej nauki i gospodarki, oddziałując również na codzienne aspekty ludzkiego życia. Postępy w dziedzinie AI, zwłaszcza w obszarze uczenia głębokiego, umożliwiły realizację zadań, które wcześniej były trudno dostępne dla klasycznych algorytmów, w tym rozpoznawania wzorców, detekcji i klasyfikacji obiektów, przetwarzania i analizy obrazów, a także podejmowania decyzji na podstawie analizy dużych zbiorów danych. W kontekście analizy obrazów szczególną rolę odegrały konwolucyjne sieci neuronowe, a także sieci typu transformer zaadaptowane z rozwiązań typowych dla przetwarzania języka naturalnego. Oba typy sieci stanowią obecnie jedne z najbardziej zaawansowanych technik sztucznej inteligencji, umożliwiając efektywną analizę dużych i złożonych danych obrazowych [LeCun i in., 2015], [Vaswani i in., 2017].

Rozwój sztucznej inteligencji wiąże się jednak z pewnymi wyzwaniami. Jednym z nich jest dostęp do zbiorów danych, które są niezbędne do trenowania zaawansowanych modeli

sztucznej inteligencji. Dla przykładu, w sektorze nowoczesnego rolnictwa rozwój systemów wykorzystujących sztuczną inteligencję wymaga ciągłego pozyskiwania danych z różnych źródeł, w tym danych teledetekcyjnych, satelitarnych, obrazów pozyskiwanych dzięki BSP oraz czujników naziemnych [Kamilaris, Prenafeta-Boldú, 2018]. Ponadto, wdrożenie takich systemów na szeroką skalę wymaga zintensyfikowanej współpracy ekspertów z dziedziny sztucznej inteligencji, biologii oraz inżynierii.

Dodatkowym wyzwaniem, przed którym stoi sztuczna inteligencja, jest dostępność zaawansowanych zasobów obliczeniowych. Najnowocześniejsze modele AI, takie jak sieci typu transformer oraz wielkoskalowe konwolucyjne sieci neuronowe, wymagają do efektywnego działania znacznych mocy obliczeniowych. Dzięki postępowi w technologii procesorów graficznych (GPU) oraz wdrożeniu dedykowanych układów, takich jak Tensor Processing Units (TPU), sztuczna inteligencja może efektywnie przetwarzać i analizować duże ilości danych w krótszym czasie, niż dotychczas [Jouppi i in., 2017]. Ponadto, rozwój chmurowych przestrzeni obliczeniowych, takich jak Google Cloud, Amazon Web Services (AWS) czy Microsoft Azure, umożliwia łatwy dostęp do zaawansowanych zasobów obliczeniowych, co stanowi istotne ułatwienie we wdrażaniu systemów sztucznej inteligencji [Deng, 2018].

Dynamiczny rozwój AI jest wspierany również przez rosnący ekosystem oprogramowania, a narzędzia takie jak PyTorch czy TensorFlow ułatwiają wdrażanie modeli sztucznej inteligencji. Ich publicznie dostępny kod źródłowy wspomaga rozwój badań nad sztuczną inteligencją oraz jej praktycznym zastosowaniem w takich dziedzinach jak przemysł, medycyna oraz nowoczesne rolnictwo [Abadi i in., 2016], [Paszke i in., 2019].

W kontekście nowoczesnego rolnictwa, a w szczególności rolnictwa precyzyjnego, sztuczna inteligencja odgrywa kluczową rolę, zwłaszcza w monitorowaniu upraw w czasie rzeczywistym oraz precyzyjnym zarządzaniu środkami ochrony roślin, nawozami, wodą i zabiegami agrotechnicznymi. Rolnictwo precyzyjne, będące częścią szerszej koncepcji rolnictwa 4.0, opiera się na zaawansowanych technologiach, takich jak teledetekcja, wykorzystanie bezzałogowych statków powietrznych, naziemne sensory oraz sztuczna inteligencja, która umożliwia analizę danych zebranych z tychże źródeł [Wolfert i in., 2017].

Wdrożenie sztucznej inteligencji w rolnictwie precyzyjnym jest uzasadnione dążeniem do zwiększenia wydajności upraw przy jednoczesnym zmniejszeniu zużycia zasobów, takich jak środki ochrony roślin i sztuczne nawozy, co odpowiada na wyzwania związane z ochroną



środowiska oraz zaspokojeniem rosnącego zapotrzebowania na żywność w kontekście rozwijającej się populacji. Sztuczna inteligencja umożliwia przetwarzanie i analizę obrazów, detekcję i klasyfikację roślin, ich faz rozwojowych, a także wykrywanie chorób roślin na wczesnym etapie, czasem jeszcze przed pojawieniem się objawów widocznych dla ludzkiego oka [Mahlein i in., 2016], [Bock i in., 2010].

Warto podkreślić, że badania nad zastosowaniem sztucznej inteligencji w rolnictwie zazwyczaj koncentrują się na wybranych roślinach uprawnych (takich jak np. kukurydza), a wypracowane techniki mogą być skutecznie przenoszone na inne rośliny i ich obrazy, co pozwala na wykorzystanie nowych metod monitorowania jakości plonów, faz wzrostu oraz prognozowania wydajności [Zhao i in., 2020], [Sharma i in., 2020]. Spośród dostępnych w literaturze wyników badań warto wyróżnić te dotyczące analizy parametrów fizjologicznych roślin, takich jak indeks wegetacyjny, struktura liści, biomasa czy zawartość chlorofilu [Li i in., 2020]. Interesujące są również badania dotyczące ryżu i pszenicy, które umożliwiają precyzyjny monitoring wschodów roślin oraz detekcję niedoborów wody, co ma szczególne znaczenie dla zoptymalizowanego nawadniania [Kamilaris, Prenafeta-Boldú, 2018].

Szczególną uwagę warto poświęcić również badaniom, w których obiektem analizy jest kukurydza. Jest to jedna z najważniejszych roślin uprawnych na świecie i dlatego stanowi przedmiot licznych badań w dziedzinie sztucznej inteligencji i rolnictwa precyzyjnego. Wprowadzane sukcesywnie udoskonalenia w zakresie automatyzacji, monitorowania wzrostu i faz rozwojowych kukurydzy przy użyciu AI przyczyniają się do bardziej efektywnego zarządzania zasobami. Przykładem takich badań jest praca Xingmei Xu i współautorów [Xu i in., 2020], w której wykorzystano algorytmy Mask R-CNN i YOLOv5 do rozpoznawania liści kukurydzy. Badania te dowodzą, że zaawansowane techniki sztucznej inteligencji mogą skutecznie monitorować rozwój roślin, niezależnie od obecności chwastów na polu uprawnym.

Podobne badania przeprowadzili Liu i współautorzy [Liu i in., 2019], którzy wykazali skuteczność systemów sztucznej inteligencji opartych na obrazach RGB pozyskiwanych za pomocą BSP w procesie monitoringu siewek kukurydzy. Zastosowanie metod AI w rolnictwie precyzyjnym pozwoliło nie tylko na automatyzację procesów, ale także na ich skalowalność, co jest niezwykle cenne przy uprawach kukurydzy na dużym obszarze [Liu i in., 2019].

Mimo postępów w automatycznej analizie i monitoringu upraw z wykorzystaniem sztucznej inteligencji, wciąż istnieją liczne obszary wymagające dalszych badań mogących przyczynić się do poszerzenia obecnego stanu wiedzy w tej dziedzinie. Po pierwsze, większość

dostępnych w literaturze badań opiera się na obrazach pozyskiwanych za pomocą bezzałogowych statków powietrznych, co ogranicza dokładność monitoringu pojedynczych roślin. W tym kontekście rozwój technologii wykorzystujących roboty polowe, poruszające się po plantacjach, może otworzyć nowe możliwości pozyskiwania obrazów z bliższej odległości, co zapewni możliwość analizy bardziej precyzyjnych danych [Zhou i in., 2020].

Po drugie, w większości badań nad detekcją i klasyfikacją faz rozwojowych kukurydzy, wykrywaniem objawów porażenia patogenami oraz określaniem poziomów nawodnienia stosuje się wyłącznie obrazowanie RGB, mimo że obrazowanie wielospektralne mogłoby dostarczyć bardziej szczegółowych informacji o kondycji roślin [Zhou i in., 2020], [Mahlein i in., 2016]. Obrazowanie wykraczające poza zakres widzialny dla ludzkiego oka umożliwia analizę zmian zawartości chlorofilu oraz identyfikację stanów chorobowych we wczesnym stadium. Badania z wykorzystaniem obrazowania wielospektralnego dowodzą, że to podejście umożliwia wczesne wykrycie anomalii, co ma kluczowe znaczenie dla ochrony roślin i przeciwdziałania patogennym czynnikom. Oprócz identyfikacji chorób, istotnym aspektem jest możliwość porównywania obrazów roślin charakteryzujących się różnym stopniem nawodnienia oraz analiza ich kształtu. Rośliny zarejestrowane na obrazach wielospektralnych wyraźnie różnią się od tła, co może być wykorzystane w nowych podejściach do detekcji i klasyfikacji faz rozwojowych.

Po trzecie, nadal brakuje dedykowanych, dobrze oznaczonych zbiorów danych zawierających obrazy kukurydzy z oznaczonymi fazami rozwojowymi w skali BBCH oraz zbiorów danych obrazujących rośliny w zróżnicowanych warunkach nawodnienia. Stworzenie takich zbiorów w warunkach odpowiadających rzeczywistym polom uprawnym, obejmujących wszystkie fazy rozwojowe oraz zmienne warunki nawodnienia roślin, stanowi duże wyzwanie badawcze, kluczowe dla dalszego rozwoju sztucznej inteligencji w tym obszarze [Zhao i in., 2020]. Wzrost skuteczności systemów AI wymaga dostępu do zbiorów danych o wystarczająco dużej różnorodności (obrazy pozyskiwane o różnych porach dnia, w różnych warunkach pogodowych i o różnym nasłonecznieniu), co pozwala modelom na lepsze uogólnianie swoich predykcji na nowe przypadki obrazowanych obiektów o określonych charakterystykach.

Po czwarte, obecne badania w tej dziedzinie są prowadzone z wykorzystaniem ograniczonej liczby architektur uczenia głębokiego. Oznacza to, że istnieje ciągła potrzeba eksploracji i testowania nowych, bardziej zaawansowanych architektur, takich jak sieci typu transformer, które wprowadzają innowacyjne podejście do analizy obrazów. Sieci oparte na architekturze transformerów, takie jak Vision Transformer, zyskują na popularności dzięki

unikalnemu podejściu do przetwarzania i analizy obrazów, w tym obrazów o wysokim stopniu skomplikowania [Dosovitskiy i in., 2020]. Potencjał tych sieci, szczególnie w rolnictwie precyzyjnym, pozostaje w dużej mierze niewykorzystany, co stanowi kolejną istotną lukę badawczą, którą warto się zająć.

Celem niniejszej pracy jest wypełnienie wspomnianych luk badawczych poprzez zaprojektowanie i wdrożenie rozwiązania wykorzystującego głębokie sieci neuronowe oraz umożliwiającego zautomatyzowane określanie faz rozwojowych kukurydzy w skali BBCH, monitoring poziomów nawodnienia i identyfikację porażenia roślin wybranymi patogenami. Opracowany system wykorzystuje wyniki badań przeprowadzonych z użyciem zobrazowania RGB i wielospektralnego, pozyskiwanych z bliskiej odległości, z pozycji charakterystycznej dla sensorów zamontowanych na robocie polowym. Wynikiem przeprowadzonych kompleksowych prac badawczych jest realizacja powyższych zadań z wykorzystaniem najbardziej optymalnego typu zobrazowania, w trybie monitoringu w czasie rzeczywistym.

Niniejsza rozprawa powstała w ramach doktoratu wdrożeniowego i dotyczy prac prowadzonych w ramach rzeczywistych projektów realizowanych przez Sieć Badawczą Łukasiewicz – Instytut Lotnictwa. Projektem wiodącym dla niniejszej rozprawy jest projekt realizowany przez Sieć Badawczą Łukasiewicz – Instytut Lotnictwa we współpracy z Siecią Badawczą Łukasiewicz – Poznańskim Instytutem Technologicznym oraz firmą UNIA, w ramach Narodowego Centrum Badań i Rozwoju, pod tytułem „*Inteligentny robot spełniający wymogi rolnictwa precyzyjnego*”. Projekt ten doskonale wpisuje się w tematykę zastosowania sztucznej inteligencji we współczesnym rolnictwie i ma na celu wsparcie rolnictwa precyzyjnego, m.in. poprzez wykorzystanie nowych osiągnięć AI w monitorowaniu pól uprawnych, z użyciem robota polowego. Drugim z projektów, w którym zastosowanie znalazły wyniki prac opisanych w niniejszej rozprawie, jest projekt Fitoexport, realizowany przez Sieć Badawczą Łukasiewicz – Instytut Lotnictwa i finansowany w ramach programu GOSPOSTRATEG przez Narodowe Centrum Badań i Rozwoju, a rezultaty tego projektu odpowiadają idei zrównoważonego rolnictwa oraz rolnictwa precyzyjnego. W obu przypadkach, wyniki przedstawione w niniejszej rozprawie doktorskiej, dzięki integracji zaawansowanych algorytmów sztucznej inteligencji z nowoczesnymi technologiami obrazowania, mogą przyczynić się do wzrostu wydajności produkcji rolnej, optymalizacji procesów agrotechnicznych oraz lepszego zarządzania zasobami środków ochrony roślin, nawożenia i nawodnienia, doskonale wpisując się w idee nowoczesnego rolnictwa precyzyjnego.



## 1.2. Cele pracy

Na podstawie przedstawionego kontekstu pracy i uzasadnienia podjęcia tematu, sformułowane zostały konkretne cele naukowe. Niniejsza praca ma za zadanie zaproponowanie rozwiązań dla systemu robota polowego, wykorzystujących uczenie głębokie oraz analizę obrazów RGB i wielospektralnych, które pozwolą na wsparcie procesu automatycznego określania kondycji roślin uprawnych, a w szczególności kukurydzy.

Główne cele pracy są zatem następujące:

- Przeprowadzenie badań z wykorzystaniem głębokich sieci neuronowych w celu analizy obrazów wielospektralnych i RGB na potrzeby opracowania rozwiązania służącego do automatycznej detekcji i klasyfikacji faz rozwoju kukurydzy w skali BBCH. Wybór najskuteczniejszego rozwiązania w oparciu o wybrany algorytm uczenia głębokiego oraz wybrany typ zobrazowania, które w połączeniu ze sobą pozwalają osiągnąć najlepsze wyniki.
- Przeprowadzenie badań z wykorzystaniem głębokich sieci neuronowych w celu analizy obrazów wielospektralnych i RGB na potrzeby opracowania rozwiązania służącego do automatycznej detekcji i klasyfikacji poziomów nawodnienia kukurydzy. Wybór najskuteczniejszego rozwiązania w oparciu o wybrany algorytm uczenia głębokiego oraz wybrany typ zobrazowania, które w połączeniu ze sobą pozwalają osiągnąć najlepsze wyniki.
- Przeprowadzenie badań z wykorzystaniem głębokich sieci neuronowych w celu analizy obrazów RGB na potrzeby opracowania rozwiązania służącego do automatycznej detekcji i klasyfikacji porażenia kukurydzy wskazanymi patogenami. Wybór najskuteczniejszego rozwiązania w oparciu o wybrany algorytm uczenia głębokiego oraz wybrany typ zobrazowania, które w połączeniu ze sobą pozwalają osiągnąć najlepsze wyniki.
- Stworzenie dedykowanych autorskich zbiorów danych (podzielonych następnie na zbiory treningowe, walidacyjne i testowe), składających się z oznaczonych obrazów RGB i wielospektralnych, służących do realizacji zadań związanych z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH oraz detekcją i klasyfikacją poziomów nawodnienia kukurydzy. Wykorzystanie do tego celu własnego poletka testowego.

### 1.3. Struktura rozprawy

Struktura niniejszej rozprawy doktorskiej jest następująca:

- **Rozdział 2** zawiera przegląd literatury związanej z tematyką rozprawy oraz opis wykorzystywanych technik. Zaprezentowano w nim informacje dotyczące teorii sztucznych sieci neuronowych i uczenia głębokiego, które były kluczowym narzędziem podczas realizacji badań. Następnie omówiono podstawy przetwarzania obrazów RGB i wielospektralnych, a także dokonano opisu oceny jakości modeli i użytych technik walidacyjnych. W kolejnym kroku skupiono się na opisie zastosowania skali BBCH, nawodnienia i identyfikacji patogenów w analizie obrazów, a na koniec tego rozdziału dokonano przeglądu literatury pod kątem istniejących rozwiązań oraz wyciągnięto wnioski i dokonano identyfikacji luk badawczych, które były inspiracją do powstania niniejszej pracy.
- **Rozdział 3** skupia się na materiałach i metodach badawczych wykorzystywanych w ramach prowadzonych prac. Dokonano rzetelnego opisu przyjętej metodyki przygotowywania danych i prowadzenia badań, opisano zbiory przygotowanych i wykorzystywanych danych obrazowych, przybliżono szczegóły implementacji architektur głębokich sieci neuronowych, omówiono proces trenowania modeli oraz cały proces detekcji i klasyfikacji obiektów z uwzględnieniem zastosowania modeli uczenia głębokiego oraz specjalnie opracowanego klasyfikatora głosującego dla danych wielospektralnych i jego zastosowania w omawianej pracy.
- **Rozdział 4** poświęcony został opisowi realizacji i wyników zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy w skali BBCH z wykorzystaniem obrazowania RGB i obrazowania z kamery wielospektralnej RedEdge. Przedstawiono schemat opracowanego rozwiązania, wyniki uzyskane dla danych wielospektralnych, wyniki dla danych RGB oraz wyniki uzyskane przy zastosowaniu klasyfikatora głosującego. Omówiono również zarejestrowane krzywe uczenia oraz porównano uzyskane wyniki i wybrano najlepsze rozwiązanie.
- **Rozdział 5** jest analogiczny do poprzedniego i zawiera opis realizacji i wyników zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH z wykorzystaniem obrazowania RGB i kamery wielospektralnej MapIR. Przedstawiono schemat opracowanego rozwiązania, wyniki uzyskane dla danych

wielospektralnych oraz wyniki dla danych RGB. Omówiono także zarejestrowane krzywe uczenia oraz porównano uzyskane wyniki i wybrano najlepsze rozwiązanie.

- **Rozdział 6** dotyczy detekcji i klasyfikacji poziomów nawodnienia kukurydzy. Zaprezentowano w nim schemat zaproponowanego rozwiązania, wyniki uzyskane dla danych wielospektralnych i danych RGB, krzywe uczenia zarejestrowane podczas procesu trenowania sztucznych sieci neuronowych, a także porównanie uzyskanych wyników i wybór najlepszych rozwiązań.
- **Rozdział 7** zawiera informacje na temat realizacji zadania związanego z detekcją i klasyfikacją porażenia kukurydzy wybranymi patogenami. Omówiono schemat zaproponowanego rozwiązania, uzyskane wyniki i ich analizę, omówienie zarejestrowanych krzywych uczenia oraz porównanie wyników i wybór najlepszego rozwiązania.
- **Rozdział 8** prezentuje szczegóły dotyczące stworzonej aplikacji oraz zawiera przykładowe wyniki opracowanego rozwiązania uzyskane na danych obrazowych. Wyniki są prezentowane zarówno w formie pokazowej aplikacji, jak i pliku wyjściowego w formacie .csv agregującego wyniki z działania opracowanego zestawu sztucznych sieci neuronowych testowanych na przykładowych obrazach kukurydzy.
- **Rozdział 9** to podsumowanie całej rozprawy wraz z omówieniem głównych osiągnięć pracy i jej oryginalnym wkładem, a także wnioskami płynącymi z przeprowadzonych eksperymentów i rekomendacjami dla przyszłych badań.
- **Załącznik A** zawiera publikacje autorki niniejszej rozprawy, informacje o złożonym wniosku patentowym, dokument wyróżnienia uzyskany na międzynarodowej konferencji naukowej FedCSIS za prezentację ściśle związaną z tematem doktoratu oraz artykuł popularnonaukowy z portalu Farming Future Food, w którym redakcja opisuje wyniki zaprezentowane przez autorkę niniejszej rozprawy w artykule naukowym z konferencji FedCSIS
- **Załącznik B** zawiera specyfikacje techniczne wykorzystywanych kamer wielospektralnych.



## 2. Przegląd literatury i wykorzystywanych technik

W niniejszym rozdziale przedstawiono przegląd literatury i technologii wykorzystywanych jako podstawa teoretyczna dla rozwiązania będącego przedmiotem niniejszej rozprawy. Szczególną uwagę poświęcono sztucznym sieciom neuronowym. Omówiono podstawy teoretyczne dotyczące konwolucyjnych sieci neuronowych oraz sieci typu transformer wraz z podstawą ich działania oraz zastosowaniem w tematyce przetwarzania i analizy obrazów. Przybliżono architektury głębokich sieci neuronowych wykorzystanych w ramach zrealizowanych prac projektowych, w tym takie architektury jak ResNet50, ResNet101, ResNeXt101 oraz zaawansowane architektury jak HTC, Mask2Former, ConvNeXt, EfficientNet, Vision Transformer oraz Swin Transformer.

W dalszej części rozdziału omówiono techniki przetwarzania obrazów RGB i wielospektralnych, z uwzględnieniem ich charakterystyki i metod przygotowania do dalszej analizy. Uwadze zostały poświęcone również takie zagadnienia jak metryki oceny jakości modeli i techniki walidacyjne, które są kluczowym elementem oceny efektywności badanych architektur.

W kolejnej części przeglądu literatury i wykorzystywanych technik omówiono zagadnienia spoza dyscypliny naukowej, której dotyczy niniejsza rozprawa. Przybliżono informacje na temat międzynarodowej skali rozwoju roślin BBCH oraz zagadnienia związane z nawodnieniem roślin i identyfikacją patogenów w kontekście analizy obrazów. Mają one szczególne znaczenie dla kontekstu niniejszej rozprawy, gdyż skupia się ona na automatycznym określaniu tychże parametrów kondycji roślin na podstawie obrazów.

Rozdział spina przegląd podobnych prac badawczych wraz z analizą wykorzystywanych rozwiązań, co było wyjściem do podsumowania kluczowych wniosków oraz identyfikacji luk badawczych i propozycji dalszych badań.

### 2.1. Sztuczne sieci neuronowe i uczenie głębokie w klasyfikacji obiektów

Uczenie głębokie jest jednym z głównych obszarów uczenia maszynowego. Jego celem jest efektywne trenowanie wielowarstwowych sztucznych sieci neuronowych, a z faktu wielowarstwowości wynika epitet „głębokie” odnoszący się do tego typu trenowania sieci. W niniejszym podrozdziale omówione zostały najważniejsze pojęcia związane ze sztucznymi

sieciami neuronowymi, które są podstawą uczenia głębokiego. Jest to punkt wyjścia do dalszej analizy problemu rozpoznawania i klasyfikacji obiektów na obrazach [Rashka, 2018].

Podstawowymi elementami wielowarstwowych sieci neuronowych są sztuczne neurony, a ich koncepcje bazują na modelach inspirowanych funkcjonowaniem ludzkiego mózgu, który rozwiązuje złożone problemy za pomocą połączeń neuronowych. Teoretyczne podstawy modeli sieci neuronowych zostały opracowane już w latach czterdziestych XX wieku. Wówczas Warren McCulloch i Walter Pitts jako pierwsi opisali model działania komórki nerwowej [Rashka, 2018].

Neurony są wzajemnie połączonymi komórkami nerwowymi, odpowiedzialnymi za przetwarzanie i przesyłanie sygnałów chemicznych i elektrycznych. Do wejścia neuronu, zwanego dendrytem, dociera wiele sygnałów, które następnie są integrowane w ciele komórki. Jeśli energia impulsu przekroczy określoną wartość progową, generowany jest sygnał wyjściowy, który jest przekazywany do wyjścia neuronu, czyli aksonu. McCulloch i Pitts opisali komórkę nerwową jako prostą bramkę logiczną, zawierającą binarne wyjścia [Rashka, 2018] [Zocca i in., 2018].

### 2.1.1. Wprowadzenie do sztucznych sieci neuronowych

Wprowadzenie do tematu sztucznych sieci neuronowych należy rozpocząć od przedstawienia podstaw koncepcji perceptronu, które opierają się na modelu neuronu zaproponowanym przez Warrena S. McCullocha i Waltera Pittsa, którzy na łamach *Bulletin of Mathematical Biophysics* opublikowali w 1943 roku pracę pt. „A Logical Calculus of the Ideas Immanent in Nervous Activity”. Publikacja ta wprowadziła matematyczny model neuronu, który stał się następnie podstawą dla sztucznych sieci neuronowych. Na bazie tejże publikacji Frank Rosenblatt opracował pierwszą słynną koncepcję reguły uczenia perceptronu [Rosenblatt, 1958]. Dzięki zastosowaniu tej reguły, stworzył następnie algorytm umożliwiający automatyczne uczenie się poprzez optymalizację współczynników wag, które są mnożone przez wartości wejściowe, co pozwala na określenie, czy neuron przekaże dalej sygnał. Z punktu widzenia uczenia nadzorowanego i klasyfikacji, perceptron może być stosowany do przewidywania przynależności próbek do poszczególnych klas [Rashka, 2018].

W sposób matematyczny, problem ten można ująć w postaci klasyfikacji binarnej, w której rozróżnia się dwie klasy: 1 (pozytywną) oraz -1 (negatywną). Aby dokonać klasyfikacji,

definiuje się funkcję aktywacji  $\phi(z)$ , która jest kombinacją wartości wejściowych  $x$  i powiązanego wektora wag  $w$ , gdzie  $z$  jest całkowitym pobudzeniem układu:

$$z = w_1x_1 + \dots + w_mx_m$$

Wektory te mają następującą postać:

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

W przypadku, gdy pobudzenie  $z$  dla wybranej próbki  $x^{(i)}$  przekracza założoną wartość progową  $\theta$ , można zaliczyć obiekt do klasy pozytywnej, a w przeciwnym razie do klasy negatywnej. W perceptronie funkcja aktywacji  $\phi(\cdot)$  jest funkcją skoku jednostkowego:

$$\phi(z) = \begin{cases} 1 & \text{jeśli } z \geq \theta \\ -1 & \text{jeśli } z < \theta \end{cases}$$

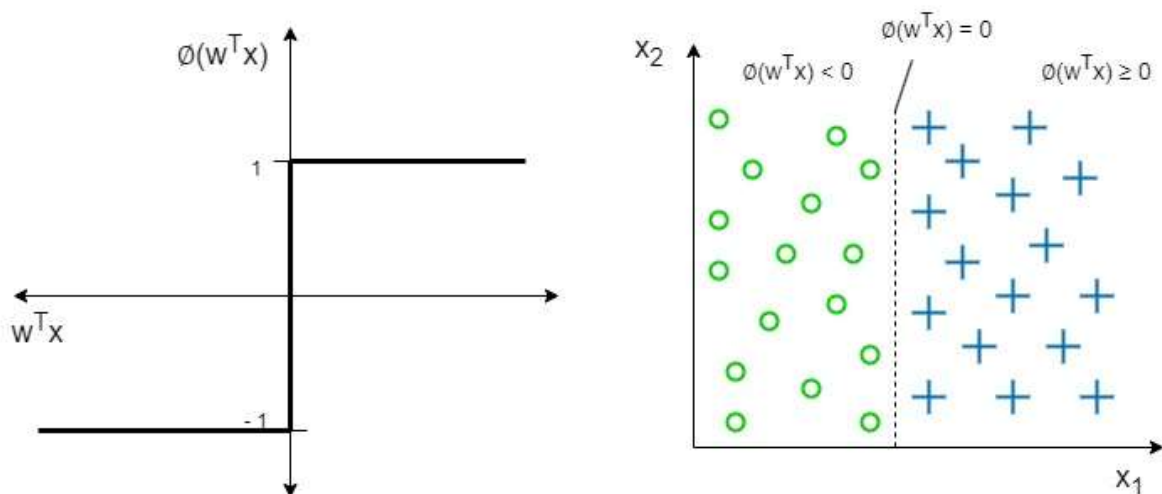
Aby uprościć zapis, należy wartość progową  $\theta$  przenieść na lewą stronę równania, określając początkową wagę w postaci  $w_0 = -\theta$  oraz  $x_0 = 1$ . Wówczas całkowite pobudzenie  $z$  można zapisać w następującej formie:

$$z = w_0x_0 + w_1x_1 + \dots + w_mx_m = w^T x,$$

zakładając, że:

$$\phi(z) = \begin{cases} 1 & \text{jeśli } z \geq 0 \\ -1 & \text{jeśli } z < 0 \end{cases}$$

Rysunek 2.1. ilustruje, jak całkowite pobudzenie  $z = w^T x$  jest przetwarzane do wartości binarnych (1 lub -1) za pomocą funkcji aktywacji perceptronu oraz jak może zostać użyte w celu oddzielenia dwóch klas odrębnych liniowo [Rashka, 2018].



**Rysunek 2.1.** Całkowite pobudzenie zastosowane w uczenia maszynowym, (rysunek wykonany na podstawie: [Rashka, 2018]).

Głównym założeniem neuronu McCullocha-Pittsa oraz modelu perceptronu progowego jest wdrożenie prostego mechanizmu naśladującego działanie komórki nerwowej, która może być aktywowana lub pozostawiona w stanie nieaktywowanym. Dzięki temu uproszczonemu mechanizmowi reguła uczenia perceptronu, opracowana przez Rosenblatta, charakteryzuje się prostotą. Można ją opisać w następujących krokach:

1. Inicjalizacja wag z wartościami 0 lub losowymi, małymi wartościami.
2. Wykonanie następujących czynności dla każdej próbki uczącej  $x^{(i)}$ :
  - obliczenie wartości wyjściowej  $\hat{y}$ .
  - aktualizacja wag.

Wartością otrzymywaną na wyjściu jest etykieta klasy prognozowana przez określoną wcześniej funkcję skoku jednostkowego. Zmianę każdej wagi  $w_j$  w wektorze wag  $w$  można wyrazić jako:

$$w_j := w_j + \Delta w_j$$

Aktualizacja wagi  $\Delta w_j$  jest obliczana zgodnie z regułą uczenia perceptronu:

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)}$$

gdzie:

$\eta$  - współczynnik uczenia (jest wielkością stałą i przyjmuje wartości od 0 do 1)



$y^{(i)}$  - rzeczywista etykieta klas i-tej próbki uczącej

$\hat{y}^{(i)}$  - prognozowana etykieta klas [Rashka, 2018].

Warto zaznaczyć, że wszystkie wagi w wektorze są odświeżane równocześnie, więc nie można przeliczać powtórnie wartości  $y^{(i)}$ , aż nie zostaną odświeżone wszystkie wagi  $\Delta w_j$ .

Uaktualnienia dla dwuwymiarowego zbioru danych można określić jako:

$$\Delta w_0 = \eta(y^{(i)} - \text{wyjście}^{(i)})$$

$$\Delta w_1 = \eta(y^{(i)} - \text{wyjście}^{(i)})x_1^{(i)}$$

$$\Delta w_2 = \eta(y^{(i)} - \text{wyjście}^{(i)})x_2^{(i)}$$

Przedstawiona reguła uczenia jest prosta i zawiera się w dwóch scenariuszach, w których perceptron we właściwy sposób przewiduje etykiety klas, a wagi nie ulegają zmianie:

$$\Delta w_j = \eta(-1 - -1)x_j^{(i)} = 0$$

$$\Delta w_j = \eta(1 - 1)x_j^{(i)} = 0$$

W sytuacji błędnego przewidywania wagi są modyfikowane w kierunku pozytywnej bądź negatywnej klasy docelowej, zgodnie z następującymi równaniami:

$$\Delta w_j = \eta(1 - -1)x_j^{(i)} = \eta(2)x_j^{(i)}$$

$$\Delta w_j = \eta(-1 - 1)x_j^{(i)} = \eta(-2)x_j^{(i)}$$

W celu lepszego zrozumienia idei mnożnika  $x_j^{(i)}$ , należy przyjrzeć się następującemu przykładowi:

$$y^{(i)} = +1$$

$$\hat{y}^{(i)} = -1$$

$$\eta = 1$$

Zakładając, że  $x_j^{(i)} = 0,5$ , a próbka ta została błędnie sklasyfikowana jako  $-1$ , to w takiej sytuacji należy zwiększyć wagę o 1, a wówczas całkowite pobudzenie będzie silniejsze,

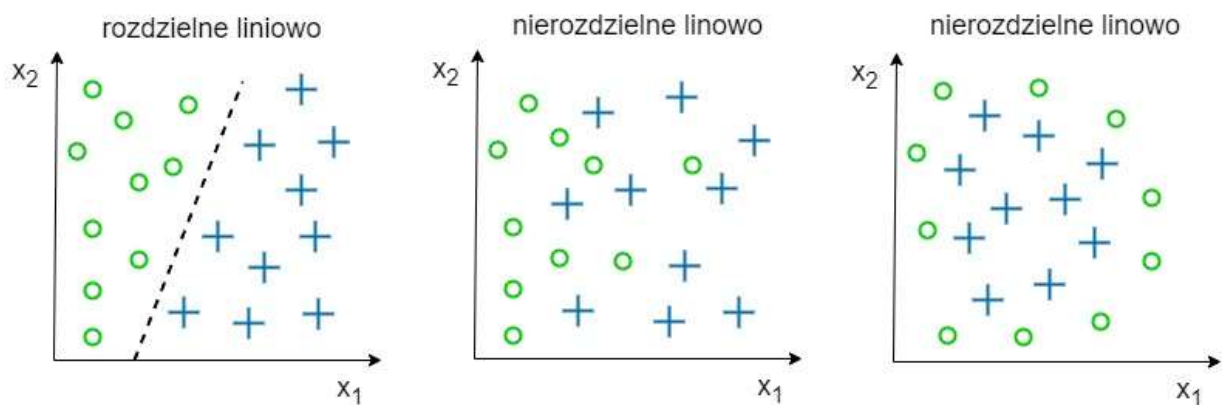
gdy ta próbka będzie napotkana ponownie. W rezultacie badany obiekt z większym prawdopodobieństwem zostanie przypisany do klasy +1 po przekroczeniu wartości progowej funkcji skokowej:

$$\Delta w_j^{(i)} = (1 - -1)0,5 = (2)0,5 = 1$$

Zmiana wagi jest wprost proporcjonalna do wartości  $x_j^{(i)}$ . Zakładając, że kolejna próbka  $x_j^{(i)} = 2$  została błędnie sklasyfikowana jako  $-1$ , można w takiej sytuacji przesunąć granicę decyzyjną jeszcze bardziej w celu poprawnej klasyfikacji próbki następnym razem:

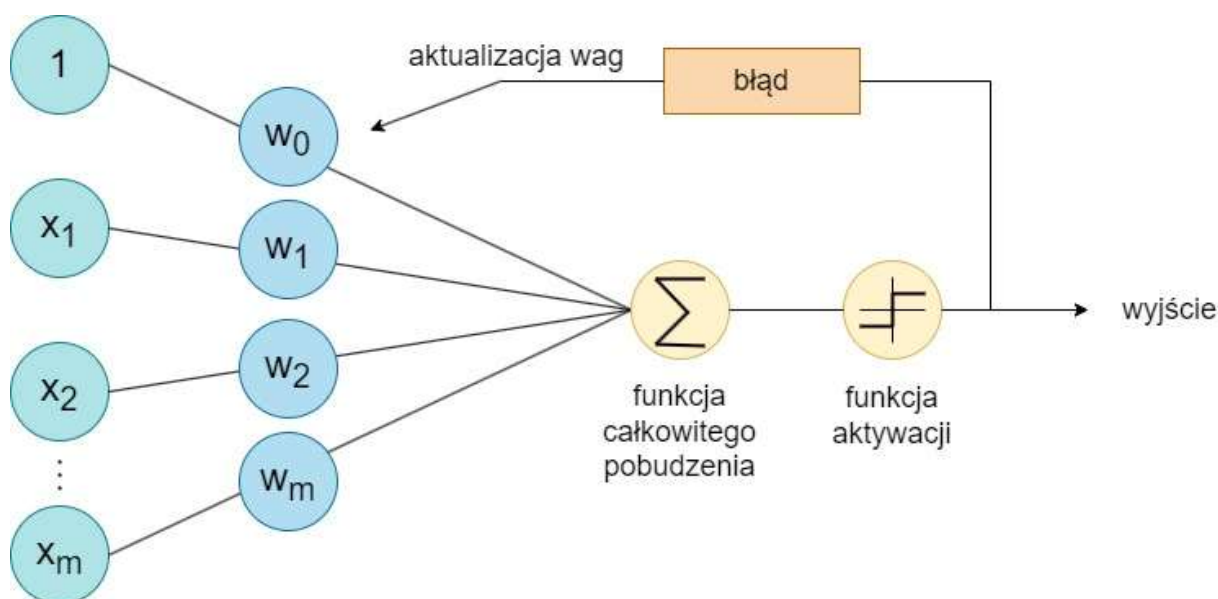
$$\Delta w_j = (1 - -1)2 = (2)2 = 4$$

Zbieżność algorytmu perceptronu jest gwarantowana jedynie w przypadku, gdy dwie klasy są liniowo separowalne (jak pokazano na poniższym rysunku), a współczynnik uczenia jest odpowiednio niski. Jeśli nie jest możliwe rozdzielenie tych klas za pomocą liniowej granicy decyzyjnej, można ustalić maksymalną liczbę iteracji (tzw. epok) algorytmu na danych treningowych oraz/lub ustalić próg akceptowalnych błędów klasyfikacji. W przeciwnym razie, perceptron mógłby w nieskończoność aktualizować wagi [Rashka, 2018].



**Rysunek 2.2.** Rozdzielczość liniowa klas, (rysunek wykonany na podstawie: [Rashka, 2018]).

Ogólny model perceptronu można przedstawić za pomocą poniższego schematu:



**Rysunek 2.3.** Ogólny model perceptronu, (rysunek wykonany na podstawie: [Rashka, 2018]).

Rysunek 2.3 przedstawia schemat działania perceptronu, który otrzymuje dane wejściowe próbek  $x$  i łączy je z wagami  $w$  w celu obliczenia funkcji całkowitego pobudzenia. Wynik tej operacji przekazywany jest następnie do funkcji aktywacji (w tym przypadku funkcji skoku jednostkowego), która generuje binarną wartość  $-1$  lub  $+1$ , odpowiadającą przewidywanej etykiecie klasy dla danej próbki. Podczas procesu uczenia wyniki te są wykorzystywane do obliczenia błędu predykcji i odpowiedniego dostosowania wag [Rashka, 2018].

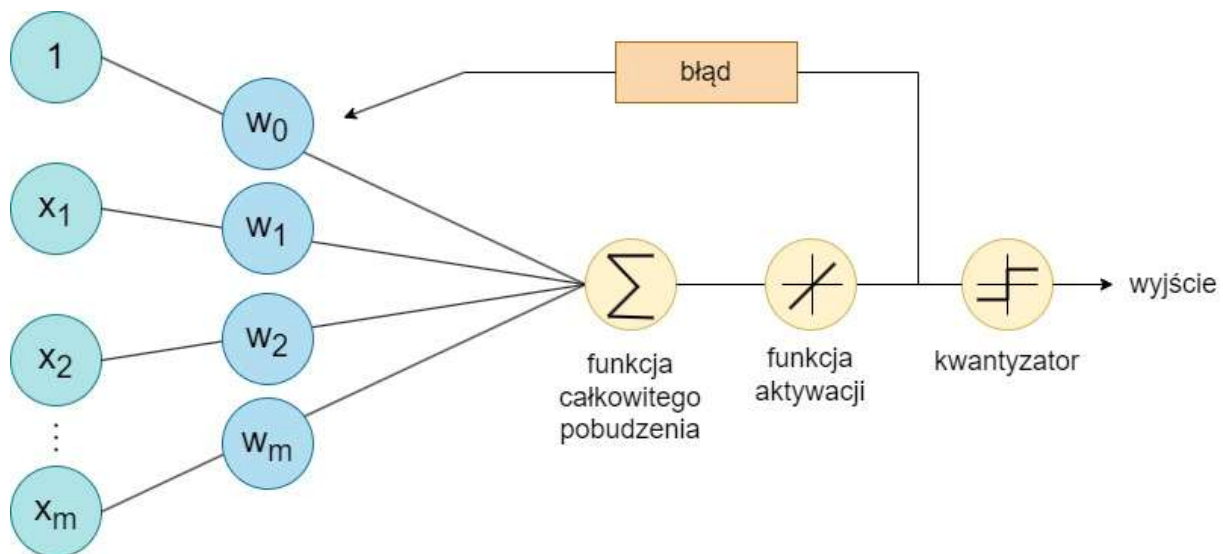
Inną odmianą jednowarstwowej sieci neuronowej jest Adaptacyjny Liniowy Neuron (w skrócie Adaline), który odgrywa istotną rolę w rozpoznawaniu obrazu. Algorytm ten został wprowadzony kilka lat po perceptronie Rosenblatta przez Bernarda Widrowa i Tedda Hoffa. Adaline można traktować jako rozszerzenie idei perceptronu [Widrow i in., 1960]. Wprowadza on kluczową koncepcję definiowania oraz minimalizacji funkcji kosztu, co stanowi fundament dla bardziej zaawansowanych algorytmów klasyfikacyjnych, takich jak regresja logistyczna, maszyny wektorów nośnych oraz modele regresji [Rashka, 2018] [Zocca i in., 2018].

Kluczową różnicą pomiędzy regułą uczenia Adaline a perceptronem jest sposób, w jaki dokonuje się aktualizacji wag: w Adaline wagi są modyfikowane na podstawie liniowej funkcji aktywacji, natomiast w perceptronie opierają się one na funkcji skoku jednostkowego. W

modelu Adaline liniowa funkcja aktywacji jest tożsama z całkowitym pobudzeniem, co oznacza, że nie stosuje się dyskretnych progów, jak w perceptronie:

$$\phi(w^T x) = w^T x$$

Liniowa funkcja aktywacji jest wykorzystywana do obliczania wag, natomiast kwantyzator, zbliżony do funkcji skoku jednostkowego, jest używany do prognozowania etykiet klas, jak pokazano na poniższym rysunku:



**Rysunek 2.4.** Schemat modelu Adaline, (rysunek wykonany na podstawie: [Rashka, 2018]).

Porównując schemat perceptronu ze schematem Adaline, główna różnica polega na tym, że Adaline wykorzystuje ciągłe wartości, obliczane przy pomocy liniowej funkcji aktywacji, do wyznaczania błędu modelu i aktualizacji wag, zamiast binarnych etykiet klas, jak ma to miejsce w perceptronie.

Porównując schemat perceptronu ze schematem Adaline, główna różnica polega na stosowaniu ciągłych wartości, obliczanych za pomocą liniowej funkcji aktywacji, do wyznaczania błędu modelu i aktualizacji wag, zamiast binarnych etykiet klas. W kontekście tego procesu, kluczowe znaczenie ma optymalizacja funkcji kosztu, co prowadzi do zagadnienia metody gradientu prostego i minimalizacji funkcji kosztu. Jednym z kluczowych zadań w algorytmach uczenia nadzorowanego jest określenie funkcji celu, która zostanie zoptymalizowana w trakcie procesu uczenia. Funkcja ta pełni rolę funkcji kosztu, której minimalizacja jest niezbędna. W przypadku modelu Adaline funkcję kosztu  $J$  można

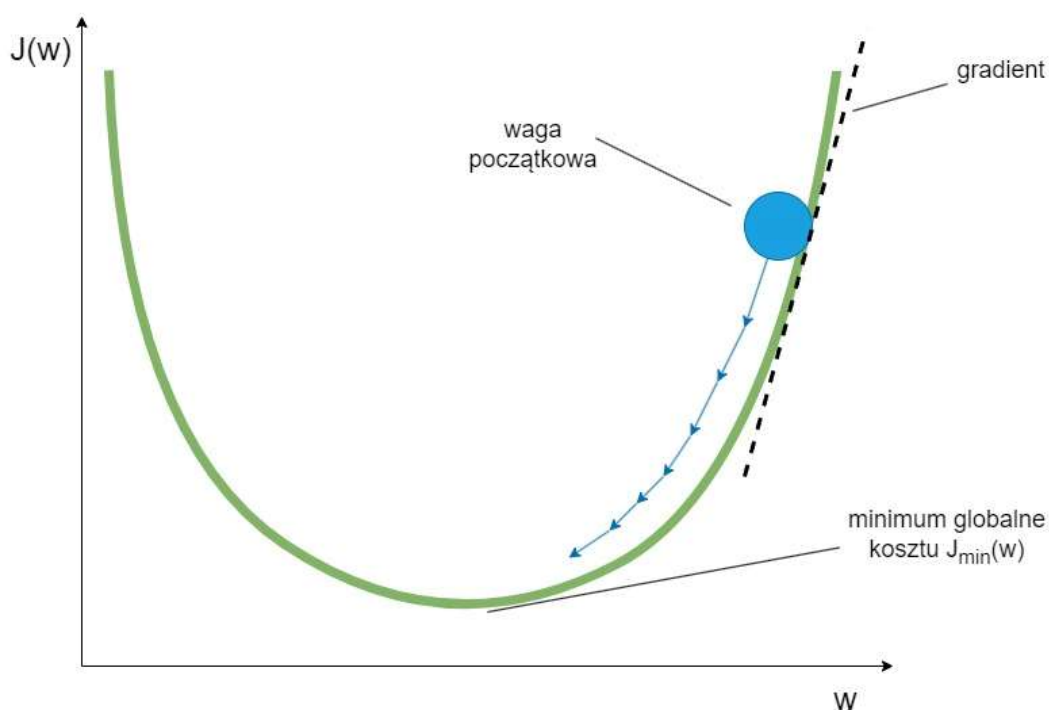


zdefiniować jako sumę kwadratów błędów (SSE, Sum of Squared Errors) pomiędzy przewidywanymi wynikami a rzeczywistymi etykietami klas. To zależność można wyrazić za pomocą następującego wzoru [Rashka, 2018]:

$$J(w) = \frac{1}{2} \sum (y^{(i)} - \phi(z^{(i)}))^2$$

Dla ułatwienia wyprowadzenia gradientu, dodano współczynnik  $\frac{1}{2}$ . Kluczową zaletą liniowej funkcji aktywacji, w porównaniu do funkcji skoku jednostkowego, jest możliwość różniczkowania funkcji kosztu. Dodatkowo, inną istotną cechą jest wypukłość tej funkcji, co pozwala na wykorzystanie prostego, ale efektywnego algorytmu optymalizacji, jakim jest metoda gradientu prostego (ang. *gradient descent*). Algorytm ten umożliwia wyznaczenie wag, które minimalizują funkcję kosztu, a tym samym klasyfikują próbki zawarte w zestawie danych [Rashka, 2018].

Na poniższym rysunku zilustrowano, jak algorytm gradientu prostego można porównać do procesu schodzenia z góry aż do momentu osiągnięcia lokalnego lub globalnego minimum funkcji kosztu. W każdej iteracji algorytm "schodzi" coraz niżej, a wielkość kolejnych kroków zależy zarówno od wartości współczynnika uczenia, jak i od nachylenia gradientu [Rashka, 2018].



**Rysunek 2.5.** Poglądowy schemat działania algorytmu gradientu prostego, (rysunek wykonany na podstawie: [Rashka, 2018]).

Korzystając z metody gradientu prostego, wagi mogą zostać zaktualizowane poprzez ruch w przeciwnym kierunku do gradientu  $\nabla J(w)$  funkcji kosztu  $J(w)$  zgodnie z poniższym równaniem:

$$w := w + \Delta w$$

Zmianę wagi  $\Delta w$  zdefiniowano jako ujemny gradient pomnożony przez współczynnik uczenia  $\eta$  zgodnie z poniższym równaniem:

$$\Delta w = -\eta \nabla J(w)$$

W celu wyliczenia gradientu funkcji kosztu należy obliczyć pochodną cząstkową tejże funkcji przy uwzględnieniu każdej z wag  $w_j$

$$\frac{\partial J}{\partial w_j} = - \sum_i (y^{(i)} - \phi(z^{(i)})) x_j^{(i)}$$

W równaniu tym  $y^{(i)}$  oznacza docelową etykietę klas danej próbki  $x^{(i)}$ , a  $\phi(z^{(i)})$  jest aktywacją neuronu, czyli funkcją liniową w specjalnym przypadku modelu Adaline. Dzięki temu można zapisać aktualizację wagi  $w_j$  jako:

$$\Delta w_j = -\eta \sum_i (y^{(i)} - \phi(z^{(i)})) x_j^{(i)}$$

Jednocześnie są aktualizowane wszystkie wagi, więc reguła uczenia Adaline wygląda następująco:

$$w := w + \Delta w$$

Pomimo faktu, że reguła uczenia w modelu Adaline wygląda tak samo jak dla perceptronu, to wartość funkcji  $\phi(z^{(i)})$ , gdzie  $z^{(i)} = w^T x^{(i)}$ , jest liczbą rzeczywistą, a nie liczbą całkowitą etykiety klas. Dodatkowo, aktualizacja wag jest wyliczana na bazie wszystkich próbek ze zbioru uczącego. Wagi nie są zatem aktualizowane przyrostowo po każdej próbce. Z tego względu technika ta jest nazywana również metodą gradientu prostego wsadowego, znaną jako *batch gradient descent* [Rashka, 2018].

Po opracowaniu w latach pięćdziesiątych modelu perceptronu przez Rosenblatta, bazującego na modelu McCollucha-Pittsa, zainteresowanie tym tematem znacząco zmalało ze względu na trudności związane z opracowaniem efektywnej metody uczenia wielowarstwowych sieci neuronowych. Przełom nastąpił w 1986 roku, kiedy David E. Rumelhart, Geoffrey E. Hinton i Ronald J. Williams odkryli na nowo oraz rozpowszechnili

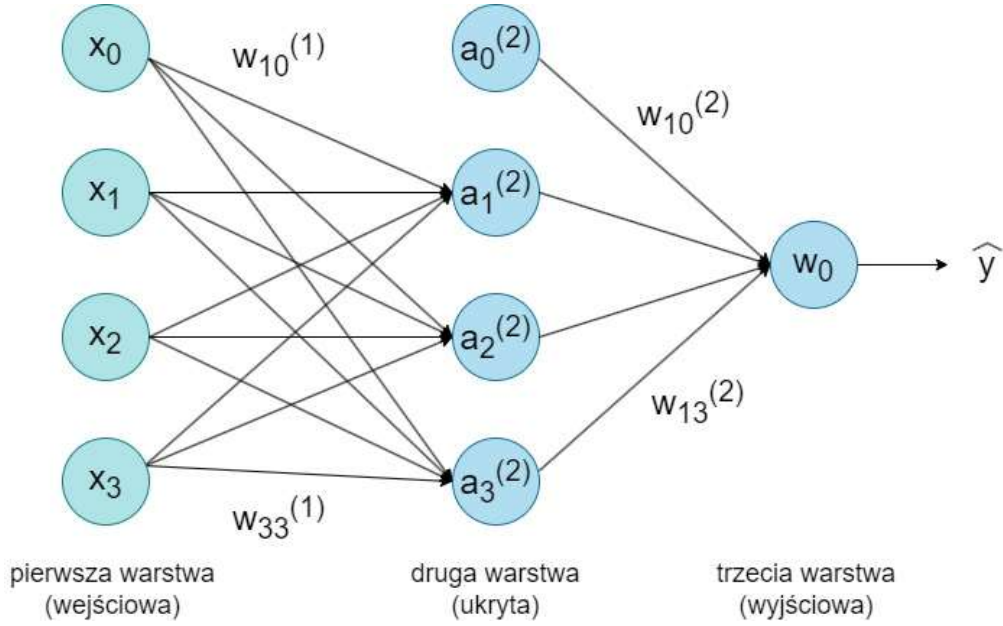
algorytm wstecznej propagacji błędów (ang. backpropagation), który umożliwił skuteczniejsze trenowanie sieci neuronowych [Rumelhart i in., 1986] [Rashka, 2018].

W ostatnich latach dokonano wielu przełomowych odkryć, które doprowadziły do rozwoju algorytmów uczenia głębokiego. Algorytmy te umożliwiają tworzenie detektorów cech (tzw. feature detectors) na podstawie nieoznakowanych danych, co pozwala na wstępne trenowanie głębokich sieci neuronowych składających się z wielu warstw. Poza istotnym wkładem naukowym, sieci te budzą ogromne zainteresowanie w sektorze gospodarczym [Rashka, 2018]. Zaawansowane sieci neuronowe wspomagane algorytmami uczenia głębokiego są uznawane za najnowocześniejsze technologie służące do rozwiązywania skomplikowanych problemów, w tym m.in. rozpoznawania obrazów, co jest tematem niniejszej rozprawy.

Dla zrozumienia tematu rozpoznawania obrazu należy przybliżyć kwestię wielowarstwowej architektury sieci neuronowych. Ważną kwestią jest sprawa łączenia wielu pojedynczych neuronów w wielowarstwową jednokierunkową sieć neuronową (ang. *multi-layer feedforward neural network*). Jest to charakterystyczny przypadek sieci zwany wielowarstwowym perceptronem (ang. *multi-layer perceptron*, w skrócie MLP). Na rysunku 2.6. pokazano trójwarstwowy model MLP zawierający warstwę wejściową, ukrytą i wyjściową. Jednostki warstwy ukrytej są w pełni połączone z jednostkami warstwy wejściowej, a jednostki warstwy wyjściowej są połączone z jednostkami warstwy ukrytej. W przypadku, gdy model zawiera więcej niż jedną warstwę ukrytą, określa się go mianem głębokiej sieci neuronowej [Rashka, 2018].

Aby stworzyć głębszą architekturę sieci MLP należy dodać kolejne warstwy ukryte. Liczbę jednostek i warstw w sieci neuronowej można przyjąć za dodatkowe hiperparametry, które można optymalizować pod kątem określonego problemu z użyciem tzw. sprawdzianu krzyżowego.

W algorytmie wstecznej propagacji gradienty błędu maleją stopniowo wraz ze wzrostem liczby warstw w sieci neuronowej. Problem zanikającego gradientu (ang. vanishing gradient) znacząco utrudnia proces uczenia modelu. Aby temu zaradzić, opracowano specjalne algorytmy do wstępnego trenowania głębokich sieci neuronowych, co stanowi istotę uczenia głębokiego [Rashka, 2018]. Jest to kluczowe zagadnienie w kontekście rozpoznawania obrazów i tematu niniejszej rozprawy.



**Rysunek 2.6.** Model wielowarstwowej sieci neuronowej, (rysunek wykonany na podstawie: [Rashka, 2018]).

Na rysunku 2.6. oznaczono  $i$ -tą jednostkę aktywacji w  $l$ -tej warstwie jako  $a_i^{(l)}$ , zaś jednostki aktywacji  $a_0^{(1)}$  i  $a_0^{(2)}$  to tzw. jednostki obciążenia (ang. *bias units*), które mają wartość 1. Aktywacja jednostek w warstwie wejściowej obejmuje ich wartości wejściowe oraz jednostkę odpowiadającą za obciążenie. Poniższe wyrażenie przedstawia tę zależność:

$$a^{(1)} = \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ \vdots \\ a_m^{(1)} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ \vdots \\ x_m^{(i)} \end{bmatrix}$$

Każda z jednostek znajdujących się w warstwie  $l$  jest połączona ze wszystkimi jednostkami w warstwie  $l + 1$  przy użyciu współczynnika wag. Np. połączenie między  $k$ -tą jednostką w warstwie  $l$  z  $j$ -tą jednostką warstwy  $l + 1$  można zapisać jako  $w_{j,k}^{(l)}$ . Górny indeks  $i$  w wyrażeniu  $x_m^{(i)}$  oznacza  $i$ -tą próbkę.

Jedna jednostka w warstwie wyjściowej wystarcza do klasyfikacji binarnej próbek, jednak na rysunku 2.6 zaprezentowano uogólnioną postać wielowarstwowej sieci neuronowej, umożliwiającą wieloklasową klasyfikację poprzez zastosowanie techniki "jeden przeciw wszystkim" (OvA). Działanie tego modelu opiera się na tzw. gorącojedynkowej reprezentacji



zmiennych kategoryzujących. Przykładem takiej reprezentacji zerojedynkowej jest poniższy sposób kodowania trzech etykiet klas:

$$0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, 1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, 2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Tego rodzaju zerojedynkowa reprezentacja wektorowa umożliwia przeprowadzanie operacji klasyfikacyjnych dla dowolnej liczby unikatowych etykiet klas obecnych w zestawie danych uczących [Rashka, 2018].

Aktywacja sieci neuronowej za pomocą propagacji w przód odgrywa kluczową rolę w procesie wyliczania wyniku w modelu MLP. Mechanizm ten, zwany propagacją w przód (ang. *forward propagation*), pozwala na przeprowadzenie obliczeń w sieci neuronowej i jest istotnym etapem w nauce perceptronu wielowarstwowego. Proces uczenia takiego modelu można podzielić na trzy główne etapy. Pierwszym krokiem jest rozsyłanie wzorców danych uczących od warstwy wejściowej do warstw wyższych wzdłuż całej sieci, w celu uzyskania odpowiedzi. Następnie obliczany jest błąd na podstawie uzyskanego wyniku, który minimalizuje się za pomocą funkcji kosztu. Ostatnim etapem jest rozprzestrzenianie wstecz obliczonego błędu, wyznaczenie jego pochodnej z uwzględnieniem wszystkich wag oraz aktualizacja modelu w celu poprawy jego skuteczności [Rashka, 2018].

Po wielokrotnym wykonaniu tych operacji przez wiele epok i wytrenowaniu wag modelu MLP, propagacja w przód może być użyta do obliczenia wyników, a następnie do zastosowania funkcji progowej w celu uzyskania przewidywanych etykiet klas w formacie „gorącojedynkowym” [Rashka, 2018].

Warto przedstawić poszczególne etapy procesu propagacji w przód, które prowadzą do wygenerowania odpowiedzi na podstawie wzorców z danych uczących. Każda jednostka w warstwie ukrytej jest połączona ze wszystkimi jednostkami warstwy wejściowej, co umożliwia obliczenie aktywacji  $a_1^{(2)}$  zgodnie z poniższym wyrażeniem:

$$z_1^{(2)} = a_0^{(1)} w_{1,0}^{(1)} + a_1^{(1)} w_{1,1}^{(1)} + \dots + a_m^{(1)} w_{1,m}^{(1)}$$

$$a_1^{(2)} = \phi(z_1^{(2)})$$

gdzie:

$z_1^{(2)}$  - to pobudzenie całkowite

$\phi(\cdot)$  - to funkcja aktywacji.

Funkcja aktywacji musi być różniczkowalna, aby możliwe było zastosowanie metody gradientowej do uczenia wag łączących neurony. Aby umożliwić modelowi MLP rozpoznawanie obrazów, konieczne jest wprowadzenie nieliniowych funkcji aktywacji, takich jak np. funkcja aktywacji logistyczna (sinusoidalna):

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Funkcja sinusoidalna ma kształt krzywej o charakterystycznym kształcie litery "S", która odwzorowuje przekształcenie całkowitego pobudzenia  $z$  w rozkład logistyczny, mieszczący się w przedziale od 0 do 1. Krzywa ta przecina oś  $y$  w punkcie  $z = 0$ .

Model MLP jest typowym przykładem jednokierunkowej sieci neuronowej (ang. *feed forward*), w której przepływ danych zachodzi w jednym kierunku — każda warstwa służy jako wejście dla kolejnej, bez tworzenia zapętleń. W przeciwieństwie do tego, sieci rekurencyjne (ang. *recurrent neural networks*) pozwalają na powstawanie sprzężeń zwrotnych. Nazwa "wielowarstwowy perceptron" nie do końca oddaje specyfikę tego modelu, ponieważ jego sztuczne neurony zazwyczaj wykorzystują funkcje sinusoidalne, a nie perceptrony. Neurony w MLP bardziej przypominają jednostki regresji logistycznej, które zwracają wartości ciągłe w przedziale od 0 do 1.

Aby zwiększyć czytelność i efektywność kodu podczas implementacji, zaleca się zapisanie aktywacji w bardziej kompaktowej formie, wykorzystując podstawowe pojęcia algebry liniowej. Dzięki temu możliwe jest wektoryzowanie operacji, na przykład przy użyciu biblioteki NumPy w języku Python, co pozwala na uniknięcie złożonych i kosztownych obliczeniowo pętli *for*:

$$z^{(2)} = W^{(1)}a^{(1)}$$

$$a^{(2)} = \phi(z^{(2)})$$

Wyrażenie  $a^{(1)}$  oznacza  $[m_1] \times 1$  –wymiarowy wektor cech dla próbki  $x^{(i)}$  wraz z jednostką obciążenia. Wyrażenie  $W^{(1)}$  to z kolei  $h \times [m + 1]$  –wymiarowa macierz wag, gdzie  $h$  definiuje liczbę ukrytych jednostek w sieci neuronowej. Po realizacji mnożenia macierzowo-wektorowego wynikiem jest  $h \times 1$  wymiarowy wektor całkowitego pobudzenia  $z^{(2)}$  dający możliwość obliczenia aktywacji  $a^{(2)}$ , gdzie  $a^{(2)} \in \mathbb{R}^{h \times 1}$ . Uogólnienie tego na wszystkie  $n$  próbek uczących wygląda następująco:

$$Z^{(2)} = W^{(1)}[A^{(1)}]^T$$

gdzie:

$A^{(1)}$  - to  $n \times [m + 1]$  – wymiarowa macierz.

W wyniku pomnożenia dwóch macierzy wynikiem będzie  $h \times n$  – wymiarowa macierz całkowitego pobudzenia  $Z^{(2)}$ . Na koniec należy przemnożyć wszystkie wartości macierzy całkowitego pobudzenia przez funkcję aktywacji  $\phi(\cdot)$  i wyliczyć w ten sposób  $h \times n$  – wymiarową macierz aktywacji  $A^{(2)}$  dla kolejnej warstwy, czyli w tym przypadku dla warstwy wyjściowej:

$$Z^{(2)} = \phi(Z^{(2)})$$

W podobny sposób można wyrazić zwektoryzowaną postać aktywacji warstwy wyjściowej:

$$Z^{(3)} = W^{(2)}A^{(2)}$$

Pomnożono tutaj  $t \times h$  - wymiarową macierz  $W^{(2)}$  ( $t$  to liczba jednostek wyjściowych) przez  $h \times n$  – wymiarową macierz  $A^{(2)}$  aby uzyskać  $h \times n$  – wymiarową macierz  $A^{(2)}$  aby uzyskać  $t \times n$  – wymiarową macierz  $Z^{(3)}$ , której kolumny tworzą wartości wyjściowe dla poszczególnych próbek [Rashka, 2018].

Sinusoidalna funkcja aktywacji służąca do generowania wynikowych wartości ciągłych wygląda następująco:

$$A^{(3)} = \phi(Z^{(3)}), A^{(3)} \in \mathbb{R}^{t \times n}$$

### 2.1.2. Uczenie głębokie i propagacja wsteczna

W procesie trenowania sztucznej sieci neuronowej w rozpoznawaniu obrazu niezwykle istotne jest zrozumienie kluczowych koncepcji optymalizacji wag, takich jak logistyczna funkcja kosztu oraz algorytm wstecznej propagacji. Te elementy są fundamentalne dla skutecznego trenowania modelu, szczególnie w kontekście zadania klasyfikacji obrazów.

Logistyczna funkcja kosztu opiera się na następującej zależności [Rashka, 2018]:

$$J(w) = - \sum_{i=1}^n y^{(i)} \log(a^{(i)}) + (1 - y^{(i)}) \log(1 - a^{(i)})$$

gdzie:

$a^{(i)}$  - sigmoidalną aktywacja  $i$ -tej jednostki w jednej z warstw.

Wartość  $a^{(i)}$  obliczana jest jako funkcja propagacji jednokierunkowej:

$$a^{(i)} = \phi(z^{(i)})$$

Aby zminimalizować ryzyko przetrenowania modelu, dodaje się wyrażenie regularyzacji, które zmniejsza efekt przetrenowania. Regularyzację można przeprowadzić na dwa sposoby: L2 i L1, które są definiowane następująco (jednostki obciążenia nie są regularyzowane):

$$L2 = \lambda \|W\|_2^2 = \lambda \sum_{j=1}^m w_j^2 \quad \text{oraz} \quad L1 = \lambda \|w\|_1 = \lambda \sum_{j=1}^m |w_j|$$

Po dodaniu regularyzacji L2 do logistycznej funkcji kosztu otrzymuje się następujące równanie:

$$J(w) = - \left[ \sum_{i=1}^n y^{(i)} \log(a^{(i)}) + (1 - y^{(i)}) \log(1 - a^{(i)}) \right] + \frac{\lambda}{2} \|w\|_2^2$$

W przypadku klasyfikacji wieloklasowej model MLP (Multi-Layer Perceptron) generuje wektor wyjściowy zawierający  $t$  elementów, które są porównywane z wektorem docelowym o wymiarach  $t \times 1$ , przekształconym do formatu "gorącojedynkowego". Na przykład, aktywacja trzeciej warstwy i przypisanie docelowej klasy (w tym przypadku klasy 2) może być przedstawiona w następujący sposób:

$$a^{(3)} = \begin{bmatrix} 0,1 \\ 0,9 \\ \vdots \\ 0,3 \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Funkcja kosztu (bez wyrażenia regularyzacji) wygląda w ten sposób:

$$J(w) = - \sum_{i=1}^n \sum_{j=1}^t y_j^{(i)} \log(a_j^{(i)}) + (1 - y_j^{(i)}) \log(1 - a_j^{(i)})$$

Uogólnione wyrażenie regularyzacji, sprowadza się do zsumowania wszystkich wag w warstwie  $l$  (z wyjątkiem jednostki obciążenia), które zostały dodane do pierwszej kolumny:



$$J(w) = - \left[ \sum_{i=1}^n \sum_{j=1}^t y_j^{(i)} \log(\phi(z_j^{(i)})) + (1 - y_j^{(i)}) \log(1 - \phi(z_j^{(i)})) \right] + \frac{\lambda}{2} \sum_{l=1}^{L-1} \sum_{i=1}^{u_l} \sum_{j=1}^{u_{l+1}} (w_{j,i}^{(l)})^2$$

Kara wyliczona z regularyzacji L2 przedstawia się za pomocą poniższego wyrażenia:

$$\frac{\lambda}{2} \sum_{l=1}^{L-1} \sum_{i=1}^{u_l} \sum_{j=1}^{u_{l+1}} (w_{j,i}^{(l)})^2$$

Celem nadrzędnym jest minimalizacja funkcji kosztu  $J(w)$  i aby to osiągnąć, konieczne jest obliczenie pochodnej cząstkowej macierzy  $W$  względem każdej wagi w każdej warstwie sieci:

$$\frac{\partial}{\partial w_{j,i}^{(l)}} J(W)$$

Macierz  $W$  składa się z kilku macierzy, w zależności od liczby warstw w modelu MLP. Dla wielowarstwowego perceptronu z jedną warstwą ukrytą można zdefiniować macierz wag  $W^{(1)}$ , która łączy warstwy wejściową z ukrytą, oraz macierz wag  $W^{(2)}$ , łączącą warstwę ukrytą z wyjściową.

Po wyjaśnieniu kluczowych koncepcji trenowania sztucznej sieci neuronowej, istotne jest teraz omówienie algorytmu wstecznej propagacji, który odgrywa centralną rolę w optymalizacji wag.

Algorytm ten, stworzony prawie trzy dekady temu, wciąż jest jedną z najefektywniejszych technik trenowania sztucznych sieci neuronowych. Pozwala on na skuteczne obliczanie pochodnych złożonej funkcji kosztu, które są następnie wykorzystywane do aktualizacji wag w sieci o wielu warstwach. Głównym wyzwaniem w tym procesie jest ogromna liczba współczynników wag w wielowymiarowej przestrzeni cech. Warto również zauważyć, że powierzchnia błędu funkcji kosztu, w odróżnieniu od niej samej, zawiera liczne nierówności, takie jak lokalne minima, które trzeba wziąć pod uwagę przed dotarciem do globalnego minimum [Rashka, 2018].

Różniczkowanie złożonych, zagnieżdżonych funkcji jest możliwe dzięki tzw. regule łańcuchowej, która pozwala na rozbicie takich funkcji, jak  $f(g(x)) = y$ , na ich podstawowe składowe:

$$\frac{\partial y}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$$

Z punktu widzenia algebry informatycznej opracowano wiele technik tzw. automatycznego różniczkowania, które rozwiązują tego rodzaju wyzwania [6]. Automatyczne różniczkowanie może być realizowane metodą w przód (ang. *forward*) oraz metodą odwrotną (ang. *reverse*). Algorytm wstecznej propagacji jest przykładem różniczkowania zwanym odwrotnym różniczkowaniem automatycznym.

Aby lepiej zrozumieć tę metodę, warto prześledzić jej działanie od strony obliczeniowej, zaczynając od zdefiniowania kosztu jako różnicy między aktywacją ostatniej warstwy a docelową etykietą klas. Propagacja jednokierunkowa (w przód) jest przeprowadzana w celu uzyskania wartości aktywacji warstwy wyjściowej, co można określić następująco [Rashka, 2018]:

- $Z^{(2)} = W^{(1)}[A^{(1)}]^T$  – jest całkowitym pobudzeniem warstwy ukrytej
- $A^{(2)} = \phi(Z^{(2)})$  – jest aktywacją warstwy ukrytej
- $Z^{(3)} = W^{(2)}A^{(2)}$  – jest całkowitym pobudzeniem warstwy wyjściowej
- $A^{(3)} = \phi(Z^{(3)})$  – jest aktywacją warstwy wyjściowej.

Podczas działania algorytmu wstecznej propagacji błąd jest przesyłany od wyjścia do wejścia. Na początku obliczany jest błąd dla warstwy wyjściowej w postaci:

$$\delta^{(3)} = a^{(3)} - y$$

gdzie:

$y$  – jest wektorem rzeczywistych etykiet klas.

Następnie obliczany jest błąd dla warstwy ukrytej:

$$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} * \frac{\partial \phi(z^{(2)})}{\partial z^{(2)}}$$

Pochodna sigmoidalnej funkcji aktywacji wygląda następująco:

$$\frac{\partial \phi(z^{(2)})}{\partial z^{(2)}} = (a^{(2)} * (1 - a^{(2)}))$$

Po obliczeniu wyrażeń  $\delta$ , można zapisać wzór na pochodną funkcji kosztu w następującej formie:

$$\frac{\partial}{\partial w_{j,i}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$$

Teraz należy wyznaczyć pochodną cząstkową dla każdego  $j$ -tego neuronu w warstwie  $l$  oraz obliczyć  $i$ -ty błąd pochodzący z warstwy  $l+1$ . Operację tę można wykonać w formie zwektoryzowanej:

$$\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)} (A^{(l)})^T$$

Po obliczeniu pochodnych cząstkowych można dodać wyrażenie regularizacyjne:

$$\Delta^{(l)} := \Delta^{(l)} + \lambda^{(l)}$$

W ostatnim etapie, po wyznaczeniu gradientów, wagi są modyfikowane poprzez wykonanie ruchu w kierunku przeciwnym do gradientu:

$$W^{(l)} := W^{(l)} - \eta \Delta^{(l)}$$

Implementacja sztucznych sieci neuronowych bywa złożona, dlatego istotne jest upewnienie się, że algorytm wstecznej propagacji został prawidłowo zaimplementowany. Aby to zweryfikować, stosuje się procedurę zwaną sprawdzaniem gradientów (ang. gradient checking), która pozwala na porównanie gradientów obliczonych analitycznie z tymi wyznaczonymi numerycznie. Ta technika nie jest ograniczona wyłącznie do sieci neuronowych z propagacją w przód, lecz może być wykorzystywana w wielu innych architekturach opartych na gradientach [Rashka, 2018].

Przy definiowaniu funkcji kosztu  $J(W)$ , gdzie  $W$  jest macierzą współczynników wag w sieci neuronowej, funkcja kosztu jest obliczana z uwzględnieniem z macierzy  $W(1)$  i  $W(2)$  w modelu wielowarstwowego perceptronu jedną jednostkę ukrytą. Pochodna funkcji kosztu względem wagi  $w_{i,j}^{(l)}$  jest wyrażona jako:

$$\frac{\partial}{\partial w_{i,j}^{(l)}} J(W)$$

Gradienty są modyfikowane poprzez wykonanie kroku w kierunku przeciwnym do ich wartości. Metoda sprawdzania gradientu polega na porównaniu gradientu wyznaczonego analitycznie z wartościami obliczonymi numerycznie:

$$\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) \approx \frac{J(w_{i,j}^{(l)} + \varepsilon) - J(w_{i,j}^{(l)})}{\varepsilon}$$

gdzie:

$\varepsilon$  - przyjmuje wartości rzędu  $1e^{-5}$ .

Lepszą aproksymację gradientu można uzyskać, obliczając symetryczny iloraz różnicowy:

$$\frac{J(w_{i,j}^{(l)} + \varepsilon) - J(w_{i,j}^{(l)} - \varepsilon)}{2\varepsilon}$$

Różnica między gradientem wyznaczonym numerycznie  $J'_n$  a jego analitycznym odpowiednikiem  $J'_a$  jest obliczana jako znormalizowany wektor przy użyciu metody L2. W praktyce wartości macierzy gradientu są rozmieszczane w jednowymiarowych wektorach, co upraszcza obliczenia:

$$\text{błąd} = \|J'_n - J'_a\|_2$$

Dla większej dokładności zaleca się obliczanie znormalizowanej różnicy:

$$\text{błąd względny} = \frac{\|J'_n - J'_a\|_2}{\|J'_n\|_2 + \|J'_a\|_2}$$

Aby wartość względnego błędu między gradientem numerycznym a analitycznym była zminimalizowana, należy przed implementacją określić dopuszczalną wartość progową błędu, która jest uzależniona od złożoności architektury sieci neuronowej. Im więcej warstw ukrytych, tym większa różnica między gradientem numerycznym a analitycznym przy poprawnej implementacji algorytmu wstecznej propagacji [Rashka, 2018].

### 2.1.3. Konwolucyjne sieci neuronowe (CNN)

Opisany wcześniej wielowarstwowy perceptron to jedna z najbardziej popularnych reprezentacji sieci neuronowej ze sprzężeniem w przód. Powstało jednak wiele innych architektur sieci neuronowych. Należy zwrócić uwagę na dwie architektury, szczególnie cenne z punktu widzenia rozpoznawania obrazów. Są to splotowe sieci neuronowe oraz rekurencyjne sieci neuronowe. Spośród nich zdecydowaną przewagą charakteryzuje się architektura CNN [Rashka, 2018]. W kontekście analizy obrazów architektura CNN zyskuje zdecydowaną

przewagę dzięki swojej zdolności do efektywnej analizy przestrzennej rozkładu pikseli w obrazie [LeCun, 1998].

Konwolucyjne sieci neuronowe znane również jako sieci CNN od ang. *convolutional neural networks* są szczególnie popularne w zastosowaniach graficznych ze względu na dużą skuteczność w klasyfikowaniu obrazów. Obecnie są jedną z najbardziej popularnych architektur używanych w głębokim uczeniu. Główną cechą w spłotowych sieciach neuronowych jest generowanie wielu warstw detektorów cech (*features detectors*), które analizują rozmieszczenie przestrzenne pikseli w wejściowym obrazie [Rashka, 2018] [Krizhevsky, 2012].

W klasycznej implementacji wielowarstwowego perceptronu obrazy są rozkładane na wektory cech, które w pełni łączy się z warstwą ukrytą, czyli informacje przestrzenne nie są kodowane w tej architekturze sieci neuronowej. W przypadku architektury CNN wykorzystywane są pola recepcyjne (ang. *receptive fields*) w celu połączenia warstwy wejściowej z mapą cech. Pola recepcyjne można porównać do nakładających się na siebie okienek, które są przesuwane po pikselach obrazu wejściowego w celu utworzenia mapy cech. Stopień przesunięcia okna oraz jego rozmiar to dodatkowe hiperparametry, które należy odgórnie zdefiniować. Tworzenie mapy cech (ang. *features map*) nazywane jest splataniem (konwolucją) [Rashka, 2018] [LeCun, 1998].

W sieciach CNN detektory cech działają jako kopie, czyli pola recepcyjne, które przekazują cechy do jednostek kolejnej warstwy, korzystając z tych samych wag. Zakłada się, że detektor cech, który jest skuteczny w jednym miejscu obrazu, może być równie użyteczny w innym jego obszarze. Kluczową zaletą tego podejścia jest znaczące zmniejszenie liczby parametrów, które trzeba zoptymalizować. Dzięki temu różne części obrazu mogą być reprezentowane w odmienny sposób, a CNN doskonale rozpoznają obiekty o różnych rozmiarach i lokalizacjach na obrazie. Co więcej, sieci te nie wymagają ani skalowania, ani centrowania analizowanych obrazów [Rashka, 2018] [Goodfellow, 2016].

W architekturze CNN po warstwie konwolucyjnej następuje warstwa poolingowa (ang. *pooling layer*). Odpowiada ona za łączenie sąsiadujących detektorów cech, co ma na celu zmniejszenie liczby przekazywanych cech do kolejnej warstwy. Pooling stanowi prostą metodę ekstrakcji cech, w której wybierana jest średnia lub maksymalna wartość z fragmentu obrazu reprezentowanego przez pobliskie cechy, a następnie wynik ten jest przekazywany do następnej warstwy. Aby zbudować głęboką konwolucyjną sieć neuronową, tworzy się wiele warstw



konwolucyjnych i poolingowych, które działają naprzemiennie. Taka struktura jest następnie połączona z wielowarstwowym perceptronem, który odpowiada za finalną klasyfikację [Rashka, 2018] [Goodfellow, 2016].

Konwolucyjne sieci neuronowe są uznawane za najbardziej efektywne modele uczenia maszynowego w zadaniach związanych z analizą obrazów. Ich istotną zaletą jest możliwość trenowania od podstaw, nawet przy użyciu niewielkich zbiorów danych, co pozwala uzyskiwać wartościowe wyniki. Głównym wyzwaniem przy pracy z małymi zbiorami danych jest ryzyko nadmiernego dopasowania modelu. W przypadku obrazów problem ten można złagodzić, stosując technikę augmentacji danych. Ponadto, ekstrakcja cech umożliwia wykorzystanie wcześniej wytrenowanej sieci konwolucyjnej do rozwiązania nowego zadania, co jest szczególnie użyteczne przy ograniczonych danych. Skuteczność tej metody można dodatkowo poprawić poprzez dostrajanie, które dostosowuje wcześniej wyuczone reprezentacje do nowego problemu, zwiększając tym samym efektywność modelu [Chollet, 2019] [Yosinski, 2014].

Konwolucyjne sieci neuronowe (CNN) to powszechnie stosowana architektura sieci neuronowych, która szczególnie dobrze sprawdza się w generowaniu predykcji z obrazów. W kolejnych fragmentach wyjaśniono, dlaczego CNN są idealne do analizy obrazów, przedstawiono ich podstawowe zasady działania oraz omówiono kluczowe aspekty ich implementacji [Krizhevsky, 2012].

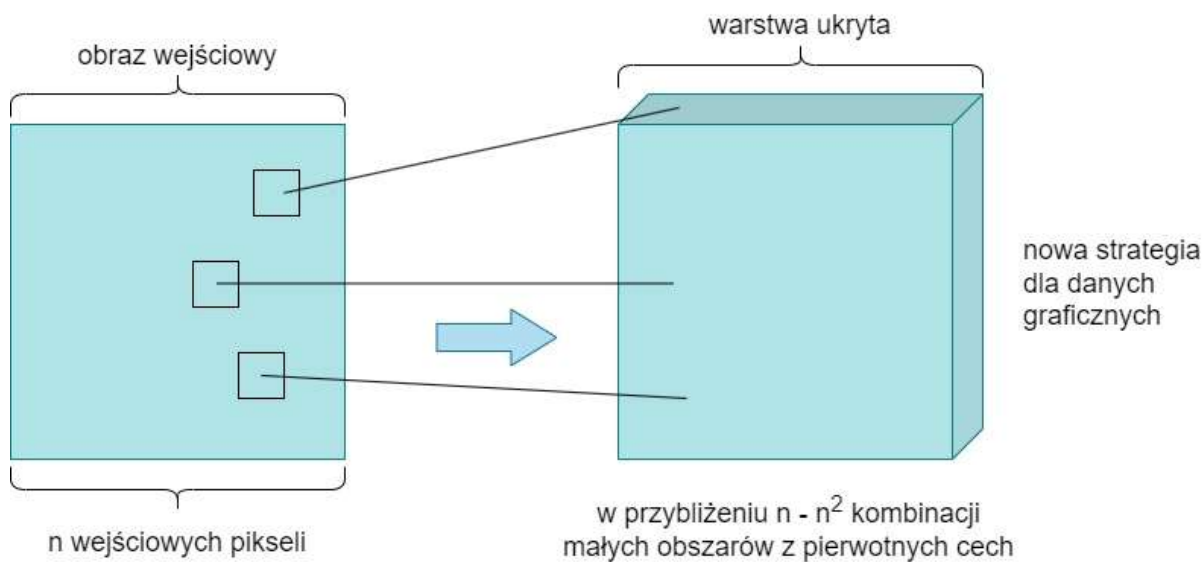
W kontekście sieci neuronowych i uczenia reprezentacji, sieci neuronowe przyjmują na wejściu dane obserwacyjne, gdzie każda obserwacja jest opisana przez zestaw określonej liczby cech. Obrazy są jednym z typowych przykładów takich obserwacji. Są one reprezentowane przez zbiór pikseli rozmieszczonych w przestrzeni dwuwymiarowej, przy czym każdy piksel określa poziom jasności. W związku z tym, każda obserwacja zawiera liczbę wartości równą liczbie pikseli w obrazie. Aby rozpoznać i sklasyfikować obiekty obecne na obrazach, można przeskalować obrazy, a następnie opracować model, który będzie trafnie prognozował wyniki dla dostarczonych danych. Prosta sieć neuronowa z jedną warstwą ukrytą okazuje się skuteczniejsza niż model bez takiej warstwy, ponieważ pozwala na naukę złożonych, nieliniowych zależności między danymi wejściowymi a wyjściowymi. Dodatkowo, w procesie uczenia maszynowego często konieczne jest tworzenie kombinacji liniowych pierwotnych cech w celu poprawy skuteczności prognozowania zmiennej docelowej [LeCun, 1998].

Rozważając typowy zbiór obrazów MNIST, gdzie wartości pikseli dla każdej cyfry są oznaczone symbolami od  $x_1$  do  $x_{784}$ . Możliwe jest, że pewne kombinacje wartości pikseli mogą

silnie sugerować, że obraz przedstawia określoną cyfrę, np. 9. Sieci neuronowe są w stanie automatycznie wykrywać takie kombinacje cech w procesie uczenia, tworząc tym samym uczenie reprezentacji, co czyni je skutecznymi w wielu dziedzinach [Goodfellow, 2016].

Warto zadać pytanie, czy istnieje uzasadnienie dla modyfikacji tego procesu w przypadku danych graficznych. Argumentem na tak jest to, że w obrazach istotne kombinacje cech (pikseli) zazwyczaj składają się z pikseli położonych blisko siebie. Na obrazie znacznie mniej prawdopodobne jest, aby ważna cecha wynikała z losowej kombinacji dziewięciu oddalonych pikseli, niż z grupy sąsiadujących pikseli, np. o wymiarach 3x3. Kluczowym celem jest więc wykorzystanie tej specyfiki danych graficznych, co oznacza, że kolejność cech ma znaczenie, ponieważ wskazuje, które piksele są ze sobą przestrzennie powiązane [Weidman, 2019].

Wspomniany wcześniej fakt znajduje zastosowanie na szerokim poziomie, gdzie tworzenie kombinacji cech odbywa się w podobny sposób, jak dotychczas, jednak z większą liczbą tych kombinacji. Kluczowym elementem jest jednak ograniczenie tworzenia kombinacji wyłącznie do pikseli znajdujących się w małych, prostokątnych obszarach obrazu wejściowego. Dzięki temu możliwe jest efektywne wykorzystanie sąsiadujących pikseli, co ilustruje rysunek 2.7.



**Rysunek 2.7.** Dla danych graficznych można zdefiniować każdą uczoną cechę jako funkcję od małego obszaru z danych, (rysunek wykonany na podstawie: [Weidman, 2019]).

Zasadniczo, w przypadku obrazów, interesujące kombinacje cech najczęściej obejmują piksele umiejscowione blisko siebie. Aby maksymalnie wykorzystać tę właściwość, w

konwolucyjnych sieciach neuronowych (CNN) tworzy się kombinacje cech wyłącznie z niewielkich, prostokątnych fragmentów obrazu wejściowego. Proces ten, nazywany konwolucją, umożliwia efektywne rozpoznawanie wzorców w obrazach [LeCun, 1998][1].

Aby dokładnie wyjaśnić działanie operacji konwolucji, konieczne jest sprecyzowanie, określenia „cecha, która jest kombinacją pikseli z określonego fragmentu obrazu”. Należy założyć, że obraz wejściowy jest reprezentowany jako macierz  $I$  o wymiarach 5x5, jak przedstawiono poniżej [Weidman, 2019]:

$$I = \begin{bmatrix} i_{11} & i_{12} & i_{13} & i_{14} & i_{15} \\ i_{21} & i_{22} & i_{23} & i_{24} & i_{25} \\ i_{31} & i_{32} & i_{33} & i_{34} & i_{35} \\ i_{41} & i_{42} & i_{43} & i_{44} & i_{45} \\ i_{51} & i_{52} & i_{53} & i_{54} & i_{55} \end{bmatrix}$$

Celem jest wyznaczenie nowej cechy, która będzie funkcją obszaru składającego się z 3x3 pikseli znajdujących się w centralnej części obrazu. W tradycyjnych sieciach neuronowych nowe cechy są określane jako liniowe kombinacje wcześniejszych cech, natomiast tutaj nowa cecha zostanie zdefiniowana jako funkcja tego wycinka 3x3. Aby to zrealizować, konieczne jest zdefiniowanie zbioru wag  $W$  o wymiarach 3x3 [Weidman, 2019]:

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

W kolejnym kroku należy obliczyć iloczyn skalarny między macierzą wag  $W$  a odpowiadającym jej fragmentem z macierzy  $I$ , aby uzyskać wartość nowej cechy w danych wyjściowych. Ponieważ fragment obrazu wejściowego jest wyśrodkowany w punkcie (3,3), wartość wyjściową można oznaczyć jako  $o_{33}$  (od angielskiego "output") [Weidman, 2019]:

$$o_{33} = w_{11} \times i_{22} + w_{12} \times i_{23} + w_{13} \times i_{24} + w_{21} \times i_{32} + w_{22} \times i_{33} + w_{23} \times i_{34} + w_{31} \times i_{42} + w_{32} \times i_{43} + w_{33} \times i_{44}$$

Wartość tą można traktować jak dowolną inną cechę obliczoną z sieci neuronowych, czyli można do niej dodać wyraz wolny, a następnie przekazać do funkcji aktywacji. Wówczas będzie reprezentować neuron lub wyuczoną cechę, która zostanie przesłana do kolejnych warstw sieci. W ten sposób definiowane są cechy jako funkcje od małych fragmentów obrazu wejściowego [Weidman, 2019].

Należy również zwrócić uwagę na sposób interpretacji uzyskanych cech, ponieważ posiadają one szczególne znaczenie. Cechy te wskazują, czy dany wzorec wizualny, określony

przez wagi, jest obecny w konkretnej części obrazu. W dziedzinie rozpoznawania obrazów macierze o wymiarach np. 3x3 lub 5x5 mogą działać jako detektory wzorców, kiedy oblicza się ich iloczyn skalarny z wartościami pikseli w określonych regionach obrazu. Na przykład, iloczyn skalarny macierzy liczb o wymiarach 3x3 :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

w połączeniu z odpowiednim fragmentem obrazu wejściowego umożliwia wykrycie obecności krawędzi w danym miejscu. Istnieją także inne powszechnie stosowane macierze, które służą do identyfikowania naroży oraz linii poziomych i pionowych w określonych obszarach obrazu [Weidman, 2019].

Można założyć, że ten sam zestaw wag  $W$  jest używany do wykrywania, czy wzorec wizualny, który one definiują, występuje w różnych miejscach obrazu wejściowego. Wygląda to jak przesuwanie zbioru wag  $W$  nad obrazem wejściowym, podczas gdy w każdej lokalizacji obliczany jest iloczyn skalarny między wagami a pikselami. W wyniku tej operacji powstaje nowy obraz  $O$ , o wielkości zbliżonej do obrazu oryginalnego (jego rozmiar może się nieznacznie różnić w zależności od sposobu traktowania brzegów obrazu). Obraz  $O$  to tzw. mapa cech, która przedstawia miejsca na obrazie wejściowym, gdzie obecny jest wzorec zdefiniowany przez  $W$ . Proces ten, kluczowy dla konwolucyjnych sieci neuronowych, nazywany jest konwolucją, a jego wynik to wspomniana mapa cech. Aby nadać tej operacji pełną funkcjonalność, można dodać dodatkowy wymiar [Weidman, 2019].

W kontekście wielokanałowej operacji konwolucji, sieci CNN różnią się od tradycyjnych sieci neuronowych tym, że generują znacznie większą liczbę cech, a każda z tych cech jest funkcją niewielkiego fragmentu obrazu wejściowego. Jeśli obraz składa się z  $n$  pikseli, operacja konwolucji stworzy  $n$  cech wyjściowych, po jednej dla każdej lokalizacji na obrazie wejściowym. Jednak konwolucyjna warstwa sieci neuronowej idzie krok dalej, tworząc  $f$  zestawów po  $n$  cech. Każdy zestaw jest związany z określonym (początkowo losowym) zbiorem wag, które definiują wizualny wzorec, a jego obecność w różnych miejscach obrazu wejściowego jest rejestrowana w mapie cech. Wspomniane  $f$  map cech są efektem  $f$  operacji konwolucji [Weidman, 2019].

Każdy zestaw cech wykryty przez określony zbiór wag nazywany jest mapą cech, jednak w kontekście warstwy konwolucyjnej, liczba takich map cech odnosi się do operacji wielokanałowej konwolucji. Dodatkowo, zestawy wag  $W_i$ , które tworzą mapy cech, określa się

mianem filtrów konwolucyjnych lub jąder konwolucji (ang. *convolution kernels*) [Weidman, 2019].

W kontekście warstw konwolucyjnych, przekształcają one dane wejściowe w wielowymiarowe tablice, które odpowiadają mapom cech. Każda warstwa konwolucyjna generuje wiele takich map, a te są następnie przetwarzane przez kolejne warstwy sieci, tworząc coraz bardziej złożone i abstrakcyjne reprezentacje danych. Dzięki temu sieć neuronowa jest w stanie rozpoznawać skomplikowane wzorce wizualne [Goodfellow, 2016][3].

Zrozumienie operacji konwolucji wielokanałowej pozwala na lepsze zrozumienie jej integracji z warstwami sieci neuronowej. W standardowych warstwach sieci neuronowej dane wyjściowe są zazwyczaj dwuwymiarowymi tablicami, jednak w warstwach konwolucyjnych dla pojedynczego obrazu wynikają trójwymiarowe tablice *ndarray*, gdzie wymiary odpowiadają liczbie map cech, wysokości i szerokości obrazu. Kluczowe pytanie dotyczy sposobu przekazywania takiej trójwymiarowej tablicy do kolejnych warstw w celu zbudowania głębokiej sieci konwolucyjnej. Operacja wielokanałowej konwolucji na danych z wieloma kanałami (co jest konieczne dla współdziałania dwóch warstw konwolucyjnych) stanowi fundament głębokich konwolucyjnych sieci neuronowych [Weidman, 2019].

Warto zastanowić się, co dzieje się w sieciach neuronowych z warstwami gęstymi. W pierwszej warstwie ukrytej powstaje  $h_1$  cech, które są kombinacjami wszystkich pierwotnych cech z warstwy wejściowej. W kolejnej warstwie te cechy są dalej łączone, tworząc nowe kombinacje oparte na cechach z poprzedniej warstwy. W rezultacie, w drugiej warstwie może pojawić się  $h_2$  cech, które są bardziej złożonymi kombinacjami tych z pierwszej warstwy. Aby utworzyć taką kolejną warstwę  $h_2$  cech, konieczne jest wykorzystanie  $h_1 \times h_2$  wag, aby każda z nowych cech była funkcją wszystkich cech z warstwy poprzedniej [Weidman, 2019].

Podobny proces ma miejsce w pierwszej warstwie konwolucyjnej sieci neuronowej. Na początku obraz wejściowy jest przekształcany za pomocą  $m_1$  filtrów konwolucyjnych, co prowadzi do powstania  $m_1$  map cech. Wynik tej warstwy można interpretować jako informację o tym, czy każdy z  $m_1$  wizualnych wzorców, zdefiniowanych przez wagi filtrów, jest obecny w konkretnych częściach obrazu wejściowego. Tak jak w gęstych warstwach sieci neuronowych liczba neuronów może się różnić, tak samo w pierwszej warstwie konwolucyjnej można zastosować  $m_2$  filtry. W celu nauki bardziej złożonych wzorców, filtry te działają jak narzędzia, które wykrywają, czy bardziej zaawansowane cechy wizualne – wynikające z połączeń  $m_1$  wzorców z poprzedniej warstwy – są obecne w danym obszarze obrazu. W



rezultacie, jeśli wyjściem warstwy konwolucyjnej jest trójwymiarowa tablica o wymiarach  $m_2$  kanałów  $\times$  wysokość  $\times$  szerokość, to każda lokalizacja na jednej z tych  $m_2$  map cech odpowiada kombinacji  $m_1$  wzorców wizualnych, wyuczonych już w poprzedniej warstwie [Weidman, 2019].

W kontekście wpływu na implementację, zrozumienie połączenia dwóch wielokanałowych warstw konwolucyjnych pozwala na właściwe zaplanowanie operacji konwolucji. Tak jak do połączenia dwóch gęstych warstw neuronowych potrzeba  $h_1 \times h_2$  wag, tak  $m_1 \times m_2$  filtrów konwolucyjnych jest wymaganych, aby połączyć warstwę z  $m_1$  kanałami z warstwą mającą  $m_2$  kanałów. Ustalenie tych szczegółów umożliwia dokładne określenie wymiarów tablic *ndarray* dla danych wejściowych, wyjściowych oraz filtrów w operacji konwolucji [Weidman, 2019].

Dane wejściowe mają kształt zależny od wielkości porcji, liczby kanałów wejściowych, a także wysokości i szerokości obrazu. Z kolei dane wyjściowe są zależne od wielkości porcji, liczby kanałów wyjściowych oraz rozmiarów obrazu, czyli jego wysokości i szerokości. Filtry konwolucyjne są opisywane poprzez liczbę kanałów wejściowych, kanałów wyjściowych oraz wysokość i szerokość filtra. Warto podkreślić, że choć kolejność tych parametrów może się różnić w zależności od użytej biblioteki, to zawsze występują cztery kluczowe wymiary, co jest niezwykle istotne przy projektowaniu i wdrażaniu operacji konwolucyjnych [Weidman, 2019].

Różnice między warstwami konwolucyjnymi, a gęstymi są znaczące, zwłaszcza ze względu na sposób interpretowania neuronów w tych warstwach. W warstwie gęstej neuron odpowiada za wykrywanie, czy wyuczona kombinacja cech z poprzedniej warstwy jest obecna w bieżącej obserwacji. Natomiast w warstwie konwolucyjnej neuron identyfikuje, czy wyuczony wzorec wizualny występuje w określonym miejscu obrazu wejściowego. Przed wprowadzeniem takiej warstwy do sieci neuronowej należy rozważyć, jak użyć wielowymiarowych tablic *ndarray*, które stanowią dane wyjściowe, w procesie generowania predykcji [Weidman, 2019].

Warstwy konwolucyjne uczą się rozpoznawać cechy wskazujące na obecność określonych wzorców wizualnych w obrazach, a zapis tych cech odbywa się w mapach cech. Ważne jest jednak, jak te mapy cech mogą być użyte do generowania predykcji. W gęstych sieciach neuronowych, podczas prognozowania przynależności obrazu do jednej z  $X$  kategorii, ostatnia warstwa zazwyczaj ma  $X$  wyjść, co pozwala na przekazanie tych wartości do funkcji straty, takiej jak softmax z entropią krzyżową, i zinterpretowanie ich jako prawdopodobieństwa.

W sieciach konwolucyjnych sytuacja wygląda nieco inaczej, ponieważ dla każdej obserwacji otrzymywana jest trójwymiarowa tablica o wymiarach  $m$  kanałów  $\times$  wysokość obrazu  $\times$  szerokość obrazu [Weidman, 2019].

Aby wyjaśnić to zagadnienie, należy zauważyć, że każdy neuron w warstwie konwolucyjnej informuje, czy dana kombinacja cech wizualnych (które w głębokich sieciach konwolucyjnych mogą być bardzo złożone) występuje w określonej lokalizacji obrazu. Jest to podobne do procesu, w którym gęsta sieć neuronowa uczy się tych cech z obrazu. W architekturze gęstej pierwsza warstwa zwykle reprezentuje cechy poszczególnych pikseli, natomiast kolejne warstwy tworzą coraz bardziej złożone kombinacje. W gęstych sieciach każda wyuczona cecha może być traktowana jako pojedynczy neuron, który służy do generowania predykcji o przynależności obrazu do określonej kategorii [Weidman, 2019].

Warstwy agregujące stanowią istotny element w strukturze sieci konwolucyjnych, mając na celu redukcję wymiarów mapy cech. Proces ten pozwala na zmniejszenie liczby parametrów oraz obciążenia obliczeniowego, co prowadzi do zwiększenia efektywności modelu i ogranicza ryzyko nadmiernego dopasowania (ang. *overfitting*). W warstwach tych najczęściej wykorzystywane są dwie techniki: max pooling oraz average pooling [Weidman, 2019].

W przypadku max pooling, z każdej grupy sąsiadujących pikseli wybierana jest największa wartość, co umożliwia zachowanie kluczowych cech obrazu, takich jak krawędzie czy wyraźne wzory. Z kolei average pooling polega na obliczaniu średniej wartości z sąsiadujących pikseli, co jest użyteczne w sytuacjach, gdy wymagane jest bardziej wygładzone wyjście. Obie metody dążą do redukcji rozmiaru map cech, upraszczając model przy jednoczesnym zachowaniu istotnych informacji przestrzennych [Weidman, 2019].

Implementowanie wielokanałowej operacji konwolucji jest istotnym elementem budowy sieci CNN. Każdy kanał obrazu, na przykład kanały RGB w przypadku obrazów kolorowych, jest przetwarzany oddzielnie, a następnie wyniki są łączone podczas konwolucji. Dla każdego kanału obrazu stosowane są różne filtry konwolucyjne, co umożliwia wykrywanie różnych cech charakterystycznych w poszczególnych kanałach [Weidman, 2019].

Każda warstwa konwolucyjna w CNN zawiera zestaw filtrów, które przetwarzają dane równolegle. W wyniku tego procesu powstaje zestaw map cech, z których każda odzwierciedla reakcję danego filtra na obraz wejściowy. Mapy cech są następnie przekazywane do kolejnych

warstw sieci, co pozwala na uzyskanie coraz bardziej abstrakcyjnych reprezentacji danych [Weidman, 2019].

Krok w przód w kontekście sieci konwolucyjnych (CNN) odnosi się do procesu przepływu danych wejściowych przez kolejne warstwy sieci w celu uzyskania wyniku końcowego. W tym procesie, każda warstwa sieci – począwszy od warstwy konwolucyjnej, przez warstwy agregujące, aż po w pełni połączoną warstwę wyjściową – przekształca dane, tworząc coraz bardziej złożone i abstrakcyjne reprezentacje. W sieciach CNN warstwy konwolucyjne są odpowiedzialne za identyfikację różnych cech obrazu, co finalnie umożliwia klasyfikację w ostatniej warstwie sieci [Weidman, 2019].

Operacja konwolucji opierająca się na kroku wstecz odnosi się do procesu aktualizacji wag filtrów konwolucyjnych oraz innych parametrów sieci podczas treningu. Ten proces realizowany jest za pomocą algorytmu wstecznej propagacji błędów (ang. *backpropagation*), który oblicza gradienty błędu względem każdej wagi w sieci [Weidman, 2019].

W sieciach konwolucyjnych (CNN) obliczanie gradientów dla filtrów jest bardziej złożone niż w klasycznych sieciach neuronowych, ze względu na wielokanałowy charakter danych i bardziej skomplikowaną strukturę warstw. Jednak dzięki zastosowaniu wydajnych algorytmów optymalizacji, takich jak gradient prosty (ang. *gradient descent*), sieci CNN mogą być skutecznie trenowane, nawet na bardzo dużych zbiorach danych [Weidman, 2019].

Porcje danych, konwolucje dwuwymiarowe i operacje wielokanałowe odgrywają kluczową rolę podczas treningu sieci CNN. Dane wejściowe są przetwarzane w porcjach (ang. *batches*), co umożliwia efektywne zarządzanie zasobami obliczeniowymi oraz stabilizuje proces uczenia. Każda porcja przechodzi przez operacje konwolucji dwuwymiarowej, gdzie filtry przetwarzają obraz w dwóch wymiarach – wysokości i szerokości [Weidman, 2019].

Operacje wielokanałowe, jak wspomniano wcześniej, dotyczą przetwarzania obrazów zawierających wiele kanałów, takich jak RGB. Dla każdego kanału stosowane są różne filtry, co prowadzi do powstania map cech uwzględniających specyficzne cechy każdego kanału, umożliwiając tym samym bardziej wszechstronną analizę obrazu [Weidman, 2019].

Konwolucje dwuwymiarowe stanowią podstawowy element budowy sieci CNN. W tej operacji filtr, zwany także jądrem konwolucji, przesuwa się po obrazie, przetwarzając grupy sąsiadujących pikseli. W efekcie powstaje mapa cech, która ukazuje obecność określonych wzorców wizualnych w obrazie. To umożliwia sieciom CNN wykrywanie kluczowych

elementów obrazu, takich jak krawędzie, tekstury i inne istotne cechy, co jest niezbędne do dalszej klasyfikacji [Weidman, 2019].

Ostatnim etapem operacji konwolucyjnej jest dodawanie kanałów (ang. channel concatenation), czyli łączenie wyników pochodzących z różnych filtrów konwolucyjnych. Każda warstwa konwolucyjna w sieci CNN tworzy mapy cech, które są następnie scalane w trójwymiarową tablicę danych o strukturze: liczba kanałów  $\times$  wysokość obrazu  $\times$  szerokość obrazu. Ta trójwymiarowa struktura przechodzi przez kolejne warstwy sieci, aż do momentu, gdy osiągnięta zostanie finalna klasyfikacja [Weidman, 2019].

Podczas treningu sieci CNN nowo utworzone operacje konwolucyjne i agregujące są wykorzystywane na każdym etapie przetwarzania danych wejściowych. Każda warstwa sieci stopniowo uczy się coraz bardziej abstrakcyjnych reprezentacji obrazu, co umożliwia sieciom CNN skuteczne rozpoznawanie i klasyfikację złożonych wzorców wizualnych. Proces ten przebiega iteracyjnie i wymaga wielu cykli treningowych (epok), podczas których sieć stopniowo doskonali swoje parametry, aby osiągnąć jak najlepsze wyniki. Tym samym kluczowe operacje leżące u podstaw działania CNN przyczyniają się do ich wysokiej efektywności w analizie obrazów [Weidman, 2019].

Warto omówić po krótku również kluczowe elementy architektury sieci konwolucyjnych (CNN), które odgrywają istotną rolę w procesie przetwarzania danych, generowania predykcji oraz klasyfikacji obrazów. Elementy takie jak przekształcanie danych wyjściowych, warstwy konwolucyjne oraz metody oceny skuteczności sieci, pozwalają na głębsze zrozumienie funkcjonowania CNN oraz ich efektywności w analizie obrazów [Weidman, 2019].

Pierwszym ważnym elementem jest operacja Flatten, która odgrywa kluczową rolę w architekturze sieci CNN. Polega ona na przekształceniu trójwymiarowej tablicy danych, powstałej w wyniku przetwarzania przez warstwy konwolucyjne, w formę jednowymiarową. Dzięki temu dane mogą zostać przekazane do w pełni połączonych warstw sieci, odpowiedzialnych za ostateczną klasyfikację. Operacja ta jest zwykle wykonywana na końcu przetwarzania przez warstwy konwolucyjne, tuż przed przekazaniem wyników do warstwy wyjściowej [Weidman, 2019].

Kolejnym istotnym elementem architektury CNN jest warstwa Conv2D, która stanowi fundament działania tych sieci. Warstwa ta składa się z filtrów konwolucyjnych, które przetwarzają dane wejściowe, generując mapy cech. Mapy te reprezentują kluczowe wzorce

wykryte w obrazach. Każda warstwa Conv2D zawiera również funkcje aktywacji, które nadają sieci nieliniowy charakter. Warstwy te są łączone w sekwencje, co pozwala na budowanie głębokich sieci zdolnych do rozpoznawania coraz bardziej złożonych wzorców. W połączeniu z warstwami agregującymi, warstwa Conv2D umożliwia efektywne przetwarzanie obrazów [Weidman, 2019].

Warto wspomnieć również o eksperymentach, które odgrywają istotną rolę w ocenie efektywności architektur CNN. Przeprowadza się je na różnych zbiorach danych, takich jak MNIST, CIFAR-10 czy ImageNet, w celu porównania skuteczności różnych modeli w rozpoznawaniu obrazów. Eksperymenty te umożliwiają ocenę wydajności poszczególnych warstw, parametrów sieci oraz metod regularizacji, co jest kluczowe dla optymalizacji modeli CNN i ich zastosowań w praktyce [Weidman, 2019].

#### 2.1.4. Architektury ResNet50, ResNet101, ResNeXt101

Wykorzystanie konwolucyjnych sieci neuronowych w uczeniu głębokim znacząco wpłynęło na obszar analizy obrazów. Zdolność sieci tego typu do przeprowadzania automatycznej ekstrakcji cech z obrazów umożliwiła ich zastosowanie zarówno w klasyfikacji, detekcji obiektów, jak i w zadaniach związanych z segmentacją obrazów. W przypadkach bardziej złożonych pojawiają się jednak trudności z trenowaniem szczególnie głębokich sieci. Występuje tu problem znikającego gradientu, który utrudnia trenowanie głębokich modeli w sposób efektywny.

W celu minimalizacji kłopotów związanych z problemem znikającego gradientu stworzono architektury sztucznych sieci neuronowych zawierających bloki rezydualne, które pozwoliły na trenowanie dużo głębszych modeli. Od nazwy bloków rezydualnych sieci tego typu nazywane są sieciami rezydualnymi [He i in., 2016]. Stały się one podstawą nowoczesnych rozwiązań w dziedzinie uczenia głębokiego, zapewniając większe dokładności w zadaniach związanych z analizą obrazów unikając konieczności wzrostu złożoności obliczeniowej [He, 2016]. W podrozdziale tym zostaną po krótko przybliżone sieci rezydualne, które zostały bezpośrednio użyte w badaniach przeprowadzonych w ramach realizowanego doktoratu. Będą nimi kolejno architektury: ResNet50, ResNet101 oraz ResNeXt101.

Architektura ResNet50 została wprowadzona przez Kaiming He i współautorów w artykule naukowym pt. „Deep Residual Learning for Image Recognition” w 2016 roku. Jest często używanym modelem sieci w zadaniach związanych z detekcją obiektów i semantyczną



segmentacją [He, 2016]. Architektura ta składa się z 50 warstw, na co wskazuje nazwa sieci. Wprowadzone bloki rezydualne umożliwiają przepływ sygnału między poszczególnymi warstwami sieci za pomocą tzw. *shortcut connections*, czyli połączeń skróconych. Z ich użyciem możliwe jest bezproblemowe trenowanie nawet bardzo głębokich sieci unikając przy okazji problemów z propagacją gradientu, co zwykle jest przeszkodą w konwergencji w przypadku klasycznych modeli głębokich [He, 2016]. W strukturze ResNet50 występują cztery główne etapy ekstrakcji cech, a w każdym z nich obecny jest zestaw bloków rezydualnych. W każdym bloku znajduje się konwolucja 1x1, konwolucja 3x3 oraz ponownie konwolucja 1x1, co optymalizuje koszt obliczeniowy i efektywność ekstrakcji cech. Pierwsza konwolucja 1x1 obecna na początku bloku ma za zadanie zmniejszenie liczby kanałów, dzięki czemu możliwa jest redukcja ilości operacji koniecznych do wykonania w dalszych warstwach i dzięki temu zmniejsza się koszt obliczeniowy. Kolejna konwolucja, tym razem 3x3 odpowiedzialna jest za przeprowadzenie głównej operacji ekstrakcji cech z wejściowych danych. Realizuje ona analizę zależności przestrzennych w obrazie i dzięki pierwotnemu zmniejszeniu liczby kanałów przez konwolucję 1x1 jest bardziej efektywna pod względem obliczeniowym. Ponowne zastosowanie konwolucji 1x1 zapewnia zwiększenie liczby kanałów do początkowej wartości, dzięki czemu do kolejnych warstw przekazywana jest znów pełna informacja. Takie zabiegi zapewniają zmniejszenie ilości obliczeń bez utraty jakości wyników. Architektura ResNet50 jest wydajnym modelem często stosowanym w analizie obrazów z uwagi na zachowaną efektywność obliczeniową i stosunkowo niski koszt obliczeniowy [He, 2016], [Simonyan, 2014].

Analogią to architektury ResNet50 jest architektura ResNet101 składająca się ze 101 warstw. Również została ona opublikowana w tej samej publikacji naukowej, co ResNet50 oraz składa się z bloków rezydualnych umożliwiających trenowanie bez znikającego gradientu. Większa liczba warstw pozwala analizować bardziej złożone zależności w obrazach, dzięki czemu zwiększa się dokładność klasyfikacji, detekcji obiektów czy też segmentacji [He, 2016]. Większa liczba bloków rezydualnych pozwala na lepsze odwzorowanie cech, co jest korzystne w przypadku zadań wymagających szczególnej precyzji jak obrazowanie medyczne czy badanie cech roślin [Litjens, 2017]. Zwiększenie precyzji odbywa się kosztem większych zasobów obliczeniowych.

Zupełnie nową, bardziej zaawansowaną wersją ResNet jest architektura typu ResNeXt101, również wykorzystana podczas prowadzonych w doktoracie badań. Architektura ta została przedstawiona po raz pierwszy w artykule pt. „Aggregated residual transformations for deep neural networks” opublikowanym przez Xie, S., Girshick, R., Dollár, P., Tu, Z., & He,

K. w 2017 roku. W tym przypadku również wykorzystano bloki rezydualne, ale nowością jest wprowadzenie grupowych konwolucji. Dzięki temu ResNeXt jest zdolny do wykonywania równoległych operacji na wielu kanałach wejściowych zebranych w podgrupy. Co ważne, ResNeXt stosuje pojęcie kardynalności, które oznacza liczbę równoległych grup filtrów obecnych w warstwach konwolucyjnych. Właśnie zwiększenie kardynalności umożliwia wydobycie zróżnicowanych cech z obrazu bez nadmiernego zwiększenia liczby parametrów. Dzięki temu przy podobnym koszcie obliczeniowym, jaki występuje w architekturach ResNet, możliwe jest osiągnięcie lepszej wydajności [Xie, 2017]. ResNeXt101 jest przykładem elastycznej i wydajnej architektury i sprawdza się w wymagających zadaniach takich jak analizowanie obrazów medycznych [Litjens, 2017], czy też w przypadku rozwiązań związanych z autonomicznymi pojazdami [Geiger, 2012].

Dokonując teraz swego rodzaju podsumowania, architektura ResNet50 zapewniła skuteczne trenowanie głębokich sztucznych sieci neuronowych unikając problemu z propagacją gradientu dzięki zastosowaniu bloków rezydualnych, architektura ResNet101 pozwoliła na uzyskanie większej precyzji wyników dzięki wprowadzeniu większej liczby warstw, z kolei architektura ResNeXt101 wprowadziła grupowe konwolucje, które zapewniły wzrost elastyczności i skalowalności wynikowego modelu. Architektury te zapewniają dobrą wydajność przy efektywności obliczeniowej zachowanej na należytym poziomie. Stąd też stanowią ważny element w rozwoju sztucznych sieci neuronowych, ze szczególnym uwzględnieniem analizy obrazów, co jest istotne z punktu widzenia niniejszej rozprawy.

#### 2.1.5. Architektura sieci typu Transformer w analizie obrazów

Architektury sztucznych sieci neuronowych typu transformer wywodzą się z zadań dotyczących przetwarzania języka naturalnego, w skrócie NLP. Nowością w ich stosowaniu są zadania związane z analizą obrazów. W odróżnieniu od sieci konwolucyjnych, sieci typu transformer korzystają z tzw. mechanizmu uwagi umożliwiającego wybiórcze skupienie uwagi tylko na ważnych częściach wejściowych danych, uniezależniając się od położenia przestrzennego tychże części danych [Vaswani, 2017].

Bazowa struktura transformera to wielowarstwowe moduły uwagi (*self-attention layers*) pozwalające na jednoczesne przetwarzanie danych oraz dynamiczne określanie wag dla konkretnych cech obrazu. W sieciach konwolucyjnych informacja przestrzenna zachowywana jest poprzez operacje konwolucji oraz pooling, zaś w przypadku transformerów jest to

realizowane przez wykorzystanie pozycyjnych kodów (*positional encodings*), których zadaniem jest dodawanie informacji na temat kolejności cech wejściowych [Vaswani, 2017].

Transformery, obecnie wykorzystywane również w analizie obrazów, pierwotnie były używane tylko i wyłącznie w przetwarzaniu danych sekwencyjnych, np. tekstów. Ich elastyczność i umiejętność modelowania zależności długoterminowych wpłynęły na wykorzystanie transformerów właśnie w analizie obrazów. Jedną z pionierskich architektur typu transformer stosowanych do analizy obrazów był Vision Transformer (ViT), który stanowił propozycję alternatywy dla konwolucyjnych sieci neuronowych [Dosovitskiy i in., 2020]. Więcej informacji o tej architekturze zostało umieszczonych w kolejnym podrozdziale, gdyż jest to architektura wykorzystywana również w praktyce w ramach niniejszej rozprawy.

Niewątpliwą zaletą architektur typu transformer jest ich umiejętność zrównoleglonego przetwarzania danych oraz bardziej efektywne modelowanie długoterminowych zależności pomiędzy wybranymi cechami obrazu. Są one elastyczne biorąc pod uwagę skalowalność, a to umożliwia ich stosowanie m.in. w dziedzinie wizji komputerowej [Khan, 2021]. Istnieją jednak pewne wyzwania związane ze stosowaniem transformerów do analizy obrazów. Jest to duża złożoność obliczeniowa pociągająca za sobą zaangażowanie większej mocy obliczeniowej w stosunku do sieci konwolucyjnych. Mogą być one również bardziej podatne na nadmierne dopasowanie w przypadkach stosowania małych zbiorów uczących, a to pociąga za sobą konieczność stosowania technik regularyzacji [Dosovitskiy i in., 2020].

#### 2.1.6. Zaawansowane architektury: HTC, Mask2Former, ConvNeXt, EfficientNet, Vision Transformer, Swin Transformer

W ramach badań przeprowadzonych w trakcie przygotowywania niniejszej rozprawy oprócz omówionych wcześniej sieci typu ResNet, wykorzystane zostały również bardziej zaawansowane architektury, takie jak Hybrid Task Cascade (HTC), Mask2Former, ConvNeXt small, EfficientNet B3, Vision Transformer oraz Swin Transformer. Są to przykłady kluczowych architektur, które odegrały istotną rolę w nowoczesnej analizie obrazów i zostaną kolejno omówione w niniejszym podrozdziale.

Architektura Hybrid Task Cascade (w skrócie HTC) znalazła zastosowanie szczególnie w zadaniach związanych z jednoczesnym wykrywaniem obiektów i segmentacją instancji [Chen i in., 2018]. Osiąga ona wysoką dokładność na zbiorach danych COCO. Przy wyborze

wersji HTC zawsze należy wskazać szkielet, czyli tzw. *backbone*. W przypadku niniejszej rozprawy wykorzystane zostały architektury HTC\_x101, HTC\_r101 oraz HTC\_r50.

HTC\_x101 jako szkielet wykorzystuje ResNeXt101, HTC\_r101 korzysta z ResNet101, zaś w przypadku HTC\_r50 zastosowaną siecią jest ResNet50, a każda z tych sieci została omówiona we wcześniejszej części niniejszej rozprawy. Wszystkie trzy konfiguracje jako tzw. „szyję” (*neck*) wykorzystują mechanizm Feature Pyramid Network (FPN) pozwalający na efektywne wykrywanie obiektów o różnej skali [Chen i in., 2018]. Jeśli chodzi zaś o tzw. „głowice” (*heads*), we wszystkich modelach są one analogiczne i należą do nich: RPN Head (Region Proposal Network) generująca propozycje regionów zainteresowania (ROI) przetwarzanych przez kolejne moduły modelu, ROI Head wykonująca wieloetapową regresję ramki ograniczającej i klasyfikację obiektów w celu zwiększenia precyzji wykrywania, Mask Head będącą modulem segmentacji instancji oraz Semantic Head dodającą informacje kontekstowe poprawiając tym samym ogólną wydajność modelu w zadaniach semantycznej segmentacji. Optymalizatorem we wszystkich trzech architekturach jest Stochastic Gradient Descent (SGD) z odpowiednio ustawionymi hiperparametrami dotyczącymi szybkości uczenia, momentum i spadku ciężaru. We wszystkich tych architekturach zastosowany jest również tzw. RoIAlign (czyli Region of Interest Align), który ma za zadanie zwiększenie dokładności wykrywania i segmentacji [Chen i in., 2018]. Samo zastosowanie różnorodnych konfiguracji HTC daje swego rodzaju elastyczność w doborze właściwej architektury do konkretnych wymagań obliczeniowych, sprawiając, że jest to wszechstronne narzędzie również do bardziej zaawansowanych zastosowań w przypadku wizji komputerowej.

Kolejną z omawianych architektur jest Mask2Former zaprezentowany po raz pierwszy w artykule pt. „Multi-scale masked transformer for image recognition” autorstwa Chen, L. C., Lu, W. C., Yu, Y., Wang, X., & Sun, J. (2021). Architektura ta dedykowana jest zadaniom segmentacji obrazów, w tym segmentacji semantycznej, segmentacji instancji i segmentacji panoptycznej. Innowacyjność tego modelu polega na nowym podejściu do generowania masek, które zapewnia poprawę dokładności i wydajności w porównaniu do wcześniejszych architektur. Warto wymienić najbardziej istotne komponenty Mask2Former, którymi są: Transformer-Based Backbone, Dynamic Mask Prediction, Unified Architecture for Multiple Segmentation Tasks, Query-Based Learning oraz Loss Functions.

Komponent Transformer-Based Backbone oznacza, że Mask2Former korzysta ze szkieletu opartego na sieci typu transformer, co zapewnia efektywne przetwarzanie danych dzięki zastosowaniu mechanizmu self-attention pozwalającego na lepsze uchwycenie

globalnych zależności w obrazach. Kolejny komponent – Dynamic Mask Prediction pozwala na generowanie masek na bieżąco zależnie od kontekstu obrazu zwiększając przy tym elastyczność modelu i precyzję wyników, co jest nowością w porównaniu do tradycyjnych metod segmentacji przewidujących maski wyłącznie dla regionów wcześniej predefiniowanych. Mechanizm Unified Architecture for Multiple Segmentation Tasks pozwala na integrację segmentacji instancji, segmentacji semantycznej i segmentacji panoptycznej w jedną spójną architekturę, dzięki czemu model można stosować do wszystkich tych zadań bez modyfikacji jego struktury. Mechanizm Query-Based Learning polega na wykorzystaniu mechanizmu zapytań (tzw. *queries*) w celu uczenia się masek, a aktualizacja zapytań w czasie treningu daje możliwość lepszego dostosowania modelu do różnych scenariuszy i przez to ulepszyć jakość generowanych masek. Kolejnym mechanizmem jest Loss Function, który umożliwia stosowanie zaawansowanych funkcji straty pomagających w przypadku nie zrównoważonych danych, co poprawia wydajność modelu i wpływa na wzrost efektywności trenowania. Trenowanie Mask2Former odbywa się z bardzo dokładnie dobranymi parametrami typu szybkość uczenia, współczynnik spadku ciężaru, dropout oraz augmentacja danych [Chen i in., 2021].

Architektura Mask2Former pozwala osiągnąć dużą wydajność na zbiorach danych COCO, wykazując przewagę nad wcześniejszymi metodami dotyczącymi segmentacji. Jest potężnym narzędziem w dziedzinie segmentacji obrazów pod kątem badawczym i w projektach wdrożeniowych [Chen i in., 2021].

Kolejną z wykorzystanych w niniejszej rozprawie architektur jest ConvNeXt small należąca do zbioru konwolucyjnych sieci neuronowych. Została ona zaprojektowana w celu zwiększenia precyzji w analizie obrazu oraz zwiększenia wydajności obliczeniowej. Model ten jest połączeniem tradycyjnych sieci konwolucyjnych z użyciem technik zapożyczonych z transformerów, co jak można się spodziewać ułatwia osiągnięcie jeszcze lepszych wyników w detekcji i klasyfikacji obrazów. Autorzy tego modelu sieci mieli za zadanie opracowanie rozwiązania, które będzie się nadawało do pracy z różnymi danymi obrazowymi, a zastosowane techniki augmentacji umożliwiły bardziej dokładne zbadanie złożonych wzorców na obrazach. Dodatkowo, stosowane nowoczesne operacje konwolucji są właściwie skalowane w celu maksymalizacji wykorzystania dostępnych zasobów sprzętowych wykonujących obliczenia [Tan, Le, 2019].

EfficientNet B3 to architektura również zaprojektowana i opublikowana przez tych samych twórców i w tym samym artykule naukowym, co ConvNeXt small [Tan, Le, 2019].

Autorzy w tym przypadku również mieli na celu optymalizację wykorzystania zasobów obliczeniowych zachowując przy tym dużą precyzję osiągania wyników. Najbardziej charakterystyczną cechą EfficientNet B3 jest wykorzystanie nowatorskiego podejścia w celu skalowania sieci uwzględniającego równoczesne zwiększenie parametrów rozdzielczości, głębokości i szerokości sieci. Architektura ta wykazuje szczególną efektywność w analizie obrazów o różnej rozdzielczości w sytuacjach, gdzie ważne jest uzyskanie wysokiej dokładności przy jak najmniejszym zużyciu zasobów obliczeniowych [Tan, Le, 2019]. W kontekście analizy obrazów roślin realizowanej w ramach niniejszej pracy, EfficientNet B3 został zastosowany do klasyfikacji poziomów nawodnienia oraz zdrowotności roślin, co pozwoliło na uzyskanie wyników o wysokiej precyzji.

Kolejną z analizowanych architektur jest Vision Transformer (ViT), który skutecznie przenosi mechanizmy charakterystyczne dla sieci typu transformer z dziedziny przetwarzania języka naturalnego do analizy obrazów. Model ten zajmuje się dzieleniem obrazów na mniejsze fragmenty (tzw. *patche*), przetwarzane następnie jako wejściowe sekwencje tokenów, podobnie jak w przypadku przetwarzania tekstu. Zastosowany w ViT mechanizm transformera umożliwia jednoczesne analizowanie lokalnych i globalnych cech obrazu, co jest istotną rzeczą w zaawansowanych zadaniach klasyfikacji i segmentacji. Model ten sprawdza się najbardziej efektywnie w przypadkach dokładnej detekcji i klasyfikacji jednocześnie wielu obiektów. Vision Transformer umożliwia osiągnięcie wyników podobnych, a czasami wręcz lepszych niż stosowanie klasycznych sieci konwolucyjnych [Dosovitskiy i in., 2020]. Również taki przypadek został zaobserwowany w przeprowadzonych w ramach niniejszej rozprawy badaniach, o czym będzie mowa w dalszej części opracowania.

Przegląd wybranych zaawansowanych architektur zamyka Swin Transformer oparty na modelu Vision Transformer. Architektura ta została wprowadzona w artykule autorstwa Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. pt. "Swin transformer: Hierarchical vision transformer using shifted windows" w 2021 roku. Wprowadza ona mechanizm *shifted windows*, czyli przesuwających się okienek, co umożliwia przetwarzanie obrazów o wysokiej rozdzielczości w sposób efektywny, a to jest istotne w zadaniach związanych z analizą złożonych struktur obrazów. Model ten pokazuje szczególną efektywność w przypadkach wymagających precyzyjnej identyfikacji i klasyfikacji obiektów na obrazach charakteryzujących się wysoką rozdzielczością [Liu i in., 2021]. W kontekście przeprowadzonej w niniejszej rozprawie analizy obrazów roślin, Swin Transformer został



zastosowany do klasyfikacji poziomów nawodnienia oraz zdrowotności roślin, co pozwoliło na uzyskanie wyników o wysokiej precyzji i efektywności.

## 2.2. Przetwarzanie obrazów RGB i wielospektralnych

W niniejszej rozprawie przetwarzanie obrazów skupia się wokół dwóch typów zobrazowań, którymi są dane RGB i dane wielospektralne. Ważne jest więc wyjaśnienie różnic i specyficznych cech obu typów zobrazowań. W tej części pracy omówione zostaną również typowe techniki przetwarzania stosowane w analizie danych obrazowych.

### 2.2.1. Charakterystyka danych RGB

Dane RGB to jeden z najbardziej powszechnych formatów danych obrazowych stosowanych zarówno komercyjnie, jak i w badaniach naukowych. Obrazy tego typu są używane zarówno w przetwarzaniu obrazów, wizji komputerowej, cyfrowej fotografii oraz grafice komputerowej. Jak sama nazwa wskazuje, składają się z trzech kanałów: czerwonego, zielonego i niebieskiego, co z kolei przekłada się na reprezentowanie obrazów w postaci wektorów trójwymiarowych, w których każdy kanał odpowiada za intensywność określonej barwy. Odzwierciedlenie szerokiego spectrum kolorów jest możliwe dzięki kombinacji tych trzech składowych w określonych proporcjach, a to jest ważne w precyzyjnej analizie obrazów, klasyfikacji obiektów i rozpoznawaniu wzorców [Smith, 2012].

Powszechność stosowania obrazów RGB jest skutkiem ich kompatybilności ze zdecydowaną większością urządzeń do akwizycji obrazów, a także do przetwarzania i prezentacji danych obrazowych. Obrazy RGB są podstawowym wyborem w licznych zastosowaniach technologii obrazowych, odznaczają się dużą wiernością odwzorowywania barw, co jest ważne w dziedzinach analizy obrazów medycznych, monitoringu itp. [Jones, 2018].

Oprócz wielu zalet, obrazy RGB wiążą się też z pewnymi ograniczeniami. Są narażone na zakłócenia związane z warunkami oświetleniowymi, a to wpływa na jakość zarówno samych obrazów, jak i ich analizy. Zmieniające się warunki oświetlenia mogą wpłynąć na problemy z interpretacją kolorów i zakłócenia w klasyfikacji i segmentacji obiektów [Brown, 2015]. Dodatkowo, obrazy RGB są narażone na szумы prowadzące do błędów w detekcji i precyzji analizy [White, 2019]. Remedium jest tutaj stosowanie technik obróbki danych takich jak

redukcja szumów czy też normalizacja oświetlenia. Pomimo tych potencjalnych ograniczeń, obrazy RGB są niezastąpionym narzędziem dzięki swej prostocie i dostępności.

### 2.2.2. Charakterystyka danych wielospektralnych

Obrazy wielospektralne rejestrują informacje wizualne w wielu pasmach spektralnych, w tym wykraczających poza zakres widzialny przez ludzkie oko. W odróżnieniu od obrazów RGB mogą rejestrować więcej informacji o fizykochemicznych właściwościach rejestrowanych obiektów. Daje to możliwość prowadzenia poszerzonych badań, niezwykle cennych w takich dziedzinach jak rolnictwo precyzyjne, badania roślin itp. [Jones, 2016].

Obrazy wielospektralne potrafią rejestrować nawet niewielkie różnice w odbiciu światła, będące niewidzialnymi dla oka ludzkiego. Pozwala to na dokładniejszą analizę wielu zjawisk, np. wykrywania chorób roślin, monitorowania gleby, czy też monitorowania stanu nawodnienia roślin. Szczególnie w rolnictwie precyzyjnym, obrazy wielospektralne są stosowane do obliczania typowych wskaźników wegetacji roślin takich jak NDVI, NDRE itp. Zdrowe i chore rośliny inaczej odbijają światło widzialne i bliską podczerwień i stąd możliwe jest obserwowanie tychże różnic [Clark, 2013].

Innym przykładem zastosowania obrazów wielospektralnych jest badanie informacji o nawodnieniu roślin na bazie kanałów spektralnych w paśmie podczerwonym, czy też analiza uszkodzeń struktury powierzchni wybranych materiałów możliwa dzięki analizie pasma ultrafioletowego [Thompson, 2015].

Przykładami technologii wykorzystujących dane wielospektralne są kamery Micasense RedEdge MX Dual oraz MapIR Survey3 RGN + NIR wykorzystane podczas badań prowadzonych podczas przygotowania niniejszej rozprawy. Są one także szeroko stosowane w badaniach nad zdrowiem roślin i monitorowaniem środowiska.

Oprócz niewątpliwych zalet, prowadzenie analizy z wykorzystaniem danych wielospektralnych jest związane z problemami technologicznymi. Co można zauważyć podczas prowadzenia badań, obrazy wielospektralne generują znacznie większe ilości danych niż obrazy RGB, a to pociąga za sobą konieczność posiadania większych zasobów do przechowywania i analizy danych. Dodatkowo, skorzystanie z pełni potencjału danych wielospektralnych pociąga za sobą konieczność stosowania zaawansowanych algorytmów mogących przetworzyć informacje z wielu kanałów i odpowiednio je zinterpretować uwzględniając specyfikę poszczególnych kanałów spektralnych [Jones, 2016].

Pomimo tychże ograniczeń, dane wielospektralne mają ogromny potencjał w badaniach środowiska, w rolnictwie precyzyjnym itp. Ciągłe rozwijanie technik rejestracji tych danych oraz ich przetwarzania daje nowe ścieżki działania w celu prowadzenia precyzyjnej analizy.

### 2.2.3. Techniki przetwarzania danych obrazowych

Przetwarzanie obrazów RGB i wielospektralnych to ważny etap analizy danych obrazowych, który wymaga użycia zaawansowanych technik prowadzących do efektywnej ekstrakcji cech, detekcji i klasyfikacji obiektów oraz zwykłej poprawy jakości obrazów. Wyzwaniem jest potrzeba przetworzenia dużej ilości danych i zmienny charakter tychże danych wymuszający zastosowanie zabiegów, których celem jest wysoka dokładność i zabezpieczenie przed zakłóceniami. Warto po krótko przybliżyć najważniejsze techniki przetwarzania obrazów, jakimi są: segmentacja, filtrowanie, ekstrakcja cech, klasyfikacja obiektów czy też różnego rodzaju transformacje obrazów.

Zaczynając od omawiania procesu segmentacji obrazów, należy podkreślić, że jest to jedno z najważniejszych zadań przetwarzania obrazów polegające na dzieleniu obrazu na mniejsze regiony o podobnych właściwościach związanych z intensywnością barw, tekstur czy też gradientów. Typowe techniki segmentacji w postaci segmentacji opartej o krawędzie, progowanie czy też stosowanie sztucznych sieci neuronowych są stosowane zarówno do obrazów RGB, jak i wielospektralnych. Najbardziej skutecznym narzędziem jest uczenie głębokie, w tym konwolucyjne sieci neuronowe i sieci typowe dla zadań segmentacji, które pozwalają na automatyzację i wzrost dokładności całego procesu. Są stosowane w wielu dziedzinach, od medycyny po rolnictwo [Gonzalez, Woods, 2008].

Kolejną techniką przetwarzania obrazów jest ich filtracja, którą stosuje się w celu poprawienia jakości obrazów czy też usunięcia szumów. Zarówno w przypadku obrazów RGB, jak i wielospektralnych stosuje się filtry dolnoprzepustowe usuwające szumy o wysokiej częstotliwości oraz filtry górnoprzepustowe podkreślające szczegóły o wyższej częstotliwości. Mniej znanymi filtrami są tzw. filtry medianowe stosowane do usuwania szumu zachowując przy tym krawędzi obrazu. Mają one duże znaczenie w przypadku obrazów wielospektralnych, w których precyzja w usuwanie szumów jest bardzo istotna [Jain, 2001].

Jeszcze inną techniką jest ekstrakcja cech obrazów. Umożliwia ona takie przekształcenie danych obrazowych, aby umożliwić klasyfikację i detekcję obiektów. Coraz częściej stosuje się w tym celu uczenie głębokie, którego metody potrafią automatyzować cały

proces ekstrakcji cech, dając złożoną i precyzyjną analizę obrazów. Konwolucyjne sieci neuronowe umożliwiają wydobycie ważnych cech bez konieczności utworzenia deskryptorów, a to podwyższa precyzję detekcji i klasyfikacji [Dalal, Triggs, 2005].

Kolejną z technik jest klasyfikacja obiektów będąca podstawowym zadaniem w dziedzinie przetwarzania obrazów. Polega ona na przypisaniu konkretnej etykiety do rozpoznanych na obrazie obiektów. Techniki klasyfikacji wykorzystujące metody uczenia maszynowego są często stosowane a analizie obrazów, w szczególności konwolucyjne sztuczne sieci neuronowe i ich warianty w postaci ResNet czy EfficientNet znalazły zastosowanie w analizie obrazów RGB i wielospektralnych [Bishop, 2006].

Przykładem kolejnych technik są transformacje obrazów, np. transformacje geometryczne, transformacje przestrzenne itp. Umożliwiają one kompresję danych, korekcję czy też po prostu rejestrację obrazów. Techniki typu transformacja falkowa czy też transformacja Fouriera są szczególnie stosowane w przypadku analizy obrazów wielospektralnych. Innym przykładem jest transformacja Hougha, ważna w detekcji krawędzi i kształtów geometrycznych [Parker, 2010].

## 2.3. Ocena jakości modeli i techniki walidacyjne

Ocena jakości modeli i zastosowanie odpowiednich technik walidacyjnych to kluczowe aspekty w procesie modelowania, szczególnie w kontekście analizy danych obrazowych. Odpowiednie metryki oraz techniki walidacyjne pozwalają na dokładne zrozumienie, jak dobrze model spełnia swoje zadanie, a także na identyfikację obszarów wymagających poprawy. W niniejszym podrozdziale omówiono główne metryki oceny modeli, techniki walidacyjne oraz metody analizy błędów i interpretacji wyników.

### 2.3.1. Metryki oceny (Accuracy, Precision, Recall, F1-score)

Metryka Accuracy jest podstawową metryką oceny definiowaną jako stosunek liczby poprawnie sklasyfikowanych próbek do całkowitej liczby próbek. Jest szczególnie przydatna w przypadku zbiorów danych, w których liczność próbek każdej klasy jest podobna. W przypadku nie zrównoważonych zbiorów danych metryka ta może prowadzić do mylnych wniosków, a model jest w stanie uzyskać wysoką dokładność nieco pomijając mniejszościową klasę [Manning i in., 2008].

Metryka Precision zajmuje się pomiarem przewidywań modelu dla klasy pozytywnej, obliczając stosunek poprawnie zaklasyfikowanych przypadków do całkowitej liczby przypadków określonych jako pozytywne. Duża precyzja określa, że model daje niewiele fałszywych alarmów, co jest ważne wtedy, kiedy koszty tych fałszywych przewidywań są duże [Powers, 2011].

Metryka Recall, czyli czułość bada zdolność modelu do wykrywania rzeczywistych pozytywnych przypadków. Jest określana jako stosunek liczby poprawnie sklasyfikowanych pozytywnych przypadków do wszystkich rzeczywiście pozytywnych przypadków. Wysoka wartość Recall pokazuje, że model rzadko pomija ważne przypadki, a to jest ważne np. w zastosowaniach medycznych [Manning i in., 2008].

Metryka F1-score jest to harmoniczna średnia z Precision i Recall, co sprawia że metryka ta jest bardziej wyważonym parametrem wtedy, gdy ważne jest zrównoważenie obu tych miar. Jest ważna w przypadkach danych, które są niezrównoważone, czyli takich gdzie liczność różnych klas znacząco się od siebie różni. Duża wartość tej metryki pokazuje, że model dobrze radzi sobie z minimalizowaniem fałszywych alarmów oraz z poprawnym klasyfikowaniem przypadków pozytywnych [Sasaki, 2007].

### 2.3.2. Techniki walidacji Hold-out

Proces walidacji modeli uczenia maszynowego jest kluczowy podczas trenowania modeli, a jego celem jest ocenienie zdolności modelu do generalizacji, tzn. skuteczności analizowania danych, których model nie widział podczas treningu. Właściwy wybór odpowiednich technik walidacji jest ważny, aby zminimalizować nadmierne dopasowanie modelu, czyli tzw. *overfitting* oraz aby dokonać jego rzeczywistej skuteczności w warunkach rzeczywistych. Podczas badań prowadzonych w ramach przygotowywania niniejszej rozprawy zastosowano metodę walidacji polegającą na podziale danych na zestawy: treningowy, walidacyjny i testowy. Jest to typowe podejście często stosowane w przetwarzaniu obrazów. Pozwala ono na ocenę modeli zarówno podczas ich dostrajania jak i ewaluacji.

Technika Hold-out, czyli właśnie walidacja z podziałem na zbiory: treningowy, walidacyjny i testowy polega na jednorazowym podziale zbioru danych na przynajmniej dwa zbiory: treningowy oraz walidacyjny. Często wyróżnia się też trzeci zbiór testowy. Spośród tych zbiorów zbiór treningowy służy do nauki modelu, a zbiór walidacyjny jest używany do oceny modelu podczas procesu trenowania i dobierania właściwych hiperparametrów. Zbiór

testowy umożliwia zaś niezależną ocenę modelu po zakończeniu procesu trenowania. Gwarantuje to obiektywną miarę jego generalizacji. Często zbiór walidacyjny jest taki sam jak zbiór testowy.

Technika walidacji typu Hold-out jest prosta w implementacji, ale jej skuteczność jest zależna od wielkości zbiorów danych. Dla małych zbiorów jednorazowy podział może doprowadzić do niestabilnych wyników z powodu losowości w doborze próbek, które trafią do konkretnych zestawów i może to generować niestabilne wyniki walidacyjne. Stąd też w przypadku małych zbiorów danych należy używać bardziej zaawansowanych technik walidacji danych, jak np. cross-validation. W przypadku większych zbiorów danych technika Hold-out jest skuteczna, wiarygodna i pozwala uzyskać stabilne wyniki. Czyni ją to często stosowaną techniką walidacji w zastosowaniach praktycznych [Hastie, Tibshirani, Friedman, 2009].

W ramach badań opisanych w niniejszej rozprawie technika Hold-out z powodzeniem została zastosowana do oceny trenowanych modeli takich jak HTC, Mask2Former, a także ConvNeXt small, ResNet50, Efficient B3, Vision Transformer oraz Swin Transformer. Zastosowanie jej z osobnym zestawem walidacyjnym pozwoliło monitorować skuteczność modeli podczas przebiegu treningu, co z kolei umożliwiło dokładne dostrojenie hiperparametrów i uniknięcie nadmiernego dopasowania względem danych treningowych.

Po fazie treningu ważnym etapem jest walidacja działania modelu na zbiorze walidacyjnym. Podczas tego etapu oceniana jest skuteczność modelu w przewidywaniu na danych, których model nie widział podczas trenowania. Zastosowanie metryk w ocenie modeli, takich jak bbox mAP (Mean Average Precision dla lokalizacji obiektów) oraz segm mAP (Mean Average Precision dla segmentacji), umożliwia dokładną ocenę jakości detekcji obiektów oraz jakości generowanych masek podczas segmentacji. Podczas oceny na zbiorze walidacyjnym można dostosować hiperparametry modelu, np. współczynnik uczenia learning rate, liczbę epok i inne parametry treningowe, aby zoptymalizować działanie modelu.

Finalna ocena modelu jest realizowana za zbiorze testowym, niezależnym od fazy treningowej. Czasami jako zbiór testowy podaje się ten sam zbiór co zbiór walidacyjny, ponieważ jego również model „nie widzi” w czystym etapie trenowania na zbiorze treningowym. Zbiór testowy jest wykorzystywany do końcowego oszacowania zdolności modelu do generalizacji. Wyniki uzyskane na tym zbiorze dają miarodajny wskaźnik efektywności modelu, co jest ważne w zadaniach takich jak np. detekcja faz rozwojowych



roślin, analiza nawodnienia oraz porażenia roślin patogenami [Hastie, Tibshirani, Friedman, 2009].

Największą zaletą metody Hold-out jest jej prostota i bardzo łatwa implementacja. Metoda ta umożliwia szybkie trenowanie oraz ocenę modelu na dużych zbiorach danych, co jest użyteczne w przypadku chęci szybkiego przeprowadzenia eksperymentu. Technika ta ma również swoje ograniczenia, ponieważ dla małych zbiorów danych jest ryzyko osiągnięcia niestabilnych wyników z powodu przypadkowego doboru próbek trafiających do zbioru walidacyjnego i testowego. Może to wpłynąć na wyniki, więc wówczas warto stosować inne techniki, np. f-fold cross-validation albo stratified cross-validation minimalizujące ryzyko błędów spowodowanych losowym podziałem danych. W zastosowaniach typu rozpoznawanie faz rozwojowych roślin, nawodnienia oraz porażenia roślin, technika Hold-out pozwala na szybką i dokładną ocenę modeli, ale warunkiem jest dysponowanie wystarczająco dużym zbiorem danych.

### 2.3.3. Analiza błędów i interpretacja wyników

Podczas analizy wyników uzyskiwanych przez modele uczenia maszynowego główne znaczenie mają metryki oceny, takie jak Accuracy, Precision, Recall i F1-score, ale nie tylko. Czasami ważna jest również pogłębiona analiza błędów podczas predykcji. Umożliwia ona poznanie obszarów wymagających poprawy. Przypadki *false positive*, czyli te, w których model błędnie klasyfikuje mało istotne elementy obrazu jako obiekty, których poszukuje, wynikać mogą ze zbyt niskiej jakości obrazów bądź też złożoności tła. Dla przykładu, w zadaniu związanym z detekcją określonych roślin, model może mylnie rozpoznać inne elementy natury jako poszukiwane rośliny [Goodfellow i in., 2016].

Z kolei przypadki *false negative*, czyli fałszywie negatywne polegające na pominięciu poszukiwanych obiektów również mogą być niekorzystny, np. w zadaniu, gdzie poszukuje się objawów porażenia roślin. Takie błędy mogą być związane ze zbyt małym zbiorem treningowym dla danych klas lub ze zbyt mało różnorodnego zbioru danych. Rozwiązaniem jest tutaj oczywiście zwiększenie zbioru treningowego, zwłaszcza dla mniej licznych klas lub wprowadzenie dodatkowych technik augmentacji danych [He i in., 2017].

Kolejnym ważnym elementem analizy wyników i ich interpretacji są przypadki graniczne w postaci predykcji charakteryzujących się niską pewnością. Mogą one wskazywać na niezdolność modelu do rozpoznawania sytuacji, które nie są jednoznaczne. Przykładem

mogą być tutaj obrazy o niskiej jakości, np. w zadaniu dotyczącym nawodnienia roślin zbyt słabe warunki oświetlenia mogą wywoływać efekt nieprawidłowego oceniania stanu nawodnienia przez model. Przeanalizowanie tego typu przypadków umożliwia pozyskanie cennych informacji na temat tego, co sprawia problemy modelowi. Rozwiązaniem tego problemu jest zastosowanie dodatkowych technik augmentacji danych oraz zwiększenie różnorodności zbioru danych treningowych [Powers, 2011].

Warto wspomnieć również o problemie *overfittingu*, czyli nadmiernym dopasowaniu, które można zbadać poprzez monitorowanie różnic w wynikach pomiędzy zbiorem treningowym, walidacyjnym i testowym. Gdy model daje dobre wyniki za zbiorze treningowym, ale dużo gorsze na zbiorze walidacyjnym i testowym, to może pokazywać, że model zbyt mocno nauczył się wzorców charakterystycznych dla zbioru treningowego, ale nietypowych dla zbioru walidacyjnego i testowego. W takim przypadku można użyć dodatkowych technik regularyzacji w celu poprawy zdolności modelu do generalizacji [Goodfellow i in., 2016]. Można np. monitorować wykresy `train_loss(steps)` i `val_acc(steps)`, co daje możliwość śledzenia postępów treningu i wykrycie oznak *overfittingu* [Zhang, Sabuncu, 2018].

Analiza błędów może wskazać rady dotyczące optymalizacji modelu. Zwiększenie zbioru treningowego oraz zastosowanie dodatkowych technik augmentacji danych może poprawić wyniki modelu, zwłaszcza przy trudnych warunkach oświetlenia [He i in., 2017]. Dodatkowo, właściwy dobór hiperparametrów modelu, np. współczynnika uczenia `learning rate`, liczby epok, głębokości sieci, może poprawić wyniki w procesie trenowania. Również śledzenie wykresów `bbox_mAP50(epoch)` może dać cenne informacje na temat skuteczności modelu [Goodfellow i in., 2016].

Analiza błędów zależy od praktycznych zastosowań opracowywanego modelu. W przypadku detekcji faz rozwojowych, nawodnienia i porażenia roślin, odpowiednia wiedza na temat przyczyn błędnych predykcji jest ważna w perspektywie dalszego udoskonalania modelu i wdrażania go do rzeczywistego projektu. Optymalizacja modelu umożliwia wzrost jego skuteczności w zastosowaniu do monitoringu upraw i identyfikacji przypadków alarmowych na wcześniejszym etapie, zanim będą widoczne [He i in., 2017].

## 2.4. Zastosowanie skali BBCH, nawodnienia i identyfikacji patogenów w analizie obrazów

Analiza skali BBCH, ocena nawodnienia oraz identyfikacja patogenów są kluczowymi czynnikami pozwalającymi na dokładną ocenę kondycji roślin. W tym podrozdziale omówione zostaną te aspekty od strony teoretycznej, przybliżone zostanie ich znaczenie i sposoby praktycznego wykorzystania w analizie obrazów.

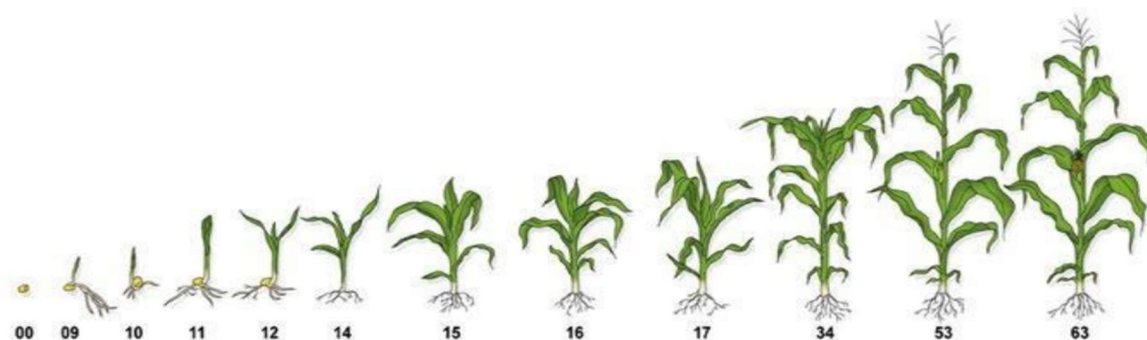
### 2.4.1. Skala BBCH i jej zastosowanie w analizie faz rozwojowych roślin

Skala BBCH (skrót od niemieckiej nazwy Biologische Bundesanstalt, Bundessortenamt und Chemische Industrie) to uniwersalny system opisujący stopień rozwoju roślin poprzez określenie fazy rozwojowej danej rośliny, począwszy od kiełkowania aż po pełną dojrzałość. Skala ta jest często wykorzystywana w badaniach naukowych dotyczących monitorowania rozwoju roślin oraz bezpośrednio w rolnictwie. Powodem jej opracowania było ujednolicenie systemu odnoszącego się do oceny faz rozwojowych różnych gatunków roślin, co umożliwia dalsze porównywanie podczas badań w różnych warunkach środowiska [Meier, 2001].

Skala BBCH ma szerokie zastosowanie w precyzyjnym rolnictwie, gdy dokładne monitorowanie aktualnego rozwoju roślin ma istotne znaczenie w doborze właściwych zabiegów agrotechnicznych typu nawadnianie, czy stosowanie środków ochrony roślin. W przypadku prowadzenia badań z wykorzystaniem obrazów, skala BBCH umożliwia automatyczną detekcję i klasyfikację faz rozwojowych tylko na podstawie danych wizualnych. Wykorzystanie algorytmów przetwarzania obrazów w postaci sztucznych sieci neuronowych umożliwia prowadzenie monitoringu etapów rozwoju roślin w sposób automatyczny i co więcej, precyzyjny. Określanie faz rozwoju roślin nie jest jednak zadaniem samym w sobie, gdyż służy w dalszej kolejności również do zbadania wpływu czynników środowiska, zabiegów agrotechnicznych itp. na szybkość rozwoju roślin [Lancashire i in., 1991].

W skali BBCH można doszukać się kilku głównych metryk dotyczących m.in. rozwoju liści, kłosów, wiechy, kolb itp. W eksperymentach prowadzonych w niniejszym badaniu wykorzystana została skala liściowa, która w łatwy sposób pozwala określić aktualną fazę rozwoju kukurydzy na podstawie liczby posiadanych przez daną roślinę liści. Skala liściowa określana jest za pomocą dziesięciu liczb od 10 do 19, gdzie cyfra dziesiątek świadczy o typie skali (w tym przypadku liściowej), a cyfra jedności odnosi się to do w pełni rozwiniętych liści, z wyłączeniem zalążków liści w postaci zwiniętej, które znajdują się w początkowej fazie

rozwoju i nie osiągnęły jeszcze pełnej długości ani funkcjonalności. Zalążki te nie są uwzględniane jako pełne liście w ocenie rozwoju rośliny. I tak, klasa 10 oznacza pojedynczy zalążek liścia w postaci zwiniętej wyrastający z ziemi, klasa 11 oznacza zalążek liścia w postaci zwiniętej plus jeden wykształcony liść, a klasa 19 oznacza zalążek liścia w postaci zwiniętej oraz minimum 9 w pełni wykształconych liści (w przypadku roślin posiadających więcej niż 9 liści, przypisuje się im również klasę BBCH rozwoju liści równą 19). W przypadku analizy obrazów, detekcja i klasyfikacja faz rozwojowych roślin umożliwia dalszy monitoring szybkości wzrostu roślin i identyfikację potencjalnych anomalii. Poniżej zamieszczono przykładowy schemat określania poszczególnych faz rozwojowych kukurydzy w skali BBCH oraz przykładowy obraz kukurydzy będącej w 16 fazie rozwoju w skali BBCH.



**Rysunek 2.8.** Skala BBCH w ujęciu graficznym (źródło: [https://pdf.helion.pl/e\\_1wwu/e\\_1wwu.pdf](https://pdf.helion.pl/e_1wwu/e_1wwu.pdf)) [Helion, 2024].



**Rysunek 2.9.** Przykład obrazu kukurydzy będącej w 16 fazie rozwoju w skali BBCH.

#### 2.4.2. Znaczenie nawodnienia roślin w kontekście analizy obrazów

Stopień nawodnienia roślin jest niewątpliwie jednym z najistotniejszych czynników wpływających na dynamikę ich wzrostu oraz ogólny stan zdrowia. Z punktu widzenia analizy obrazów, ocena stanu nawodnienia roślin może opierać się przede wszystkim na aspektach wizualnych, czyli np. na miąższości liści, będącej wynikiem odpowiedniego turgoru. Pozwala to na klasyfikację roślin w trzy główne klasy: rośliny przesuszone, rośliny średnio nawodnione oraz rośliny dobrze nawodnione. Dobrze nawodnione rośliny o mięsistych liściach i łodygach odznaczają się wysokim turgorem, a rośliny z niedoborem wody okazują zwiotczenie tkanek i utratę sprężystości z powodu niskiego turgoru [Jones, 2004].

W przeprowadzonych w ramach niniejszej rozprawy badań z wykorzystaniem głębokich sieci neuronowych oraz analizy i przetwarzania obrazów RGB i wielospektralnych, oznaki wizualne nawodnienia roślin są podstawą do ich klasyfikacji. Sztuczne sieci neuronowe uczą się rozpoznawania subtelnych różnic w zewnętrznym wyglądzie roślin, co pozwala na ich detekcję i klasyfikowanie pod względem stopnia nawodnienia [Berger et al., 2010].

W przypadku kukurydzy oznaki niedoboru wody mogą być wyraźne, a analiza obrazów daje możliwość dokładnego monitorowania tychże zmian. Jest to ważne z punktu widzenia rolnictwa precyzyjnego, w którym właściwe zarządzanie zasobami wody ma bezpośredni wpływ na plonowanie.

#### 2.4.3. Identyfikacja patogenów roślinnych przy użyciu analizy obrazów

Wczesne wykrywanie patogenów atakujących rośliny jest niemiernie istotne ze względu na możliwość szybkiej reakcji i uruchomienie zbiegów zaradczych, które pozwalają na minimalizację efektu obniżenia plonów. Tradycyjne rozpoznawanie chorób roślin dotyczy ich wizualnej inspekcji, co niewątpliwie może być czasochłonne, ponieważ to, że w danej części pola uprawnego nie ma patogenów, nie oznacza, że identyczna sytuacja jest w całym analizowanym obszarze. Rozwój metod przetwarzania obrazów cyfrowych oraz technik sztucznej inteligencji wpłynął na zwiększenie roli autonomicznych systemów w diagnostyce roślin, pozwalających na dokładną i szybszą identyfikację objawów porażenia roślin patogenami [Bock et al., 2010].

W niniejszej rozprawie skupiono się na wykrywaniu porażenia roślin patogenami wyłącznie na bazie analizy obrazów RGB, bez wykorzystania obrazów wielospektralnych, z

uwagi na ograniczony dostęp do poletek badawczych z porażoną kukurydzą. Stąd też rozważania teoretyczne dotyczą głównie obrazowania RGB. Analizowanie obrazów RGB pozwala na detekcję wizualnych objawów porażenia roślin patogenami, takich jak zmiany barwy, plamki na liściach czy też zniekształcenia i deformacje liści. Zmiany te są wywoływane przez działanie czynników patogennych na strukturę roślin i pojawienie się widocznych uszkodzeń. Dla przykładu, analizowana w ramach badań opisanych w niniejszej rozprawie Helminthosporioza objawia się charakterystycznymi przebarwieniami na liściach, które bardzo łatwo identyfikować można na obrazach RGB [Mahlein, 2016].

Modele uczenia maszynowego mogą pomóc w szybkim i dokładnym klasyfikowaniu zdrowych i porażonych roślin na bazie danych obrazowych. Modele wykorzystujące sztuczne sieci neuronowe, w tym modele uczenia głębokiego mogą uczyć się różnic w wyglądzie roślin zdrowych i porażonych patogenami, co pozwala na wczesne wykrycie symptomów chorobowych, które mogą dotyczyć koloru liści, ich deformacji oraz uszkodzeń mechanicznych, jak np. w przypadku porażenia owadem *Spodoptera Frugiperda* [Zhang i in., 2019].

Wykorzystanie obrazowania RGB ma istotne znaczenie podczas monitorowania roślin na szerszą skalę, gdyż umożliwia przetwarzanie dużych ilości danych obrazowych w stosunkowo krótkim czasie. Mimo ograniczeń wynikających z braku informacji spoza zakresu światła widzialnego, analizowanie obrazów RGB wciąż jest ważną metodą w diagnostyce roślin, a zwłaszcza, gdy można zastosować tutaj zdobycze sztucznej inteligencji. Dzięki niej umożliwiające jest wykrywanie objawów porażenia na wczesnym etapie i tworzenie prognoz rozprzestrzeniania się potencjalnych chorób, co jest niezwykle ważne w procesie zarządzania zdrowiem roślin w rolnictwie precyzyjnym [Bock et al., 2010].

## 2.5. Przegląd podobnych rozwiązań

Określanie faz rozwojowych roślin, a w przypadku opisywanych badań - kukurydzy, jest istotna w odniesieniu do determinacji parametrów jej kondycji [Meier, 2001]. Zgodnie z ideą rolnictwa precyzyjnego, oprócz samego faktu określania kondycji roślin, ważne jest również robienie tego w sposób zautomatyzowany, z precyzją rzędu pojedynczych roślin lub niewielkich stref wchodzących w skład wielkohektarowych upraw [Mahlein, 2016]. To przekłada się z kolei na utrzymanie właściwej ciągłej kontroli upraw w celu utrzymania



właściwych dla rozwoju roślin warunków i co za tym idzie wysokiej jakości warunków i co za tym idzie wysokiej jakości wyprodukowanej żywności.

Określanie zarówno faz rozwojowych roślin, jak i poziomów ich nawodnienia oraz wykazywania objawów porażenia wybranymi patogenami odbywało się do tej pory przede wszystkim w sposób ręczny, który w przypadku wielkopowierzchniowych upraw jest czasochłonny, a co za tym idzie również kosztowny.

Automatyczne określanie faz rozwojowych roślin, a na tej podstawie także ich kondycji, zyskało na znaczeniu dzięki wykorzystaniu metod uczenia głębokiego oraz przetwarzania i analizy obrazów cyfrowych. Podczas, gdy tradycyjne metody skupiają się na ręcznym badaniu roślin i na tej bazie notowaniu ich faz rozwojowych, nowoczesne podejście skierowane jest na automatyzm tego procesu z wykorzystaniem uczenia się reprezentacji na podstawie zadanych zbiorów obrazów [Lancashire i in., 1991]. Uczenie się reprezentacji przeprowadzane jest w celu predykcji wyników i przypisywania roślinom odpowiednich klas w postaci wartości fazy rozwojowej w skali BBCH, stopnia nawodnienia, czy też porażenia wybranymi patogenami. W tej dziedzinie inni badacze zaproponowali już liczne rozwiązania, których przegląd został omówiony w kolejnym podrozdziale i który to jest punktem wyjścia do identyfikacji obecnych luk badawczych.

### 2.5.1. Przegląd literatury i analiza istniejących rozwiązań

Rozwiązaniem, które warto przytoczyć podczas przeglądu istniejących rozwiązań w ramach automatycznego monitorowania faz rozwojowych kukurydzy jest artykuł Xingmei Xu i współautorów z 2020 roku [Xu, X., i in., 2020]. Jego autorzy opracowali dwustopniową metodę detekcji liści kukurydzy polegającą na segmentacji instancji oraz wykrywaniu obiektów przy użyciu Mas R-CNN i YOLOv5. Metoda oparta na obrazach RGB pozyskanych za pomocą BSP pokazała swą skuteczność w dokładnym wykrywaniu i zliczaniu liści radząc sobie z tłem i obecnością chwastów na polu uprawnym.

Kolejnym przykładem użycia obrazowania RGB w celu monitoringu faz rozwojowych roślin kukurydzy jest praca Minguo Liu i współautorów z 2019 roku, w której badacze stworzyli system pomiaru wschodów kukurydzy [Liu, M., i in., 2019]. Rozwiązanie to wykorzystuje algorytm YOLO oraz metodę TOPSIS, umożliwiając precyzyjną analizę liczby siewek, łącznie z ich wielkością, jednolitością i rozmieszczeniem, radząc sobie przy tym z wpływem cieni i

różną gęstością sadzenia. Badanie to również eksponuje potencjał monitoringu roślin za pomocą BSP [Liu, M., i in., 2019].

Innym rozwiązaniem jest z kolei opracowanie Danyang Yu i współautorów z 2021 roku [Yu, D., i in., 2021]. Badacze w swojej pracy zastosowali wieloźródłowe obrazy pozyskiwane za pomocą BSP z wykorzystaniem głębokich konwolucyjnych sieci neuronowych do określania nadziemnej biomasy kukurydzy. Uzyskane wyniki pokazują, iż głębokie sztuczne sieci konwolucyjne są skuteczne w szacowaniu nadziemnej biomasy w różnych fazach rozwoju roślin, co przekłada się na precyzyjne przewidywanie wielkości plonów [Yu, D., et al., 2021].

Kolejnym przykładem interesującego rozwiązania jest praca Xuli Zan i współautorów z 2018 roku [Zan, X., i in., 2018]. Badacze skupili się w niej na detekcji wiech kukurydzy, wykorzystując obrazowanie RGB pozyskiwane za pomocą BSP analizując dane z użyciem algorytmu random forest oraz sztucznej sieci neuronowej VGG16. Opracowana metoda pokazała swą skuteczność w dokładnej detekcji wiech kukurydzy w fazie jej kwitnienia. Jest to ważny element oceny cech fenotypowych kukurydzy i jest potrzebny w zarządzaniu produkcją nasion [Zan, X., i in., 2018].

Następnym badaniem z puli analizowanych przykładów jest praca Mahlein i współautorów z 2016 roku [Mahlein, A. K., 2016]. Badacze zastosowali obrazy RGB i hyperspektralne do detekcji chorób roślin. Przeprowadzona analiza obrazów pozwoliła na wczesną detekcję patogenów, w tym grzybów, bakterii i wirusów, dzięki analizie zmian barwy i struktury powierzchni roślin. Praca udowodniła, że systemy teledetekcji wykorzystujące obrazy mogą istotnie zwiększać efektywność procesu monitoringu zdrowotności roślin.

W przypadku analizy zdrowotności roślin podobne rozwiązanie zaprezentowali Bock i współautorzy opracowania z 2010 roku [Bock, C. H., i in., 2010]. Autorzy porównali skuteczność różnych metod analizy obrazów, począwszy od obrazów RGB po hyperspektralne, aby prowadzić detekcję chorób roślin. Uzyskane wyniki pokazały, że metody teledetekcji są w stanie skutecznie monitorować wielkoobszarowe plantacje, dając przy tym możliwość wykrycia przypadków alarmowych [Bock, C. H., i in., 2010].

Jeszcze innym przypadkiem badania znalezione podczas przeglądu literatury jest artykuł pt. *Deep Learning for Maize Growth Stage Classification Based on RGB Images* [Zhao i in., 2020]. Badacze użyli do klasyfikacji faz wzrostu kukurydzy obrazowanie RGB i zastosowanie sieci ResNet. Opracowany model pokazał dużą skuteczność w detekcji faz

rozwijających roślin, co ma istotny wkład w monitorowanie upraw w sposób automatyczny na wielkoobszarowych uprawach [Zhao i in., 2020].

Jak widać na podstawie przytoczonych publikacji, do automatycznego określania faz rozwoju kukurydzy oraz parametrów związanych z jej zdrowotnością szeroko stosowane jest obrazowanie wykonywane za pomocą bezzałogowych statków powietrznych (BSP) oraz rozwiązania z dziedziny uczenia głębokiego. Zdecydowana większość badań skupia się na przetwarzaniu i analizie obrazów RGB, lecz obrazowanie multispektralne również może dostarczyć bardziej dokładnych informacji o kondycji roślin.

### 2.5.2. Wnioski i identyfikacja luk badawczych

Przegląd literatury pod kątem aktualnych rozwiązań w dziedzinie detekcji i klasyfikacji faz rozwoju roślin, w tym kukurydzy, klasyfikacji poziomów jej nawodnienia oraz wykrywania objawów porażenia wybranymi patogenami pokazał, że szczególne zastosowanie w tym temacie znalazły techniki teledetekcji oraz uczenia głębokiego, co wpisuje się w ideę rolnictwa precyzyjnego. Szersze przemyslenia pozwoliły natomiast na zidentyfikowanie pewnych luk badawczych, które wymagają dalszego rozwoju lub wypełnienia od podstaw.

Po pierwsze, znakomita większość badań opiera się na obrazowaniu pozyskiwanym za pomocą BSP, co niewątpliwie ogranicza dokładność monitorowania pojedynczych roślin. Warto zastosować obrazowanie z bliskiej odległości, co daje perspektywę bardziej szczegółowej analizy kondycji roślin (w tym określania faz rozwojowych, poziomów nawodnienia i potencjalnego porażenia wybranymi patogenami). Oczywiście konieczne w tym scenariuszu jest zastosowanie innych urządzeń do pobierania zobrażeń niż BSP. Chodzi tutaj o roboty polowe, urządzenia lub pojazdy, które mogą poruszać się na polu uprawnym. To może niewątpliwie wpłynąć na wzrost dokładności detekcji i jej znacznie większą szczegółowość. Taki też cel przyświecał badaniom prowadzonym w ramach niniejszej rozprawy.

Po drugie, obecnie prym wiodzie zdecydowanie obrazowanie RGB, a obrazowanie wielospektralne wciąż występuje rzadziej, pomimo faktu, że może dostarczyć bardziej kompleksowych danych na temat kondycji roślin. Obrazowanie tego typu może potencjalnie pomóc w detekcji delikatnych zmian, które nie są jeszcze widoczne w spektrum światła widzialnego, a to czyni tę technikę bardzo obiecującą. Stąd też w prowadzonych badaniach starano się wypełnić tę lukę badawczą.

Po trzecie, obecnie wciąż brakuje dostępnych zbiorów danych treningowych dotyczących faz rozwojowych roślin w skali BBCH, w tym kukurydzy oraz zbiorów danych zawierających obrazy roślin będących w różnych warunkach nawodnienia. Niewątpliwie stanowi to lukę badawczą, gdyż stworzenie tego typu datasetów nie jest łatwe i wymaga zaprzężenia dodatkowych prac polowych i regularnego pozyskiwania zobrazowań z plantacji o określonych warunkach nawodnienia i o określonych etapach rozwoju rosnących na tymże polu roślin. W realizowanych badaniach opracowane zostały dedykowane datasety z obrazami kukurydzy będącej w określonych fazach rozwojowych oraz w określonych poziomach nawodnienia. Wypełniają one zauważoną lukę badawczą i mogą stanowić istotny wkład w rozwój nauki, umożliwiając dalsze badania.

Po czwarte, zbiór architektur wykorzystywanych podczas dodatkowych badań z użyciem uczenia głębokiego nie został jeszcze wyczerpany. Wciąż istnieje jeszcze spora grupa algorytmów, która nie została zbadana pod kątem wykorzystania w zadaniach związanych z detekcją i klasyfikacją faz rozwojowych roślin, z detekcją poziomów nawodnienia oraz z detekcją porażenia wybranymi patogenami. Stąd też wykorzystane zaawansowane architektury typu HTC, czy przykłady sieci typu transformer niewątpliwie wypełniają dotychczasową lukę badawczą.

### 3. Materiały i metody badawcze

W niniejszym rozdziale przedstawiono szczegółowe informacje dotyczące użytych danych, w tym procesu przygotowywania pól testowych, pozyskiwania tychże danych, procesu ich oznaczania oraz zastosowanych metod badawczych. Opisano poszczególne rodzaje danych obrazowych w postaci obrazów RGB, obrazów wielospektralnych z kamery RedEdge oraz obrazów wielospektralnych z kamery MapIR, które posłużyły jako podstawa do przeprowadzonych badań i analiz. W kolejnej części przybliżono architektury głębokich sieci neuronowych wykorzystanych do realizacji procesu detekcji i klasyfikacji obiektów na obrazach. Omówiono metodykę implementacji tychże architektur oraz przyjęte parametry treningu i dostosowanie modeli. W dalszej części przybliżono proces detekcji i klasyfikacji obiektów, w tym implementację i zastosowanie modeli oraz omówiono metodę opracowanego klasyfikatora głosującego, który został wykorzystywany do polepszania wyników detekcji i klasyfikacji z wykorzystaniem modeli sztucznych głębokich sieci neuronowych wytrenowanych na pojedynczych kanałach spektralnych. Rozdział ten ma za zadanie przybliżenie wszystkich typów zobrazowań wykorzystywanych w niniejszej rozprawie doktorskiej, w tym sposobu ich pozyskiwania, a także metod wykorzystywanych do prowadzenia procesu trenowania sztucznych sieci neuronowych oraz docelowej detekcji i klasyfikacji obiektów.

#### 3.1. Przyjęta metodyka przygotowywania danych oraz prowadzenia badań

Na wstępie prac badawczych ustalona została ogólna metodyka badań, która została następnie uszczegółowiona na podstawie precyzyjnych zaleceń wynikających z wymagań stricte technicznych dotyczących poszczególnych zagadnień.

Na bazie dostępności sprzętu, warunków środowiskowych oraz na bieżąco monitorowanych powiązań między poszczególnymi zadaniami realizowanymi w ramach prac projektowych, wyłoniony został zestaw punktów składających się na kompletną metodykę badań. Poniżej zostały one wymienione w sposób chronologiczny, zgodnie z realizacją następujących po sobie poszczególnych zagadnień:

- a) Utworzenie niezbędnych pól testowych w celu zapewnienia możliwości pobrania pełnego materiału obrazowego

Jest to punkt niezbędny na drodze po późniejszego pozyskiwania materiału badawczego w postaci obrazów RGB oraz obrazów wielospektralnych. Na poletkach testowych należało stworzyć odpowiednie warunki do siewu, uprawy i wzrostu kukurydzy, która została wybrana jako roślina, której uprawę będzie wspomagał robot polowy. Poletka testowe musiały charakteryzować się odpowiednią wielkością oraz położeniem umożliwiającym codzienne pobieranie obrazów, aby zachować ciągłość monitoringu. Szczegółowe zasady oraz dane na temat utworzonych poletek testowych znajdują się w rozdziale 3 niniejszej rozprawy.

b) Dobór kamer oraz odpowiedniej pozycji pozyskiwania obrazów

Jest to punkt kluczowy w procesie pobierania obrazów oraz powiązania wymagań projektowych z dostępnością sprzętu znajdującego się w posiadaniu Działu Teledetekcji Sieci Badawczej Łukasiewicz – Instytutu Lotnictwa, w którym został zrealizowany doktorat wdrożeniowy, którego efektem jest niniejsza rozprawa. Sprzęt musiał być identyczny lub bardzo zbliżony do tego, który będzie docelowo zamontowany na prototypie robota polowego, a jednocześnie musiał być dostępny przez cały okres pobierania obrazów podczas trwania projektu. Dodatkowo, musiał charakteryzować się względną niezawodnością oraz stabilnym trybem pracy. Kolejnym, ważnym aspektem jest dobór odpowiedniej pozycji pozyskiwania obrazów, która powinna być tożsama z pozycją danej kamery zamontowanej na prototypie opracowywanego robota polowego.

c) Pozyskanie niezbędnych obrazów w odpowiednim przedziale czasowym

W przypadku pozyskiwania obrazów roślin uprawianych w warunkach polowych niezwykle istotne jest pobieranie obrazów podczas okresu wegetacyjnego roślin. Drugim ważnym parametrem jest odpowiednia częstotliwość pobierania obrazów, aby można było udokumentować zmieniające się parametry kondycji roślin takie jak wartość fazy rozwojowej, poziom nawodnienia oraz to czy dana roślina wykazuje oznaki porażenia wybranymi patogenami.

d) Dobór narzędzia do oznaczania zebranych obrazów oraz sukcesywne oznaczanie zebranego materiału

Punkt ten dotyczy wyboru optymalnego narzędzia służącego do przeprowadzania procesu oznaczania obrazów celem późniejszego ich wykorzystania w procesie trenowania sztucznych sieci neuronowych do rozpoznawania obrazów.

e) Dobór sprzętu obliczeniowego

Po etapie pobierania i gromadzenia materiału obrazowego pozyskanego z wcześniej przygotowanych pól testowych niezmiernie ważny jest dobór odpowiedniego sprzętu obliczeniowego, który posłuży do realizacji procesu analizy pozyskanego materiału obrazowego oraz trenowania sztucznych sieci neuronowych.

f) Instalacja niezbędnego oprogramowania oraz bibliotek do analizy i przetwarzania obrazów

Na dobranym odpowiednio sprzęcie obliczeniowym należało zainstalować odpowiednie oprogramowanie wraz z szeregiem niezbędnych bibliotek, które pozwolą na przeprowadzenie procesu trenowania sztucznych sieci neuronowych oraz generowania i analizy uzyskanych wyników, a także prowadzenia ich dalszych przetworzeń.

g) Dobór algorytmów uczenia głębokiego opartych o architekturę konwolucyjnych sieci neuronowych

Algorytmy umożliwiające trenowanie sztucznych sieci neuronowych nie mogły być dobrane w sposób dowolny. Należało wybrać te algorytmy, które w dziedzinie rozpoznawania obrazów znajdują się w czołówce obecnych rozwiązań, czyli tzw. state of the art. wykrywania obiektów na obrazach.

h) Realizacja procesu trenowania sztucznych głębokich sieci neuronowych

Dobór odpowiedniego sprzętu obliczeniowego, zainstalowanie właściwego oprogramowania z niezbędnymi bibliotekami oraz przygotowanie danych, na których będą prowadzone analizy tworzą razem niezbędne wymagania do rozpoczęcia procesu trenowania głębokich sieci neuronowych do rozpoznawania obrazów.

i) Analiza wyników

Kolejnym bardzo ważnym punktem metodologii badań jest analiza uzyskanych wyników. Ma ona na celu wyłonienie najbardziej skutecznego rozwiązania w dziedzinie rozpoznawania obiektów na obrazach oraz obróbkę uzyskanych wyników w odpowiedni sposób, który zapewni płynny przepływ danych do bazy danych.

j) Opracowanie aplikacji realizującej wszystkie opracowane zagadnienia badawcze

Punkt ten mówi o aplikacji wizualizującej opracowane rozwiązania i pozwalającej na testowanie wytrenowanych głębokich sieci neuronowych na dowolnych obrazach podanych na wejściu aplikacji. Etap ten pozwala na wygodne sprawdzenie działania opracowanych



algorytmów na rzeczywistych obrazach, których sztuczne sieci neuronowe „nie widziały” podczas treningu.

- k) Opracowanie architektury rozwiązania, które zapewni generowanie odpowiedniego pliku wyjściowego, w którym będą znajdowały się wszystkie odpowiedzi sztucznych sieci neuronowych oraz niezbędne parametry procesu wykrywania obiektów, które następnie mogły trafić do odpowiedniej bazy danych.

Chodzi tutaj o system do generowania plików w formacie .csv, w których będą zapisywane wszystkie odpowiedzi sztucznych sieci neuronowych podczas działania opracowanego rozwiązania w czasie rzeczywistym oraz dodatkowy zapis innych niezbędnych danych, takich jak czas przetworzenia, nazwa analizowanego obrazu oraz wartość timestamp.

- l) Testy i wdrożenie rozwiązania na rzeczywistym prototypie robota polowego.

Punktem podsumowującym metodologię badań były testy i wdrożenie opracowanego rozwiązania na rzeczywistym prototypie robota polowego. Pozwalają one na zrewidowanie poprawności i dokładności opracowanego rozwiązania oraz na posługiwaniu się uzyskanymi wynikami do dalszych zadań przewidzianych w projekcie.

Poszczególne punkty przedstawionej metodologii badań zostały rozszerzone i uszczegółowione w kolejnych częściach niniejszego opracowania.

## 3.2. Opis danych obrazowych

W sekcji tej zostaną omówione szczegóły na temat danych, które były przedmiotem prowadzonych badań. Omówione zostały kolejno takie kwestie jak: źródła danych, przygotowanie odpowiednich pól testowych, szczegóły na temat procesu pozyskiwania, magazynowania, hierarchizacji i przetwarzania danych RGB, danych z kamery wielospektralnej RedEdge oraz danych z kamery wielospektralnej MapIR. Opis tychże danych jest kluczowy dla zrozumienia kontekstu badawczego oraz późniejszych etapów ich analizy i przetwarzania.

### 3.2.1. Źródła danych i przygotowanie pól testowych

Obrazy pozyskiwane w ramach prowadzonych prac miały posłużyć trzem głównym zadaniom postawionym w ramach prac badawczych i wdrożeniowych, a były nimi: detekcja i

klasyfikacja faz rozwoju kukurydzy, detekcja i klasyfikacja poziomów nawodnienia kukurydzy oraz detekcja i klasyfikacja kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami.

Pozyskiwanie obrazów w celu realizacji zadania mającego na celu detekcję kukurydzy i klasyfikację jej faz rozwojowych miało miejsce na przestrzeni trzech sezonów wegetacyjnych: 2021, 2022 oraz 2024. W sezonie wegetacyjnym 2021 przygotowano dwa poletka testowe: jedno - w szklarni oraz drugie - na otwartym polu uprawnym. Z kolei w sezonach wegetacyjnych 2022 i 2024 przygotowano większe poletka testowe na otwartym polu uprawnym.



**Rysunek 3.1.** Widok szklarni z zewnątrz i wewnątrz, w której prowadzona była uprawa kukurydzy w sezonie wegetacyjnym 2021, miejscowość Borowiec k. Warszawy.



**Rysunek 3.2.** Widok poletka testowego, na którym prowadzona była uprawa kukurydzy w sezonie wegetacyjnym 2022, miejscowość Borowiec k. Warszawy.



**Rysunek 3.3.** Widok poletka testowego, na którym prowadzona była uprawa kukurydzy w sezonie wegetacyjnym 2022 i 2024, miejscowość Borowiec k. Warszawy.

Kukurydza na poletkach testowych została posiana w odstępach między rzędami i między poszczególnymi roślinami w sposób analogiczny do siewu, jaki prowadzi się na rzeczywistych polach uprawnych. Wszystko po to, aby zasymulować w skali 1:1 warunki rzeczywistej uprawy kukurydzy. Roślinom zapewniono warunki wzrostu typowe dla pól uprawnych prowadzonych na terenie Polski.

Należy również podkreślić, iż na poletku testowym w szklarni zbierane były początkowo wyłącznie obrazy z użyciem kamery RGB. Zauważono, iż elementy konstrukcyjne szklarni zbytnio ingerują w początkowe wyniki działania wytrenowanych na ich podstawie sztucznych sieci neuronowych, więc pociągnęło to za sobą decyzję, iż dalsze badania będą kontynuowane wyłącznie na zewnętrznych poletkach testowych na otwartym polu, aby jeszcze lepiej zasymulować rzeczywiste warunki panujące na polach uprawnych. I tak, w sezonie wegetacyjnym 2021 na zewnętrznym poletku testowym wysiewana była kukurydza i pobierane w czasie jej wzrostu obrazy RGB i wielospektralne (z użyciem kamery wielospektralnej MicaSense Red Edge MX Dual), mające służyć do realizacji zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy. W sezonie wegetacyjnym 2022 ponowiono wysiew kukurydzy i w czasie jej wzrostu pobierano obrazy RGB i wielospektralne (z użyciem kamery wielospektralnej Red Edge) mające służyć do realizacji zadania związanego z detekcją i klasyfikacją etapów rozwoju kukurydzy. Z kolei w sezonie wegetacyjnym 2024 również ponowiono wysiew kukurydzy i w czasie jej wzrostu pobierano obrazy RGB i wielospektralne (tym razem z użyciem nowej kamery wielospektralnej MapIR Survey3 Red Green + NIR) mające służyć do realizacji zadania związanego z detekcją i klasyfikacją etapów rozwoju



kukurydzy oraz zadania związanego z detekcją i klasyfikacją poziomów nawodnienia kukurydzy.

W przypadku realizacji zadania związanego z detekcją i klasyfikacją kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami początkowo założono w tym celu, dzięki działaniom Działu Teledetekcji Sieci Badawczej Łukasiewicz – Instytutu Lotnictwa, poletko testowe znajdujące się w miejscowości Winniki k. Poznania. Inakulowane tam wybrane gatunki grzybów prawdopodobnie ze względu na występującą suszę nie rozwinęły się i nie wywołały śladów porażenia kukurydzy. W tej sytuacji do przeprowadzenia badań posłużył zewnętrzny zbiór danych. W zbiorze tym, obok innych roślin, można odnaleźć zbiór obrazów kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami (Helmintosporioza, Rdza Pospolita, Szara Plamistość, Gąsiennica Spodoptera Frugiperda).

### 3.2.2. Krótkie wprowadzenie dotyczące użytych kamer

W ramach opisywanych badań posłużono się zobrazowaniem RGB i wielospektralnym. Do pozyskiwania tychże zobrazowań posłużyła kamera RGB oraz dwa rodzaje kamer wielospektralnych. Kamera wielospektralna MicaSense RedEdge MX Dual posłużyła do pozyskania obrazów wielospektralnych w sezonach wegetacyjnych 2021 i 2022, zaś kamera wielospektralna MapIR Survey3 Red + GGreen + NIR posłużyła do pozyskiwania obrazów wielospektralnych w sezonie wegetacyjnym 2024.

Szczegółowe parametry techniczne użytych kamer znajdują się w załączniku B.

### 3.2.3. Dane RGB

Na przygotowanych wcześniej poletkach testowych prowadzony był proces pozyskiwania obrazów RGB w celu późniejszego wykrywania na nich obiektów w postaci kukurydzy oraz w celu klasyfikacji wykrytej kukurydzy pod względem faz rozwojowych, w których znajdują się poszczególne rośliny. Obrazy pozyskiwane były również w formie obrazów wielospektralnych, o czym będzie mowa w kolejnym podrozdziale.

Obrazy były pozyskiwane od momentu pojawienia się roślin ponad powierzchnią ziemi, czyli od etapu wykształcania się załążka pierwszego liścia w postaci zwiniętej struktury. Obrazy pozyskiwane były codziennie, aż do momentu wykształcenia maksymalnych

rozmiarów roślin. Zagwarantowało to pozyskanie danych z każdego etapu rozwoju poszczególnych roślin (szczególnie faz BBCH 10-19).

Obrazy RGB pozyskiwane były o różnych porach dnia oraz przy różnych warunkach pogodowych, aby zwielokrotnić różnorodność pozyskiwanego zbioru. Pozyskiwanie obrazów w szklarni wiązało się z rejestracją dodatkowych, nietypowych dla rzeczywistej uprawy elementów tła, co stanowiło istotną przeszkodę w symulacji rzeczywistych warunków polowych. Obrazy te zatem posłużyły jedynie do początkowych testów i nie trafiły do finalnego zbioru oznaczonych obrazów RGB. Obrazy pozyskiwane na zewnętrznych poletkach testowych były natomiast idealnym odzwierciedleniem warunków panujących na rzeczywistym polu uprawnym kukurydzy. Dodatkowo, na części poletka posiano typowe chwasty występujące w uprawie zbóż na terenie Polski, a na pozostałej części poletka udało się utrzymać uprawę względnie pozbawioną chwastów, co również zwielokrotniło różnorodność pozyskanego zbioru.

Ważna była również pozycja pobierania obrazów. Była ona dostosowana do pozycji kamery RGB zamontowanej na rzeczywistym prototypie robota polowego i gwarantowała pozyskiwanie obrazów z identycznej pozycji i pod tym samym kątem, co zobrazowania pobierane podczas rzeczywistej pracy robota w polu. Była to pozycja w odległości około 30 cm od rzędu roślin w pozycji prostopadle skierowanej do rzędu kukurydzy.

Poniżej przedstawiono przykładowe obrazy kukurydzy pozyskiwanej podczas pobierania obrazów na prowadzonych poletkach testowych. Na Rysunku 3.4. przedstawiono obrazy pozyskane na poletku testowym prowadzonym w szklarni. Kolejne obrazy przedstawione na Rysunku 3.5. to obrazy kukurydzy w jej początkowych fazach rozwojowych. Z kolei obrazy na Rysunku 3.6. przedstawiają kukurydzę w jej większych fazach rozwojowych, przy czym po lewej stronie widać kukurydzę pozbawioną chwastów, a po prawej kukurydzę z wyrastającymi pomiędzy nią chwastami. Rysunek 3.7. przedstawia natomiast obrazy kukurydzy wykonane wieczorem przy mniejszej dostępności światła.



**Rysunek 3.4.** Obraz kukurydzy wykonane w szklarni testowej.



**Rysunek 3.5.** Obrazy kukurydzy będącej w jej początkowych fazach rozwojowych.



**Rysunek 3.6.** Obrazy kukurydzy niezachwaszczonej i zachwaszczonej będącej w jej wyższych fazach rozwojowych.





**Rysunek 3.7.** Obraz kukurydzy wykonane wieczorem.

Przy wykorzystaniu obrazów RGB kukurydzy pozyskanych w sezonach wegetacyjnych 2021-2022 oraz w sezonie wegetacyjnym 2024 powstały dwa autorskie datasety dotyczące faz rozwojowych kukurydzy oraz 1 autorski dataset dotyczący poziomów nawodnienia kukurydzy, które w kolejnych krokach posłużyły do trenowania zestawu głębokich sztucznych sieci neuronowych. Prowadzone badania ewoluowały na przestrzeni realizacji doktoratu i konieczne było stworzenie dwóch datasetów RGB dla określania faz rozwojowych kukurydzy, ponieważ pierwszy z nich był referencyjny dla datasetu obrazów wielospektralnych pozyskiwanych przy użyciu kamery MicaSense RedEdge MX Dual w sezonach wegetacyjnych 2021-2022 i przedstawiał te same rośliny, których zobrazowania wielospektralne pozyskiwano w tym samym czasie. Drugi dataset RGB dotyczący faz rozwojowych kukurydzy z obrazami pozyskiwanymi w sezonie wegetacyjnym 2024 był referencyjny dla datasetu obrazów wielospektralnych pozyskiwanych przy użyciu kamery wielospektralnej MapIR Survey3 Red + Green + NIR dla tych samych roślin i w tym samym czasie, co zbiór obrazów RGB. Trzeci dataset RGB dotyczący poziomów nawodnienia kukurydzy ze obrazami pozyskiwanymi w sezonie wegetacyjnym 2024 był referencyjny dla datasetu obrazów wielospektralnych pozyskiwanych przy użyciu kamery wielospektralnej MapIR Survey3 Red + Green + NIR dla tych samych roślin i w tym samym czasie, co zbiór obrazów RGB. Zachowanie pełnej spójności fotografowanych obiektów i pozyskiwanie zobrazowań RGB i poszczególnych zobrazowań wielospektralnych w tym samym czasie gwarantowało zachowanie tych samych warunków pozyskiwania danych, co było kluczowe w momencie porównywania uzyskanych wyników w zależności od zastosowanego zobrazowania, RGB oraz wielospektralnego.



Podsumowując, w ramach autorskich zbiorów danych RGB powstały następujące zbiory: zbiór obrazów RGB odnoszący się do faz rozwojowych kukurydzy z sezonów wegetacyjnych 2021-2022, zbiór obrazów RGB odnoszący się do faz rozwojowych kukurydzy z sezonu wegetacyjnego 2024 oraz zbiór obrazów RGB odnoszący się do różnych poziomów nawodnienia kukurydzy w sezonie wegetacyjnym 2024.

Poszczególne etapy przygotowywania datasetów stanowiły:

- przegląd zebranych obrazów i odrzucenie obrazów nienadających się do dalszej analizy, czyli np. prześwietlonych, pozbawionych elementów w postaci kukurydzy itp.,
- określenie klas obrazów do późniejszego oznaczania - jako etap kluczowy z punktu widzenia całego procesu,
- wybór narzędzia do oznaczania obrazów,
- przeprowadzenie procesu oznaczania obrazów oraz dobór optymalnego sposobu oznaczania obiektów na obrazach,
- przygotowanie tzw. outputu, czyli plików wyjściowych (w przypadku oznaczania metodą polygon) lub folderów (w przypadku oznaczania metodą ground truth) po zakończeniu procesu oznaczania obrazów.

Poniżej zamieszczono przykłady obrazów RGB dla kukurydzy o niskim poziomie nawodnienia oraz dla kukurydzy o wysokim poziomie nawodnienia.



**Rysunek 3.8.** Obrazy kukurydzy charakteryzującej się niskim (po lewej) oraz wysokim poziomem nawodnienia (po prawej).

W podrozdziale 3.2.6. „Proces oznaczania obrazów” zostanie przybliżony szczegółowo proces oznaczania zarówno obrazów RGB jak i wielospektralnych oraz charakterystyka powstałych datasetów pod względem liczności obrazów i obiektów należących do poszczególnych klas.

Podczas przygotowań do realizacji tego zadania na polach testowych prowadzonych przez Sieć Badawczą Łukasiewicz – Instytut Lotnictwa przeprowadzono proces inakulacji grzybów, które miały stanowić patogeny porażające kukurydzę. Na skutek niesprzyjających temu procesowi warunków pogodowych (zbyt mało deszczu) inakulowane grzyby nie rozwinęły się. W związku z tym rozpoczęto poszukiwania alternatywnej drogi zebrania obrazów kukurydzy porażonej patogenami. Pomocy stał się zewnętrzny zbiór danych.

W zbiorze znajdują się następujące klasy obrazów:

- Gąsiennica *Spodoptera frugiperda*
- Helminosporioza - grzyby *Setosphaeria turcica*
- Rdza pospolita - grzyby z rzędu rdzowców (*Pucciniales*)
- Szara plamistość liści - grzyby *Aureobasidium zeae*
- Zdrowa kukurydza

Poniżej zamieszczono kilka przykładowych obrazów pochodzących z analizowanego datasetu.

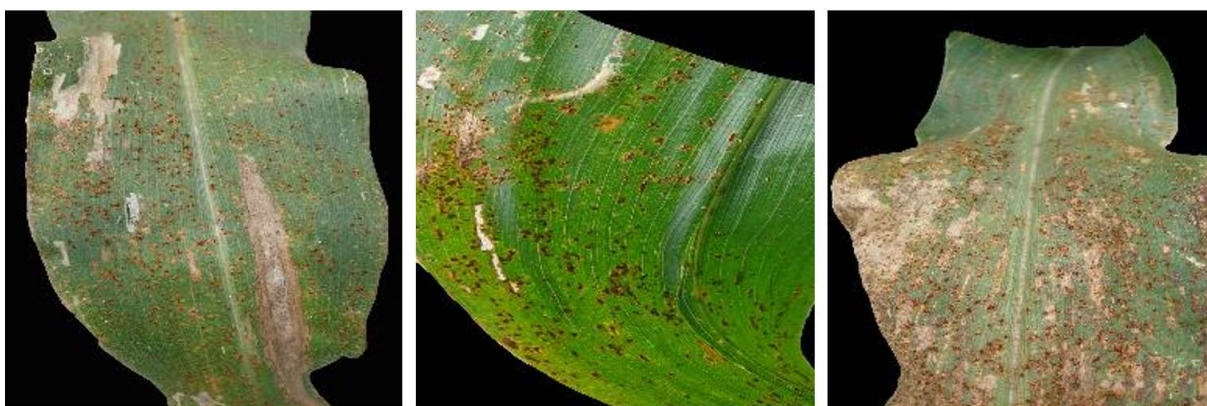


**Rysunek 3.9.** Przykładowe obrazy pochodzące z analizowanego datasetu - gąsiennica *Spodoptera frugiperda*





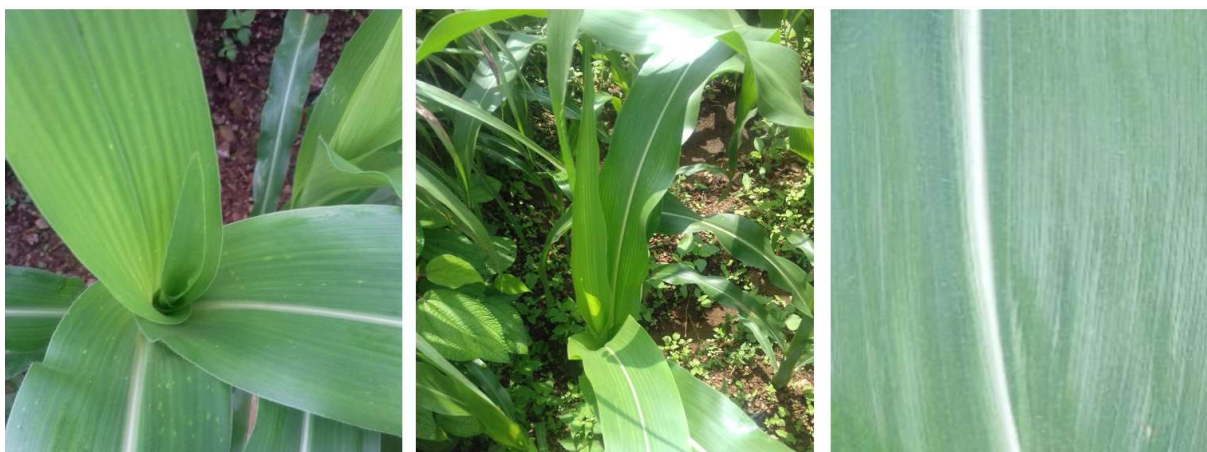
**Rysunek 3.10.** Przykładowe obrazy pochodzące z analizowanego datasetu - Helmintosporioza - grzyby *Setosphaeria turcica*



**Rysunek 3.11.** Przykładowe obrazy pochodzące z analizowanego datasetu - rdza pospolita - grzyby z rzędu rdzowców (Pucciniales)



**Rysunek 3.12.** Przykładowe obrazy pochodzące z analizowanego datasetu - szara plamistość liści - grzyby *Aureobasidium zeae*



**Rysunek 3.13.** Przykładowe obrazy pochodzące z analizowanego datasetu - zdrowa kukurydza

Pobrany dataset został podzielony na foldery odpowiadające poszczególnym klasom obrazów, tzn.: Gąsiennica *Spodoptera frugiperda*, Helminosporioza, Rdza pospolita, Szara plamistość liści ora zdrowa kukurydza. W ten sposób został oznaczony z użyciem etykiety ground truth, która została im przypisana w momencie podziału na odpowiednie foldery. W kolejnym kroku podzielono poszczególne zbiory obrazów na zbiory treningowe i walidacyjne oraz rozpoczęto testowanie wybranych algorytmów uczenia głębokiego.

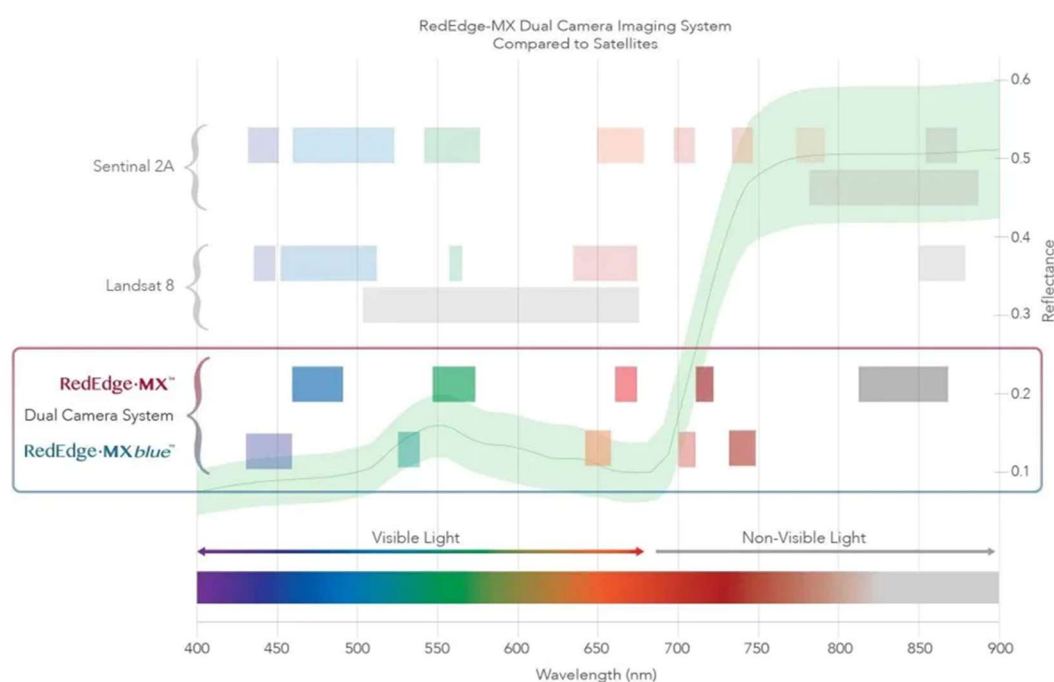
#### 3.2.4. Dane wielospektralne z kamery RedEdge

Obrazy wielospektralne kukurydzy pozyskiwane na przygotowanych do tego wcześniej poletkach testowych, tożsamyh z poletkami, z których pozyskiwane były obrazy RGB, pobierane były z identyczną częstotliwością, co obrazy RGB. Również w tym przypadku zapewniono maksymalną różnorodność obrazów pod względem ilości roślin, obecności wśród nich chwastów lub ich braku oraz pory dnia w momencie pozyskiwania obrazów.

Obrazy wielospektralne pozyskiwane były za pomocą kamery wielospektralnej MicaSense RedEdge MX DUAL, która potrafi pobierać zobrazowania w 10 kanałach spektralnych charakteryzujących się następującymi parametrami:

- 1 kanał spektralny – pasmo Blue (Niebieski), środek długości fali 475, zakres pasma 32
- 2 kanał spektralny – pasmo Green (Zielony), środek długości fali 560, zakres pasma 27
- 3 kanał spektralny – pasmo Red (Czerwony), środek długości fali 668, zakres pasma 14

- 4 kanał spektralny – pasmo Near Infrared (Bliska Podczerwień), środek długości fali 842, zakres pasma 57
- 5 kanał spektralny – pasmo Red Edge („Czerwona krawędź” 1), środek długości fali 717, zakres pasma 12
- 6 kanał spektralny – pasmo Coastal Blue (Niebieski morski), środek długości fali 444, zakres pasma 28
- 7 kanał spektralny – pasmo Green (Zielony), środek długości fali 531, zakres pasma 14
- 8 kanał spektralny – pasmo Red (Czerwony), środek długości fali 650, zakres pasma 16
- 9 kanał spektralny – pasmo Red Edge („Czerwona krawędź” 2), środek długości fali 705, zakres pasma 10
- 10 kanał spektralny – pasmo Blue („Czerwona krawędź” 3), środek długości fali 740, zakres pasma 18



**Rysunek 3.14.** Parametry kamery MicaSense RedEdge MX DUAL.

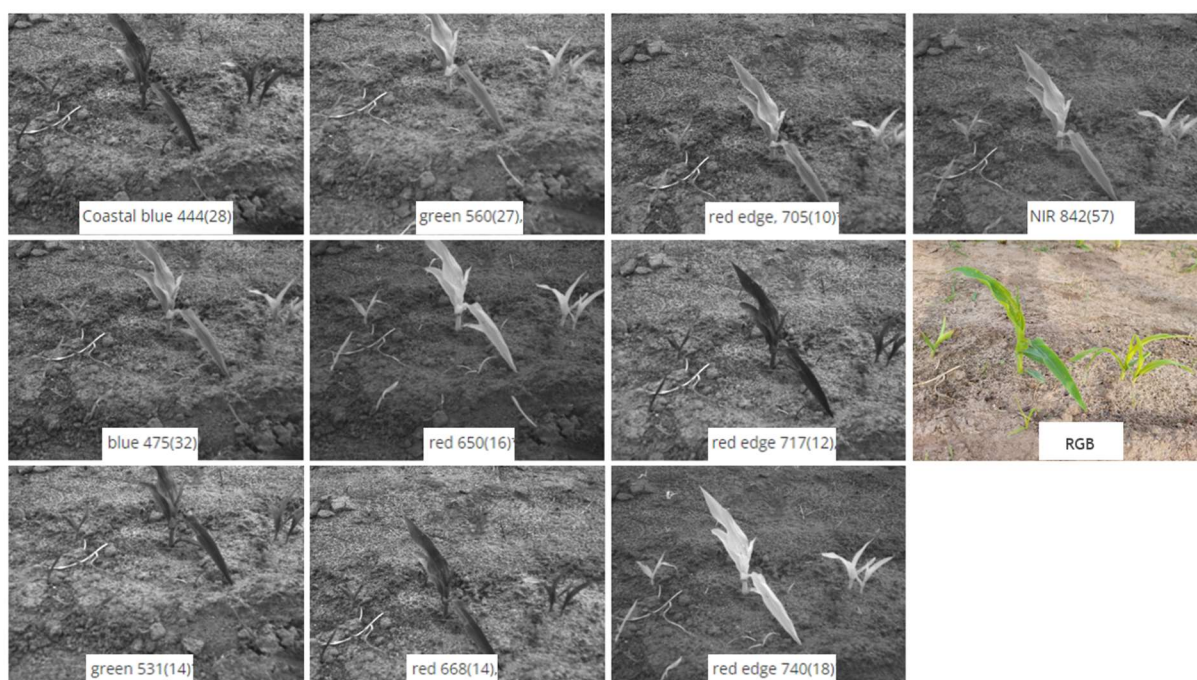
Na Rysunku 3.14. przedstawiono parametry kamery MicaSense RedEdge MX DUAL wraz z odniesieniem do wizualnego przedstawienia poszczególnych jej pasm obrazowania. Na Rysunku 3.15. przedstawiono z kolei wygląd kamery. Dokumentacja kamery znajduje się w załączniku B.





**Rysunek 3.15.** Wygląd kamery MicaSense RedEdge-MX DUAL.

Poniżej zamieszczono przykłady obrazów wielospektralnych oraz RGB (dla porównania) sporządzonych dla tej samej rośliny. Obrazy z kamery MicaSense RedEdge-MX DUAL są generowane w formacie TIFF.



**Rysunek 3.16.** Przykładowe obrazy kukurydzy zarejestrowane przez poszczególne obiektywy kamery wielospektralnej MicaSense RedEdge-MX DUAL oraz przykładowy obraz tych samych roślin zarejestrowany przez kamerę RGB.



Na podstawie pozyskanych obrazów kukurydzy, kolejnym krokiem w realizacji zamierzonych zadań było przygotowanie datasetów, które w kolejnych krokach posłużyły do trenowania zestawu głębokich sztucznych sieci neuronowych. W przypadku obrazów wielospektralnych zadaniami, do których realizacji posłużyły te zobrazowania była detekcja i klasyfikacja faz rozwoju kukurydzy oraz detekcja i klasyfikacja poziomów nawodnienia kukurydzy, z tymże dla każdego z tych zadań ze względu na ich specyfikę utworzono oddzielne zbiory danych. Poszczególne etapy przygotowywania datasetów stanowiły:

- Przegląd zebranych obrazów i odrzucenie obrazów nienadających się do dalszej analizy
- Określenie klas obrazów do późniejszego oznaczania
- Wybór narzędzia do oznaczania obrazów
- Przeprowadzenie procesu oznaczania obrazów oraz dobór optymalnego sposobu oznaczania obiektów na obrazach
- Przygotowanie tzw. outputu, czyli pliku wyjściowego po zakończeniu procesu oznaczania obrazów.

W podrozdziale 3.2.6. „*Proces oznaczania obrazów*” zostanie przybliżony szczegółowo również proces oznaczania obrazów wielospektralnych z kamery MicaSense Red Edge MX-Dual oraz charakterystyka powstałych datasetów pod względem liczności obrazów i obiektów należących do poszczególnych klas.

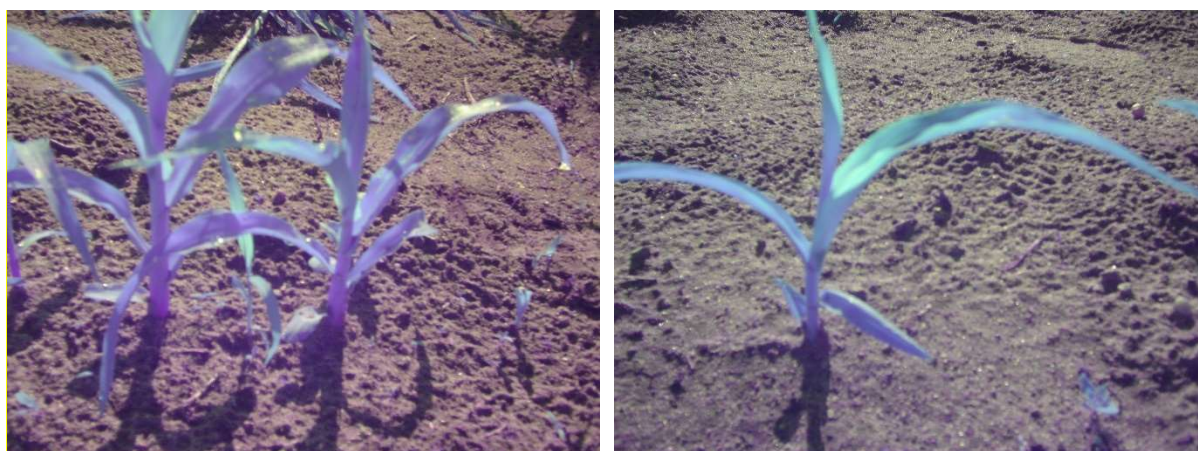
### 3.2.5. Dane wielospektralne z kamery MapIR

Obrazy wielospektralne kukurydzy wykonywane za pomocą kamery MapIR Survey3 Red + Green + NIR na przygotowanym do tego wcześniej poletku testowym, tożsamym z poletkiem, z którego pozyskiwane były obrazy RGB w sezonie wegetacyjnym 2024, pobierane były z identyczną częstotliwością, co obrazy RGB. Również w tym przypadku zapewniono maksymalną różnorodność obrazów pod względem ilości roślin, obecności wśród nich chwastów lub ich braku oraz pory dnia pozyskiwania poszczególnych obrazów. Na Rysunku 3.17. przedstawiono z kolei wygląd kamery.



**Rysunek 3.17.** Wygląd kamery MapIR.

Poniżej zamieszczono przykłady obrazów wielospektralnych pozyskiwanych za pomocą kamery wielospektralnej MapIR. Obrazy te są generowane w formacie JPG.



**Rysunek 3.18.** Obrazy pozyskane za pomocą kamery MapIR.

Na podstawie pozyskanych w sezonie wegetacyjnym 2024 obrazów kukurydzy, kolejnym krokiem w realizacji zamierzonych zadań było przygotowanie datasetu, który w kolejnych krokach posłużył do trenowania zestawu głębokich sztucznych sieci neuronowych. W przypadku obrazów wielospektralnych z kamery MapIR zadaniami, do których realizacji posłużyły te zobrażenia była detekcja i klasyfikacja faz rozwoju kukurydzy oraz detekcja i klasyfikacja poziomów nawodnienia kukurydzy, z tymże dla każdego z tych zadań ze względu

na ich specyfikę utworzono oddzielne zbiory danych. Poszczególne etapy przygotowywania datasetów stanowiły:

- Przegląd zebranych obrazów i odrzucenie obrazów nienadających się do dalszej analizy
- Określenie klas obrazów do późniejszego oznaczania
- Wybór narzędzia do oznaczania obrazów
- Przeprowadzenie procesu oznaczania obrazów oraz dobór optymalnego sposobu oznaczania obiektów na obrazach
- Przygotowanie tzw. outputu, czyli pliku wyjściowego po zakończeniu procesu oznaczania obrazów.

W kolejnej części niniejszej rozprawy zostanie przybliżony szczegółowo proces oznaczania również obrazów wielospektralnych z kamery MapIR oraz charakterystyka powstałych datasetów pod względem liczności obrazów i obiektów należących do poszczególnych klas.

### 3.2.6. Proces oznaczania obrazów

Proces oznaczania obrazów jest jednym z kluczowych etapów w procesie przygotowywania zbiorów oznaczonych obrazów służących następnie do realizacji procesu trenowania sztucznych sieci neuronowych i tworzenia na tej podstawie modeli zdolnych do realizowania określonych zadań.

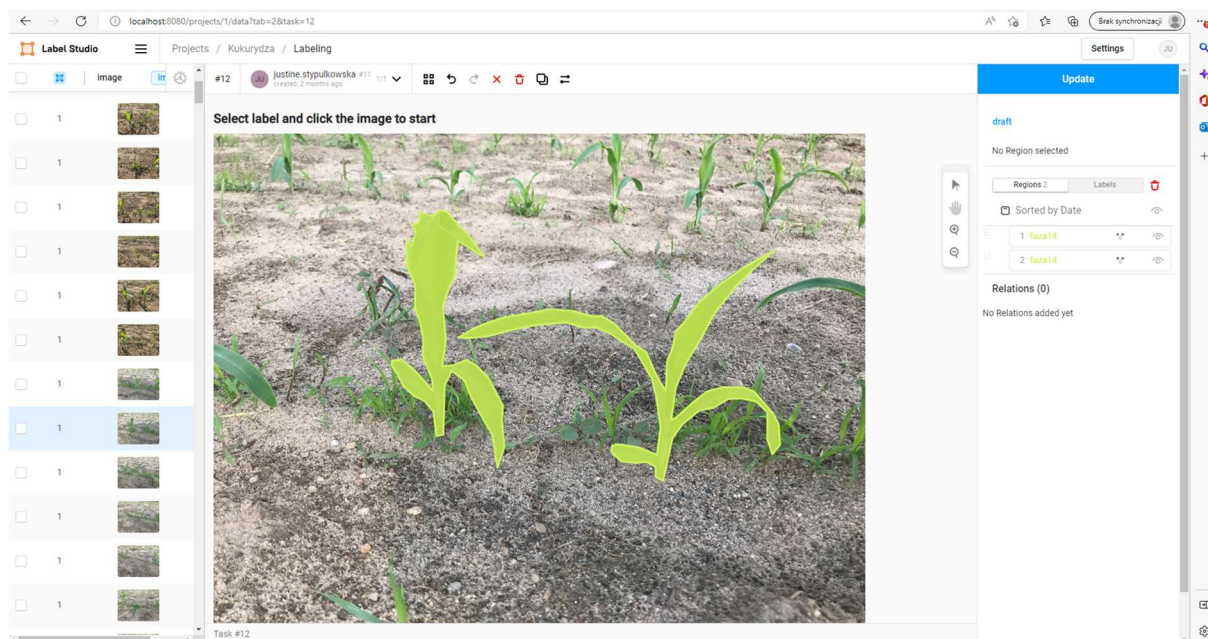
Pierwszym punktem w procesie przygotowywania obrazów do procesu ich oznaczania był przegląd zebranych obrazów (mowa tutaj zarówno o obrazach RGB jak i wielospektralnych, zarówno z kamery MicaSense RedEdge MX-Dual jak i z kamery MapIR) oraz odrzucenie tych, które nie wniosą ze sobą cennych informacji do przeprowadzenia późniejszego etapu trenowania głębokich sztucznych sieci neuronowych pod kątem wykrywania kukurydzy i klasyfikacji jej faz rozwojowych oraz klasyfikacji jej poziomów nawodnienia. Obrazy odrzucane jeszcze przed procesem oznaczania to najczęściej obrazy prześwietlone, rozmyte, zbyt przysłonięte o elementy nie będące kukurydzą oraz takie, na których nie zarejestrowano kukurydzy.

Po zakończonym etapie przeglądania pozyskanych obrazów RGB i wielospektralnych z poszczególnych kamer, należało określić klasy, którym przypisane zostaną poszczególne zarejestrowane rośliny podczas procesu oznaczania obiektów na obrazach.

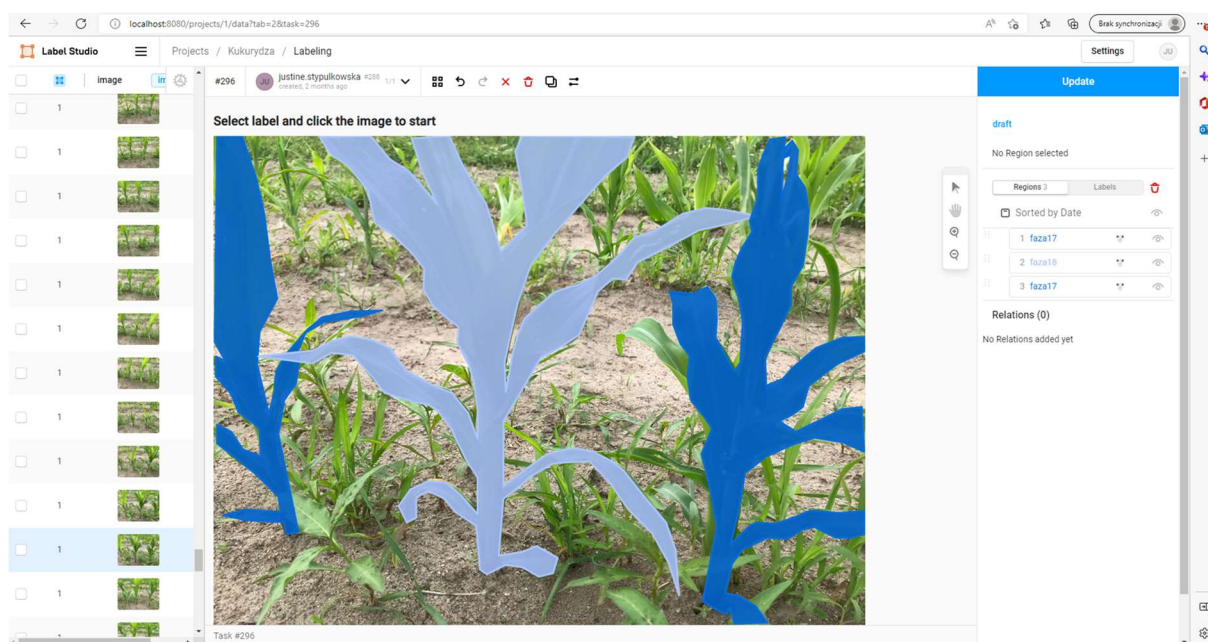
W przypadku zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy posłużono się tutaj międzynarodową skalą rozwoju roślin BBCH oraz jej uszczegółowieniem dla kukurydzy. Szczegóły tej metryki opisano w rozdziale 2 poświęconym przeglądowi literaturowemu w podrozdziale 2.6.1. *Skala BBCH i jej zastosowanie w analizie faz rozwojowych roślin*. W przypadku tego zadania klasami zostały poszczególne wartości faz rozwojowych BBCH: *faza10, faza11, faza12, faza13, faza14, faza15, faza16, faza17, faza18 i faza19*.

W przypadku zadania związanego z detekcją i klasyfikacją poziomów nawodnienia kukurydzy klasami zostały zaś określenia odnoszące się do poziomów nawodnienia: *niski poziom nawodnienia, średni poziom nawodnienia i wysoki poziom nawodnienia*.

W przypadku zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy po etapie określania klas obiektów kolejnym kluczowym krokiem był wybór narzędzia do oznaczania obiektów na obrazach. Początkowy wybór padł na narzędzie VGG Annotator, lecz po etapie testów kilku innych narzędzi zdecydowanie najbardziej optymalne stało się środowisko Label Studio, które pozwala na wybór różnych technik oznaczania oraz umożliwia tworzenie różnych typów plików wyjściowych. Tuż po dokonaniu wyboru narzędzia do oznaczania kolejnym etapem był proces dostosowywania środowiska do realizowanego procesu oznaczania obrazów. W tym celu ustawiono w Label Studio następujące klasy oznaczanych obiektów: *faza10, faza11, faza12, faza13, faza14, faza15, faza16, faza17, faza18* oraz *faza19* i przypisano im różne kolory, aby ułatwić proces oznaczania. Dodatkowym, szczególnie ważnym parametrem oznaczania był wybór sposobu oznaczania obiektów. W tym przypadku standardowe oznaczanie prostokątem mogłoby się okazać zbyt mało dokładne, więc dokonano wyboru oznaczania wielokątem, tzw. polygon, który jest metodą wystarczająco precyzyjną. Po dokonaniu tychże wyborów rozpoczęto właściwy proces oznaczania obiektów na obrazach. Na rysunkach poniżej przedstawiono zrzuty ekranu z widoku narzędzia Label Studio podczas dokonywania procesu oznaczania.

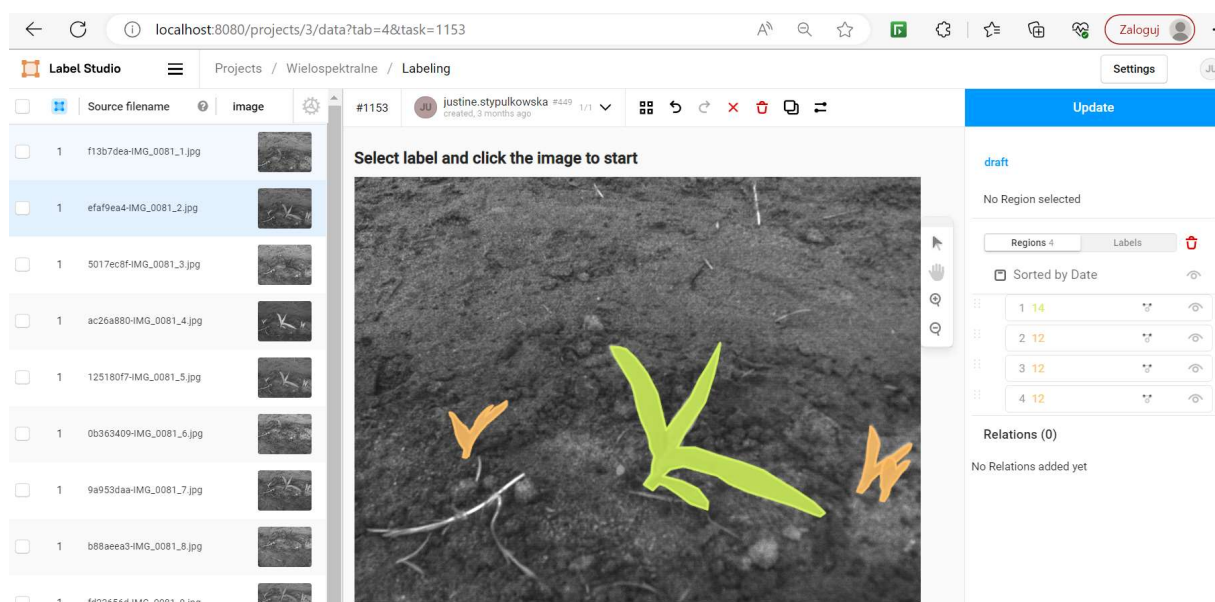


**Rysunek 3.19.** Widok środowiska Label Studio podczas procesu oznaczania. Na ekranie widać rośliny będące w fazie 14 oznaczone metodą wielokąta (ang. „polygon”).

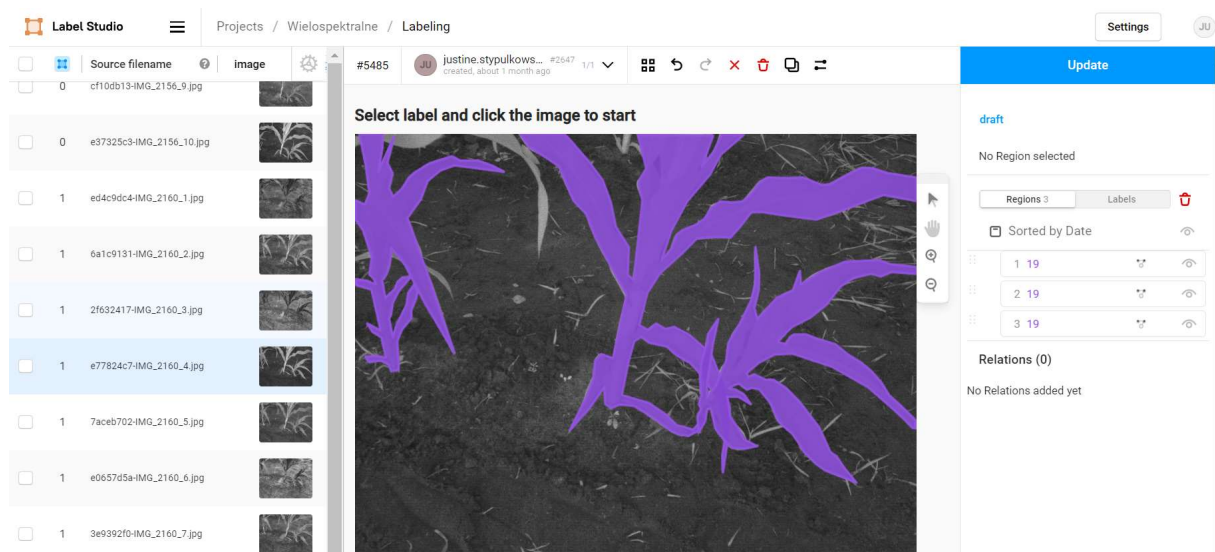


**Rysunek 3.20.** Widok środowiska Label Studio podczas procesu oznaczania. Na ekranie widać rośliny będące w fazie 17 i 18 oznaczone metodą wielokąta (ang. „polygon”).





**Rysunek 3.21.** Widok środowiska Label Studio podczas procesu oznaczania. Na ekranie widać rośliny będące w fazie 14 oznaczone metodą wielokąta (ang. „polygon”).



**Rysunek 3.22.** Widok środowiska Label Studio podczas procesu oznaczania. Na ekranie widać rośliny będące w fazie 19 oznaczone metodą wielokąta (ang. „polygon”).

W zbiorze obrazów RGB znajduje się 700 oznaczonych obrazów kukurydzy o rozdzielczości 12 MP i zapisanych w formacie JPG. Z kolei w zbiorze obrazów wielospektralnych znajduje się po 556 oznaczonych obrazów dla każdego kanału spektralnego, czyli łącznie 5560 obrazów dla 10 kanałów spektralnych o następujących parametrach: coastal blue 444 (28), blue 475 (32), green 531 (14), green 560 (27), red 650 (16), red 668 (14), red edge 705 (10), red edge 717 (12), red edge 740 (18), NIR 842



(57), gdzie główna liczba oznacza środkową częstotliwość danego pasma, a wartość w nawiasie oznacza szerokość pasma. Obrazy te charakteryzują się rozdzielczością 5 MP i były zapisywane w formacie TIFF, a następnie konwertowane do formatu JPG ze względu na wymagania narzędzia do oznaczania. Obrazy były przez nas pozyskiwane przy zróżnicowanych warunkach pogodowych, różnym nasłonecznieniu oraz o różnej porze dnia, aby zwielokrotnić różnorodność zbieranych zbiorów. Podczas oznaczania starano się, aby zachować względnie równą licznosc obiektów oznaczonych w poszczególnych klasach.

Na każdym obrazie zostało zarejestrowanych od 1 do 4 roślin kukurydzy, co przełożyło się na łączną ilość 1646 oznaczonych obiektów w zbiorze obrazów RGB, łączną ilość 9110 oznaczonych obiektów w zbiorze obrazów wielospektralnych z kamery Red Edge (czyli 911 na każdy kanał spektralny) oraz łączną ilość 1000 obiektów w zbiorze obrazów z kamery wielospektralnej MapIR. Dodatkowym celem było to, aby ilość obiektów danej klasy, czyli ilość roślin będących w danej fazie rozwojowej, była zrównoważona w każdym ze zbiorów i osiągnięta została następującą licznosc obiektów w poszczególnych klasach:

- w zbiorze obrazów RGB: faza 10: 84 obiekty, faza 11: 141 obiektów, faza 12: 137 obiektów, faza 13: 216 obiektów, faza 14: 366 obiektów, faza 15: 143 obiekty, faza 16: 110 obiektów, faza 17: 155 obiektów, faza 18: 114 obiektów, faza 19: 180 obiektów,
- w zbiorze obrazów wielospektralnych z kamery Red Edge dla każdego pojedynczego kanału spektralnego: faza 10: 82 obiekty, faza 11: 86 obiektów, faza 12: 163 obiekty, faza 13: 82 obiekty, faza 14: 82 obiekty, faza 15: 83 obiekty, faza 16: 84 obiekty, faza 17: 83 obiekty, faza 18: 84 obiekty, faza 19: 82 obiekty
- w zbiorze obrazów wielospektralnych z kamery MapIR: faza 10: 100 obiektów, faza 11: 100 obiektów, faza 12: 100 obiektów, faza 13: 100 obiektów, faza 14: 100 obiektów, faza 15: 100 obiektów, faza 16: 100 obiektów, faza 17: 100 obiektów, faza 18: 100 obiektów, faza 19: 100 obiektów.

Po zakończeniu procesu oznaczania obrazów należało przejść do kolejnego etapu, czyli przygotowania tzw. outputu, czyli pliku wyjściowego z wynikami wszystkich pojedynczych oznaczeń. Środowisko Label Studio oferuje kilka typów formatu plików wyjściowych (jak pokazano na rysunku 3.23), a dla opisywanego zadania wybrano format .COCO, który jest optymalny dla prowadzenia procesu trenowania głębokich sieci neuronowych pod kątem wykrywania obiektów na obrazach. Po wygenerowaniu pliku w formacie .COCO na wyjściu tworzony jest plik w formacie .JSON wraz z integralnym folderem, w którym znajdują się wszystkie obrazy, na których dokonano oznaczenia.

Export data ✕

You can export dataset in one of the following formats:

☐ JSON

List of items in raw JSON format stored in one JSON file. Use to export both the data and the annotations for a dataset. It's Label Studio Common Format

☐ JSON-MIN

List of items where only "from\_name", "to\_name" values from the raw JSON format are exported. Use to export only the annotations for a dataset.

☐ CSV

Results are stored as comma-separated values with the column names specified by the values of the "from\_name" and "to\_name" fields.

☐ TSV

Results are stored in tab-separated tabular file with column names specified by "from\_name" "to\_name" values

☒ COCO

image segmentation

object detection

Popular machine learning format used by the COCO dataset for object detection and image segmentation tasks with polygons and rectangles.

☐ CONLL2003

sequence labeling

text tagging

named entity recognition

Popular format used for the CoNLL-2003 named entity recognition challenge.

Export

**Rysunek 3.23.** Widok panelu z wyborem formatu wyjściowego w środowisku Label Studio.

Po wygenerowaniu plików wyjściowych w formacie COCO JSON. Dane w konkretnych zbiorach zostały podzielone na zbiory treningowe, które stanowiły około 70% poszczególnych zbiorów oraz zbiory walidacyjne (tożsame ze zbiorami testowymi) stanowiące około 30% poszczególnych zbiorów. Podział został dokonany w sposób losowy, a stratyfikacja zapewniła równomierny podział obrazów z każdej fazy BBCH w każdym z tychże zbiorów.

W przypadku zadania związanego z detekcją i klasyfikacją poziomów nawodnienia kukurydzy użycie narzędzia do oznaczania stało się zbędne, gdyż zbiory należące do klas: *niski poziom nawodnienia*, *średni poziom nawodnienia* i *wysoki poziom nawodnienia* oznaczone zostały z użyciem metody *ground truth*. Zgodnie z ideą metody poszczególne zbiory obrazów podzielono na 3 podfoldery, w których zostały umieszczone obrazy przypisane poszczególnym klasom. Następnie podczas tworzenia zbiorów treningowych i walidacyjnych (testowych) obrazy z poszczególnych klas zostały przydzielone w 80% do zbioru treningowego i w 20% do zbioru walidacyjnego (testowego). Struktura folderów z obrazami oznaczonymi metodą *ground truth* i podzielonymi na zbiór treningowy i walidacyjny (testowy) wygląda następująco:

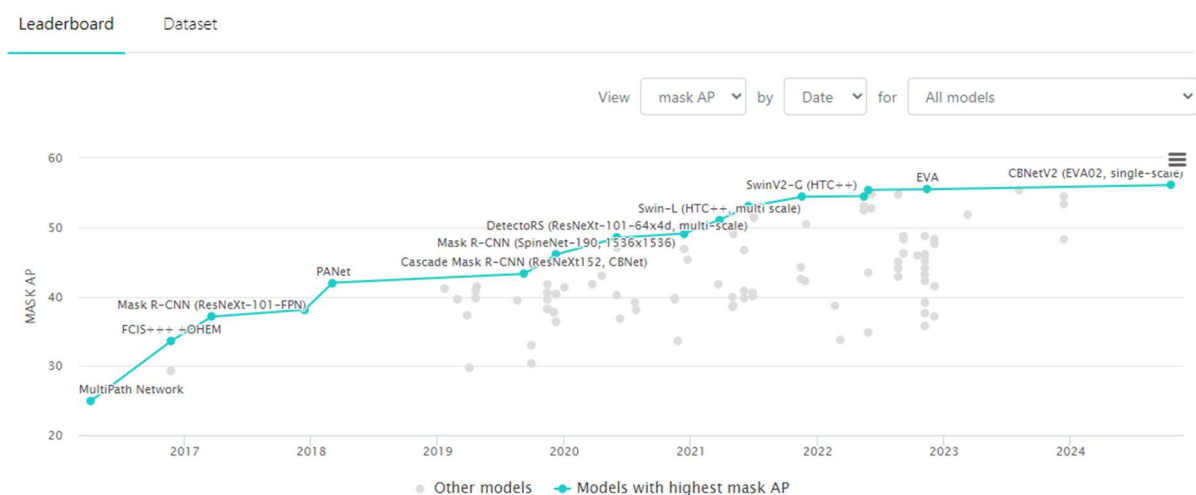
- **train** - zbiór treningowy
  - **niski** – niski poziom nawodnienia
  - **średni** – średni poziom nawodnienia
  - **wysoki** – wysoki poziom nawodnienia
- **val** - zbiór walidacyjny (testowy)
  - **niski** – niski poziom nawodnienia
  - **średni** – średni poziom nawodnienia
  - **wysoki** – wysoki poziom nawodnienia

### 3.3. Implementacja architektur głębokich sieci neuronowych

Głębokie sztuczne sieci neuronowe stanowią obecnie podstawę nowoczesnych metod analizy obrazów i wykrywania obiektów. Dzięki zdolności do automatycznego ekstraktowania cech i uczenia się reprezentacji, sieci te stały się kluczowym elementem w zadaniach takich jak klasyfikacja obrazów, segmentacja semantyczna, segmentacja instancyjna oraz detekcja obiektów [LeCun i in., 2015].

W ramach niniejszej rozprawy doktorskiej zastosowano i zaadaptowano różne architektury głębokich sieci neuronowych w celu realizacji zadań związanych z oceną kondycji kukurydzy, w tym klasyfikacji faz rozwojowych, poziomów nawodnienia oraz identyfikacji kukurydzy zdrowej i porażonej patogenami. Szczególną uwagę poświęcono modelom HTC (Hybrid Task Cascade) [Chen i in., 2018], YOLOX [Redmon, Farhadi, 2018], Mask2Former [Chen i in., 2021], ConvNeXt small [Tan, Le, 2019], ResNet50 [He i in., 2016], EfficientNet\_B3 [Tan, Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] oraz Swin Transformer [Liu i in., 2021]. Każda z tych architektur została dostosowana do specyficznych wymagań danego zadania, co pozwoliło na osiągnięcie wysokiej precyzji i wydajności.

# Instance Segmentation on COCO test-dev



**Rysunek 3.24.** Wykrywanie obiektów – state of the art [paperswithcode.com, 2024].

## 3.3.1. Implementacja modeli HTC\_x101, HTC\_r101, HTC\_r50

W ramach niniejszej pracy zastosowano trzy modele oparte na architekturze Hybrid Task Cascade (HTC): HTC\_x101 (z ResNeXt101 jako backbone), HTC\_r101 (z ResNet101 jako backbone) i HTC\_r50 (z ResNet50 jako backbone). Architektura HTC została wybrana ze względu na jej zdolność do jednoczesnej realizacji zadań segmentacji semantycznej, instancyjnej oraz detekcji obiektów [Chen i in., 2019]. Jest to szczególnie istotne w kontekście analizy obrazów roślin, gdzie konieczna jest dokładna ocena różnych etapów rozwoju oraz identyfikacja kluczowych cech morfologicznych.

Implementacja modeli HTC\_x101, HTC\_r101 i HTC\_r50 została przeprowadzona przy użyciu biblioteki PyTorch, z wykorzystaniem funkcji i modułów dostępnych w frameworku MMDetection [Chen i in., 2019]. Kluczowym elementem implementacji było dostosowanie backbone'ów, tj. ResNeXt101, ResNet101 oraz ResNet50, które zapewniają różne poziomy głębokości i złożoności modeli, co przekłada się na różnorodne możliwości w zakresie ekstrakcji cech z obrazów [Xie i in., 2017], [He i in., 2016]. Sieci te zostały skonfigurowane z Feature Pyramid Network (FPN) jako warstwą łączącą, co pozwala na efektywne wykorzystanie informacji pochodzących z różnych poziomów głębokości sieci [Lin i in., 2017].

Backbone ResNeXt101, zastosowany w modelu HTC\_x101, wprowadza mechanizm grupowych konwolucji, co pozwala na efektywne przetwarzanie danych bez znacznego zwiększania liczby parametrów. W implementacji tego modelu skonfigurowano 32 grupy konwolucyjne, co pozwoliło na uzyskanie równowagi między wydajnością obliczeniową a dokładnością modelu [Xie i in., 2017].

Każdy z modeli HTC został przystosowany do pracy z dziesięcioma klasami, które odpowiadają różnym fazom rozwoju kukurydzy. Proces implementacji obejmował także konfigurację warstw odpowiedzialnych za przetwarzanie regionów zainteresowania (ROI) oraz generowanie masek segmentacyjnych, co pozwoliło na precyzyjną segmentację i detekcję obiektów na obrazach [He i in., 2016], [Chen i in., 2019].

### 3.3.2. Implementacja Mask2Former

Mask2Former to zaawansowana architektura sieci neuronowej, która łączy zalety konwolucyjnych sieci neuronowych z mechanizmami Transformera, co umożliwia precyzyjną segmentację obiektów na obrazach [Chen i in., 2021]. Model ten został zaprojektowany z myślą o równoczesnym przetwarzaniu globalnych oraz lokalnych cech obrazów, co jest kluczowe w złożonych zadaniach związanych z analizą roślin.

Implementacja Mask2Former została zrealizowana z wykorzystaniem ResNet50 jako backbone, który dostarcza bogatych reprezentacji obrazów dzięki swoim 50 warstwom [He i in., 2016]. Sieć ta jest w stanie uchwycić zarówno drobne szczegóły lokalne, jak i globalną strukturę obrazu, co jest kluczowe dla precyzyjnej segmentacji różnych faz rozwoju kukurydzy.

W modelu Mask2Former centralnym elementem jest Panoptic Head, który integruje informacje pochodzące z różnych poziomów sieci ResNet50 i przetwarza je przy użyciu mechanizmu Transformera [Chen i in., 2021]. Mask2FormerHead, będący integralną częścią Panoptic Head, umożliwia łączenie wielopoziomowych reprezentacji obrazów oraz ich wykorzystanie do prognozowania masek obiektów. Implementacja obejmowała także modyfikacje liczby zapytań w mechanizmie Transformera, co pozwoliło na zmniejszenie zapotrzebowania na zasoby obliczeniowe przy jednoczesnym zachowaniu wysokiej precyzji segmentacji [Chen i in., 2021].

### 3.3.3. Implementacja modeli ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer, Swin Transformer

W kontekście realizacji zadań związanych z klasyfikacją poziomów nawodnienia kukurydzy oraz oceny zdrowotności roślin, w niniejszej pracy zaimplementowano i przetestowano pięć zaawansowanych architektur sieci neuronowych: ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer. Każdy z tych modeli został dostosowany do specyficznych wymagań badania, uwzględniając charakterystykę danych obrazowych oraz cele analizy.

ConvNeXt small jest nowoczesną architekturą głębokiej sieci neuronowej, która łączy tradycyjne podejście do konwolucyjnych sieci neuronowych z nowymi technikami przetwarzania obrazu, inspirowanymi architekturami transformers. Model ten został zaprojektowany, aby zmaksymalizować wydajność obliczeniową i jednocześnie zwiększyć precyzję w zadaniach analizy obrazów [Liu i in., 2022].

Implementacja modelu ConvNeXt small w niniejszej pracy była realizowana z użyciem biblioteki timm, która umożliwia łatwe zarządzanie różnymi konfiguracjami modeli. ConvNeXt small został skonfigurowany do klasyfikacji trzech klas związanych z poziomami nawodnienia kukurydzy, przy czym kluczowe znaczenie miały techniki augmentacji, takie jak rotacje, odbicia lustrzane, zmiany jasności oraz kontrastu. Te techniki augmentacyjne pozwoliły na poprawę zdolności generalizacyjnych modelu, co było kluczowe dla uzyskania stabilnych i precyzyjnych wyników w zróżnicowanych warunkach środowiskowych [Tan, Le, 2019].

ConvNeXt small wykorzystuje zaawansowane operacje konwolucyjne, które są odpowiednio skalowane, aby maksymalnie wykorzystać dostępne zasoby obliczeniowe. Model ten został wyposażony w 384 jednostki konwolucyjne na poziom, co umożliwia efektywne przetwarzanie obrazów o dużej rozdzielczości. Dzięki zastosowaniu nowoczesnych warstw normalizacji i aktywacji, model jest w stanie dokładnie uchwycić złożone wzorce w danych obrazowych [Liu i in., 2022].

ResNet50 jest klasyczną architekturą głębokiej sieci neuronowej, która zyskała szerokie uznanie dzięki wprowadzeniu mechanizmu residual connections, umożliwiającego trenowanie bardzo głębokich sieci bez problemu zanikania gradientu [He i in., 2016]. W niniejszej pracy ResNet50 został zastosowany do klasyfikacji obrazów kukurydzy, koncentrując się na ocenie poziomów nawodnienia i zdrowotności roślin.



ResNet50 został zaimplementowany przy użyciu biblioteki `timm`, z zastosowaniem standardowego zestawu warstw konwolucyjnych i mechanizmu `skip connections`, co umożliwia efektywne przekazywanie informacji w głąb sieci. Model ten został dostosowany do specyficznych wymagań zadania poprzez zastosowanie intensywnych technik augmentacji danych, które miały na celu zwiększenie zdolności generalizacyjnych oraz stabilności modelu [He i in., 2016].

Model ResNet50 charakteryzuje się wysoką wydajnością obliczeniową, co pozwala na przetwarzanie dużych ilości danych obrazowych w stosunkowo krótkim czasie. W niniejszej pracy szczególną uwagę zwrócono na optymalizację procesu trenowania modelu, obejmującą dobór odpowiednich hiperparametrów, takich jak współczynnik uczenia, moment oraz wartość regularyzacji wagi [He i in., 2016].

EfficientNet\_B3 to model, który zyskał popularność dzięki zdolności do skalowania sieci w sposób efektywny, uwzględniając zarówno głębokość, szerokość, jak i rozdzielczość wejściową obrazów [Tan, Le, 2019]. W kontekście niniejszej pracy EfficientNet\_B3 został zastosowany do analizy obrazów kukurydzy, szczególnie pod kątem oceny poziomów nawodnienia oraz zdrowotności roślin.

Implementacja EfficientNet\_B3 obejmowała wykorzystanie modelu jako narzędzia do efektywnego przetwarzania obrazów o różnych rozdzielczościach, co jest szczególnie istotne w przypadku złożonych zadań klasyfikacyjnych. Model ten został skonfigurowany z uwzględnieniem specyficznych wymagań zadania, w tym z klasyfikacją trzech klas, oraz zastosowaniem zaawansowanych technik augmentacji, takich jak skalowanie, obracanie, zmiana jasności i kontrastu [Tan, Le, 2019].

Dzięki zastosowaniu architektury EfficientNet\_B3 możliwe było uzyskanie wysokiej precyzji w zadaniach klasyfikacyjnych, przy jednoczesnym ograniczeniu zużycia zasobów obliczeniowych. Model ten został zoptymalizowany pod kątem zwiększenia zdolności generalizacyjnych oraz stabilności procesu trenowania, co umożliwiło uzyskanie wyników o wysokiej jakości [Tan, Le, 2019].

Vision Transformer (ViT) to innowacyjna architektura, która przenosi mechanizmy `transformers` z dziedziny przetwarzania języka naturalnego do analizy obrazów [Dosovitskiy i in., 2020]. W niniejszej pracy ViT został zastosowany do klasyfikacji obrazów kukurydzy, koncentrując się na ocenie poziomów nawodnienia oraz zdrowotności roślin.

Model ViT charakteryzuje się nowatorskim podejściem do analizy obrazów, gdzie obrazy są dzielone na mniejsze fragmenty (patche), które są następnie przetwarzane jako sekwencje wejściowe [Dosovitskiy i in., 2020]. W niniejszej pracy implementacja ViT obejmowała konfigurację modelu do pracy z trzema klasami, a także zastosowanie intensywnych technik augmentacji danych, takich jak skalowanie, rotacja, zmiana jasności i kontrastu, co miało na celu zwiększenie zdolności generalizacyjnych oraz stabilności modelu.

Dzięki zastosowaniu mechanizmów transformers, Vision Transformer jest w stanie równocześnie analizować zarówno lokalne, jak i globalne cechy obrazu, co jest kluczowe w kontekście złożonych zadań klasyfikacyjnych [Dosovitskiy i in., 2020]. W niniejszej pracy ViT został zoptymalizowany pod kątem specyficznych wymagań analizy obrazów roślin, co umożliwiło uzyskanie wyników o wysokiej precyzji.

Swin Transformer to zaawansowana architektura oparta na Vision Transformer, która wprowadza mechanizm przesuwających się okienek (shifted windows), co pozwala na efektywne przetwarzanie obrazów o dużej rozdzielczości [Liu i in., 2021]. W niniejszej pracy Swin Transformer został zastosowany do analizy obrazów kukurydzy, z naciskiem na klasyfikację poziomów nawodnienia oraz zdrowotności roślin.

Implementacja Swin Transformer obejmowała konfigurację modelu do pracy z trzema klasami oraz zastosowanie zaawansowanych technik augmentacji danych, takich jak losowe rotacje, odbicia lustrzane, zmiany jasności oraz kontrastu [Liu i in., 2021]. Model ten został zoptymalizowany pod kątem efektywnego wykorzystania zasobów obliczeniowych, co było istotne w kontekście analizy dużych zbiorów danych obrazowych.

Swin Transformer charakteryzuje się zdolnością do uchwycenia złożonych wzorców w danych obrazowych dzięki zastosowaniu mechanizmu przesuwających się okienek, co umożliwia lepsze przetwarzanie obrazów o wysokiej rozdzielczości [Liu i in., 2021]. W niniejszej pracy Swin Transformer został dostosowany do specyficznych wymagań analizy obrazów roślin, co pozwoliło na uzyskanie wyników o wysokiej precyzji i efektywności [Liu i in., 2021].

### 3.4. Proces trenowania modeli

W niniejszej pracy zastosowano i dostosowano różnorodne architektury głębokich sieci neuronowych, takie jak Hybrid Task Cascade (HTC) [Chen i in., 2019], YOLOX [Ge i in., 2021], Mask2Former [Chen i in., 2021], ConvNeXt small [Liu i in., 2022], ResNet50 [He i in.,

2016], EfficientNet\_B3 [Tan & Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] oraz Swin Transformer [Liu i in., 2021], w celu analizy obrazów kukurydzy. Celem procesu trenowania było uzyskanie stabilnych i dokładnych wyników, co wymagało starannej optymalizacji parametrów, takich jak liczba epok, wielkość batchu, funkcja kosztu, algorytm optymalizacji, a także modyfikacje architektury modeli.

Modele były trenowane przez różną liczbę epok, dostosowaną do specyfiki danej architektury i zadania. Na przykład, modele HTC, Mask2Former oraz YOLOX były trenowane przez 100 epok, co pozwoliło na uzyskanie stabilnych wyników. W przypadku modelu YOLOX początkowo przewidziane 300 epok zostało skrócone do 100, aby przyspieszyć proces treningu bez kompromisów na jakości wyników [Ge i in., 2021]. Modele ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer były trenowane przez 40 epok, z logowaniem wyników do TensorBoard [Liu i in., 2022].

Wielkość batchu była zróżnicowana w zależności od modelu i dostępnych zasobów sprzętowych. Na przykład w modelu Mask2Former, ze względu na ograniczenia pamięci GPU, zdecydowano się na batch o wielkości 1, podczas gdy w modelach HTC stosowano batch o wielkości 2 [Chen i in., 2021]. W przypadku modeli ConvNeXt, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer, batch size został ustawiony na 8, co pozwalało na efektywne przetwarzanie danych [He i in., 2016], [Tan & Le, 2019].

Optymalizacja modeli opierała się na algorytmie SGD (Stochastic Gradient Descent) lub Adam, w zależności od architektury. SGD był stosowany w modelach HTC, Mask2Former oraz YOLOX, z początkową wartością współczynnika uczenia wynoszącą 0,0003, momentem równym 0,9 oraz regularyzacją wagi wynoszącą 0,0001 [Chen i in., 2019], [Ge i in., 2021]. W modelach ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer zastosowano optymalizator Adam z początkowym współczynnikiem uczenia wynoszącym  $1e-5$ , co umożliwiało bardziej precyzyjne dostosowanie tempa uczenia do specyficznych wymagań zadania [Kingma & Ba, 2015].

W celu oceny wydajności modeli zastosowano różne metryki ewaluacyjne, takie jak mAP (mean Average Precision), dokładność, precyzja, czułość oraz współczynnik IoU (Intersection over Union). Wyniki trenowania były monitorowane za pomocą TensorBoard, co umożliwiało bieżące dostosowywanie parametrów oraz kontrolowanie procesu trenowania [Chen i in., 2019], [Liu i in., 2022].

### 3.4.1. Trenowanie modeli HTC, Mask2Former, YOLOX

Trenowanie modeli HTC, Mask2Former oraz YOLOX odbyło się w środowisku opartym na bibliotece PyTorch, z wykorzystaniem funkcji i modułów dostępnych w frameworku MMDetection [Chen i in., 2019], [Chen i in., 2021], [Ge i in., 2021]. Każdy model został starannie przystosowany do specyficznych zadań związanych z detekcją i segmentacją obrazów kukurydzy.

Proces trenowania rozpoczęto od podziału dostępnych danych na zbiory treningowe i walidacyjne. Dane wejściowe były poddawane intensywnej augmentacji, co miało na celu zwiększenie różnorodności próbek treningowych i poprawę zdolności generalizacyjnych modeli. Techniki augmentacji obejmowały rotacje, odbicia lustrzane oraz zmiany rozmiaru i jasności obrazów [Shorten & Khoshgoftaar, 2019].

Optymalizacja modeli opierała się na zastosowaniu optymalizatora SGD, z początkowym współczynnikiem uczenia wynoszącym 0,0003, momentem równym 0,9 oraz regularyzacją wagi na poziomie 0,0001 [Kingma & Ba, 2015]. Zastosowany harmonogram uczenia oparty na krokach (step-based learning rate schedule) pozwalał na adaptacyjne dostosowywanie tempa uczenia do zmieniających się warunków treningowych [Chen i in., 2019], [Ge i in., 2021].

W celu oceny wydajności i dokładności modeli zastosowano różne metryki ewaluacyjne, takie jak mAP, dokładność, precyzja, czułość oraz współczynnik IoU [He i in., 2016] [Ge i in., 2021]. Wyniki trenowania były monitorowane za pomocą narzędzi takich jak TensorBoard, co umożliwiało dostosowywanie parametrów i podejmowanie decyzji o zakończeniu procesu treningowego [Chen i in., 2019].

### 3.4.2. Trenowanie modeli ConvNeXt, ResNet, EfficientNet, Vision Transformer, Swin Transformer

Modele ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer były trenowane w środowisku opartym na bibliotece timm, z zastosowaniem optymalizatora Adam, z początkową wartością współczynnika uczenia wynoszącą  $1e-5$  [Wightman, 2019], [Kingma & Ba, 2015]. Proces trenowania obejmował 40 epok, podczas których dane wejściowe były przetwarzane przy użyciu zaawansowanych technik augmentacji, takich jak rotacje, odbicia lustrzane oraz zmiany jasności [Shorten & Khoshgoftaar, 2019].

Celem augmentacji było zwiększenie zdolności generalizacyjnych modeli oraz ich odporności na zmienność danych wejściowych.

Optymalizacja tych modeli opierała się na zastosowaniu algorytmu Adam, który pozwala na bardziej precyzyjne dostosowanie tempa uczenia w zależności od postępu procesu treningowego [Kingma & Ba, 2015]. Zastosowano także techniki regularyzacji, takie jak dropout oraz weight decay, aby zapobiec przeuczeniu modeli i zapewnić ich lepszą generalizację na nowych danych [Srivastava i in., 2014].

W celu oceny wydajności i dokładności modeli ConvNeXt, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer zastosowano różne metryki ewaluacyjne, takie jak accuracy, precision, recall oraz F1-score [Sokolova & Lapalme, 2009]. Wyniki trenowania były monitorowane i analizowane z użyciem TensorBoard, co umożliwiało śledzenie postępów, analizowanie błędów oraz optymalizację parametrów w trakcie treningu [Abadi i in., 2016].

Trenowanie tych modeli odbywało się na zaawansowanym sprzęcie GPU, co umożliwiało efektywne przetwarzanie dużych zbiorów danych i skrócenie czasu potrzebnego na uzyskanie stabilnych wyników [Huang i in., 2019].

### 3.4.3. Techniczne aspekty trenowania modeli

Trenowanie modeli w ramach niniejszej pracy odbywało się na stacjonarnym komputerze wyposażonym w kartę graficzną NVIDIA RTX 3080. Karta ta jest znana z doskonałej wydajności w zadaniach związanych z uczeniem maszynowym, oferując 10 GB pamięci VRAM oraz architekturę Ampere, co umożliwia efektywne przetwarzanie złożonych obliczeń związanych z głębokimi sieciami neuronowymi [NVIDIA, 2020].

Karta graficzna umożliwiała szybsze przetwarzanie danych w porównaniu do tradycyjnych procesorów CPU, co znacząco skróciło czas potrzebny na trenowanie modeli. Wykorzystanie jednostek CUDA przyczyniło się do poprawy wydajności obliczeń związanych z propagacją wsteczną oraz aktualizacją wag modeli [Nickolls i in., 2008].

Identyczna karta graficzna znajduje się na prototypie robota polowego, skąd prowadzone obliczenia można analizować na komputerze stacjonarnym tak, jakby były wykonywane na rzeczywistym robocie podczas jego pracy w polu. Częstotliwość przetwarzania obrazów z użyciem najlepszego z badanych algorytmów, czyli Hybrid Task

Cascade (HTC) z ResNeXt101 jako backbone wynosi nieco poniżej 1 Hz, co z dużym nadstatkiem spełnia wymagania projektowe. Realizację tego punktu zadania można zatem podsumować jako pełen sukces.

Wszystkie eksperymenty były przeprowadzane w środowisku programistycznym opartym na języku Python, z wykorzystaniem popularnych bibliotek do głębokiego uczenia:

- PyTorch [Paszke i in., 2019]
- MMDetection [MMDetection, 2020]
- Timm [Wightman, 2019]
- TensorBoard [Abadi i in., 2016]

Środowisko było skonfigurowane w środowisku wirtualnym Anaconda zainstalowanym na systemie Windows, a do zarządzania pakietami oraz zależnościami użyto narzędzi takich jak Conda oraz Pip [Anaconda, 2020].

Czas trenowania modeli różnił się w zależności od architektury oraz złożoności zadania. Przykładowo, trenowanie modeli HTC\_r101 na zestawie danych RGB obejmującym 1000 obrazów reprezentujących określone fazy rozwoju kukurydzy trwało około 11 godzin, a analogiczny trening z wykorzystaniem modelu Mask2Former trwał około 17 godzin [Chen i in., 2019], [Chen i in., 2021], [Ge i in., 2021]. Z kolei trenowanie modeli ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer w celu określania poziomów nawodnienia kukurydzy trwało średnio 5 godzin na model [Liu i in., 2022], [He i in., 2016], [Tan & Le, 2019], [Dosovitskiy i in., 2020], [Liu i in., 2021]. Zasoby obliczeniowe były w pełni wykorzystywane podczas trenowania, z GPU pracującym na maksymalnej wydajności [NVIDIA, 2020].

### 3.5. Proces detekcji i klasyfikacji obiektów

Proces detekcji i klasyfikacji obiektów w ramach niniejszej pracy obejmował kilka kluczowych etapów, począwszy od wstępnej implementacji modeli głębokich sieci neuronowych, aż po ostateczną analizę wyników. Celem tego procesu było precyzyjne rozpoznanie i klasyfikacja różnych faz rozwoju kukurydzy zgodnie ze skalą BBCH, a także ocena poziomów nawodnienia kukurydzy oraz identyfikacja roślin zdrowych i porażonych wybranymi patogenami.



Pierwszym krokiem było przygotowanie i dostosowanie modeli głębokich sieci neuronowych do specyfiki każdego zadania. W pracy zastosowano dwa zestawy zaawansowanych modeli, dostosowanych do odpowiednich celów badawczych. Do zadania detekcji i klasyfikacji faz rozwojowych kukurydzy zastosowano modele HTC\_x101, HTC\_r101, HTC\_r50, Mask2Former oraz YOLOX. Każdy z tych modeli został zaimplementowany z wykorzystaniem kodów źródłowych pochodzących z repozytorium MMDetection, które następnie odpowiednio zmodyfikowano, aby optymalnie działały w kontekście analizy obrazów rolniczych.

Modele HTC (Hybrid Task Cascade), w szczególności HTC\_x101, HTC\_r101 i HTC\_r50, wybrano ze względu na ich zdolność do jednoczesnej segmentacji semantycznej, instancyjnej oraz detekcji obiektów. Implementacja tych modeli obejmowała dostosowanie sieci bazowych, takich jak ResNet-50, ResNet-101 i ResNeXt-101, oraz dostosowanie parametrów treningowych, takich jak współczynnik uczenia, liczba epok treningowych i architektura Feature Pyramid Network (FPN), która umożliwia efektywne przetwarzanie cech z różnych poziomów sieci.

Model Mask2Former, oparty na architekturze Transformer, wdrożono z myślą o precyzyjnej segmentacji obiektów na poziomie instancyjnym. Jego zdolność do analizy zarówno globalnych, jak i lokalnych cech obrazów umożliwia dokładne wykrywanie i klasyfikację poszczególnych faz rozwoju kukurydzy, co jest szczególnie istotne w złożonych scenach rolniczych.

Drugi zestaw modeli – ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer – zastosowano do zadań związanych z określaniem poziomów nawodnienia kukurydzy oraz klasyfikacją stanu zdrowotnego roślin (czy są zdrowe, czy porażone patogenami). Modele te zostały wybrane ze względu na ich zdolność do efektywnego przetwarzania złożonych wzorców na obrazach oraz analizy danych multispektralnych. ConvNeXt small, będący nowoczesną architekturą łączącą elementy klasycznych konwolucyjnych sieci neuronowych z technologiami inspirowanymi Transformerami, został wybrany ze względu na jego zdolność do efektywnego przetwarzania obrazów przy zachowaniu wysokiej precyzji. ResNet50, znany ze swojego mechanizmu residual connections, umożliwił skuteczne trenowanie głębokiej sieci przy zachowaniu stabilności procesu uczenia. EfficientNet\_B3, model znany ze zdolności do efektywnego skalowania, pozwolił na uzyskanie wysokiej dokładności przy optymalnym zużyciu zasobów obliczeniowych. Vision Transformer i Swin Transformer, reprezentujące nowoczesne

podejście do przetwarzania obrazów z wykorzystaniem mechanizmów Transformer, zostały zastosowane ze względu na ich zdolność do jednoczesnego przetwarzania lokalnych i globalnych cech obrazów.

Po wstępnej implementacji każdy z modeli został przeszkolony na odpowiednio przygotowanych zbiorach danych obrazowych. Proces trenowania obejmował ładowanie obrazów, ich wstępną obróbkę, normalizację oraz randomizację, co miało na celu zwiększenie generalizacji modeli i poprawę ich odporności na różnorodność danych wejściowych. Zbiory danych treningowych dla modeli HTC oraz Mask2Former zawierały obrazy przedstawiające różne fazy rozwoju kukurydzy, co umożliwiło modelom naukę rozpoznawania i klasyfikacji tych faz zgodnie ze skalą BBCH. Natomiast zbiory danych dla modeli ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer zawierały obrazy pozwalające na ocenę poziomów nawodnienia kukurydzy oraz identyfikację stanu zdrowotnego roślin. Trenowanie modeli zoptymalizowano pod kątem efektywnego wykorzystania zasobów obliczeniowych, z zastosowaniem zaawansowanych technik optymalizacji, takich jak Stochastic Gradient Descent (SGD), Adam oraz harmonogramy uczenia oparte na krokach (step-based learning rate schedule). Każdy model był trenowany przez określoną liczbę epok, a w trakcie treningu monitorowano jego wydajność, aby zapewnić optymalne dostrojenie parametrów.

Po przeszkoleniu modele zastosowano do detekcji i klasyfikacji na nowych, nieznanych wcześniej obrazach. Proces ten obejmował kilka kluczowych etapów. Każdy obraz został wstępnie przetworzony zgodnie z wcześniej ustalonym pipeline'em, który obejmował normalizację, zmniejszanie rozdzielczości oraz przygotowanie danych w formacie odpowiednim do przetwarzania przez modele.

Modele HTC i Mask2Former zastosowano do wykrywania obiektów na obrazach w zadaniu klasyfikacji faz rozwoju kukurydzy. Wykrywanie polegało na identyfikacji poszczególnych faz rozwoju kukurydzy oraz przypisaniu odpowiednich etykiet klasyfikacyjnych zgodnie ze skalą BBCH. Wykryte obiekty były następnie segmentowane, a każdej wykrytej fazie rozwoju kukurydzy przypisywano odpowiednią etykietę klasyfikacyjną. Modele HTC oraz Mask2Former, dzięki swoim zaawansowanym mechanizmom segmentacji instancyjnej, umożliwiły precyzyjne rozdzielenie i klasyfikację różnych faz rozwoju roślin w złożonych scenach.

Z kolei modele ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer były stosowane do zadań związanych z oceną poziomów nawodnienia kukurydzy oraz identyfikacją stanu zdrowotnego roślin. Modele te były szczególnie skuteczne w analizie obrazów multispektralnych oraz RGB, w których kluczowe było jednocześnie przetwarzanie lokalnych i globalnych cech. Dzięki zaawansowanym technikom przetwarzania danych, takie jak konwolucje (w ConvNeXt small, ResNet50 i EfficientNet\_B3) oraz mechanizmy Transformer (w Vision Transformer i Swin Transformer), modele te były w stanie precyzyjnie wykrywać i klasyfikować różnorodne stany roślin na podstawie obrazów.

Wyniki detekcji i klasyfikacji zostały dokładnie przeanalizowane, aby ocenić skuteczność i precyzję zastosowanych modeli. Analiza ta obejmowała ocenę jakości predykcji na podstawie wskaźników takich jak dokładność, precyzja, recall oraz miara F1, co pozwoliło na pełną ocenę zdolności modeli do wykrywania i klasyfikacji zarówno faz rozwoju kukurydzy, jak i poziomów nawodnienia oraz stanu zdrowotnego roślin.

Wyniki uzyskane w procesie detekcji i klasyfikacji porównano z danymi referencyjnymi, aby ocenić dokładność modeli w rozpoznawaniu faz rozwoju kukurydzy oraz w identyfikacji stanu zdrowotnego roślin. Modele HTC\_x101, HTC\_r101 i HTC\_r50 wykazały wysoką precyzję w segmentacji i klasyfikacji, szczególnie w złożonych scenariuszach. Mask2Former, dzięki swojej zaawansowanej architekturze Transformer, okazał się szczególnie efektywny w segmentacji instancyjnej, umożliwiając precyzyjną identyfikację poszczególnych faz rozwoju. Natomiast modele ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer wykazały wysoką skuteczność w zadaniach związanych z oceną nawodnienia oraz identyfikacją porażenia roślin przez patogeny.

### 3.5.1. Zastosowanie modeli

W ramach niniejszej pracy, implementacja modeli do detekcji i klasyfikacji faz rozwoju kukurydzy, poziomów nawodnienia oraz stanu zdrowotnego roślin została przeprowadzona w specjalnie przygotowanym środowisku badawczym. Aby zapewnić optymalną wydajność oraz dokładność analiz, wykorzystano zaawansowane narzędzia i technologie, takie jak PyTorch oraz środowisko obliczeniowe oparte na jednostkach GPU. W szczególności użyto karty graficznej NVIDIA RTX 3080, która dzięki swojej dużej mocy obliczeniowej umożliwiła efektywne przetwarzanie złożonych modeli głębokiego uczenia. Karta ta, wyposażona w 10 GB pamięci GDDR6X oraz 8704 rdzeni CUDA, zapewniła doskonałą wydajność w zadaniach

związanych z trenowaniem modeli głębokich sieci neuronowych. Dzięki temu możliwe było szybkie przetwarzanie dużych zbiorów danych obrazowych, co znacznie przyspieszyło proces implementacji i trenowania modeli takich jak HTC\_x101, HTC\_r101, HTC\_r50, Mask2Former, ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer.

Cały proces implementacji i trenowania modeli odbywał się w środowisku PyTorch, które oferuje bogaty zestaw funkcji do efektywnego wykorzystania mocy obliczeniowej GPU. Dzięki wsparciu dla CUDA, PyTorch umożliwił przetwarzanie równoległe, co było kluczowe dla szybkiego i efektywnego trenowania złożonych modeli głębokiego uczenia. Implementacja modeli rozpoczęła się od przygotowania środowiska obliczeniowego, które obejmowało instalację niezbędnych bibliotek, takich jak PyTorch, MMDetection, timm oraz sterowników CUDA, co umożliwiło pełne wykorzystanie możliwości karty NVIDIA RTX 3080. Następnie przystąpiono do implementacji i dostosowania modeli HTC\_x101, HTC\_r101, HTC\_r50, Mask2Former, ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer do specyficznych wymagań zadania detekcji i klasyfikacji faz rozwoju kukurydzy oraz oceny poziomów nawodnienia i stanu zdrowotnego roślin. Każdy model został skonfigurowany z uwzględnieniem parametrów, takich jak głębokość sieci bazowych, liczba epok treningowych oraz współczynnik uczenia.

Dane wejściowe, czyli obrazy przedstawiające kukurydzę w różnych fazach rozwoju, a także obrazy dotyczące oceny nawodnienia i stanu zdrowotnego roślin, zostały wstępnie przetworzone zgodnie z opracowanym pipeline'em, który obejmował normalizację, skalowanie do odpowiednich rozmiarów oraz randomizację. Przetworzone obrazy były następnie wprowadzane do modeli, które przeprowadzały proces detekcji i klasyfikacji. Wprowadzenie danych do modeli odbywało się w formie batchy, co umożliwiało równoczesne przetwarzanie wielu obrazów, maksymalizując efektywność obliczeń na GPU.

W szczególności, modele HTC (Hybrid Task Cascade) przeprowadzały jednoczesną segmentację semantyczną i instancyjną, co pozwalało na precyzyjne rozdzielenie poszczególnych faz rozwoju w złożonych scenach. Mask2Former, dzięki swojej architekturze Transformer, analizował zarówno globalne, jak i lokalne cechy obrazów, umożliwiając dokładne wykrywanie i klasyfikację faz rozwoju kukurydzy.

Modele ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer oraz Swin Transformer były wykorzystywane do klasyfikacji poziomów nawodnienia kukurydzy oraz

oceny zdrowotności roślin. Modele te były szczególnie skuteczne w analizie obrazów multispektralnych oraz RGB, w których kluczowe było jednocześnie przetwarzanie lokalnych i globalnych cech. Dzięki zaawansowanym technikom przetwarzania danych, takie jak konwolucje (w ConvNeXt small, ResNet50 i EfficientNet\_B3) oraz mechanizmy Transformer (w Vision Transformer i Swin Transformer), modele te były w stanie precyzyjnie wykrywać i klasyfikować różnorodne stany roślin na podstawie obrazów.

Podczas implementacji napotkałam kilka wyzwań technicznych, które wymagały opracowania skutecznych rozwiązań. Jednym z głównych problemów było zabezpieczenie przed brakiem pamięci GPU (Out of Memory), co mogło prowadzić do przerwania procesu trenowania modeli, zwłaszcza przy przetwarzaniu dużych batchy lub obrazów o wysokiej rozdzielczości. Aby zminimalizować ryzyko wystąpienia tego problemu, zastosowałam techniki optymalizacji pamięci, takie jak zmniejszenie rozmiaru batchy oraz obniżenie rozdzielczości obrazów w trakcie treningu. W modelach HTC oraz Mask2Former, ograniczono liczbę zapytań w mechanizmie Transformera oraz dostosowano maksymalną liczbę wykryć na obraz, co pozwoliło na bardziej efektywne zarządzanie pamięcią GPU.

Ponadto, monitorowanie wykorzystania zasobów GPU na bieżąco umożliwiło szybką reakcję na potencjalne problemy, co zapewniło stabilność procesu trenowania. Kolejnym wyzwaniem było zapewnienie stabilności i konwergencji modeli podczas treningu. Zastosowanie optymalizatora Stochastic Gradient Descent (SGD) oraz Adam z odpowiednio dobranymi parametrami, takimi jak współczynnik uczenia i momentum, pozwoliło na stabilne uczenie modeli i osiągnięcie optymalnych wyników bez ryzyka przeuczenia.

Dzięki tym działaniom, zaimplementowane modele działały poprawnie i były w stanie skutecznie wykrywać i klasyfikować fazy rozwoju kukurydzy oraz oceniać poziomy nawodnienia i stan zdrowotny roślin. To z kolei umożliwiło osiągnięcie wysokiej precyzji i skuteczności w zadaniu analizy obrazów rolniczych, co stanowiło istotny element w realizacji celów badania.

### 3.5.2. Klasyfikator głosujący i jego zastosowanie

Dla danych ze zbioru obrazów wielospektralnych zarejestrowanych przez kamerę MicaSense Red Edge MX-Dual w 10 pojedynczych kanałach spektralnych można zauważyć pewną prawidłowość dotyczącą obrazów tego samego obiektu wykonanych przy użyciu różnych obiektywów odpowiadających poszczególnym kanałom spektralnym. Obiekty na

poszczególnych obrazach były umiejscowione w różnych odległościach od krawędzi obrazów, co jest zrozumiałe przy różnym kącie, pod którym fotografowany jest dany obiekt. Dodatkowo, szczególnie w przypadku obiektów kukurydzy reprezentujących wyższe fazy rozwojowe (fazy 14-19 w skali BBCH), różne kąty fotografowania powodowały zmianę kształtu rejestrowanych obiektów. Podczas gdy z jednego obiektywu liść był dość prosty, inny obiektyw rejestrował go jako wygięty. To uniemożliwiło proces nakładania kilku obrazów reprezentujących pojedyncze kanały spektralne w jedno zobrazowanie.

Należało zatem obrać inną drogę postępowania, aby do detekcji i klasyfikacji obiektów zarejestrowanych na obrazach z kamery wielospektralnej wykorzystywać jednocześnie wszystkie kanały spektralne kamery RedEdge. Naturalnym rozwiązaniem okazało się przyjęcie metody głosowania poszczególnych modeli wytrenowanych na pojedynczych kanałach spektralnych. W opracowanym rozwiązaniu ocenie podlegały dwa podejścia wyboru klasy:

- single model - model wytrenowany na jednym kanale spektralnym lub na kanałach RGB,
- ensemble - 10 modeli wytrenowanych na różnych kanałach spektralnych głosuje nad wyborem klasy (BBCH 10-19) i wygrywa klasa większościowa.

Klasyfikator głosujący to zaawansowana metoda agregacji wyników predykcji pochodzących z różnych modeli, która może poprawić skuteczność klasyfikacji w systemach uczenia maszynowego. W badaniach opisanych w niniejszej rozprawie, klasyfikator ten został zastosowany do analizy wyników modeli uczenia głębokiego wytrenowanych na pojedynczych kanałach spektralnych [Breiman, 1996]. W celu realizacji tego podejścia stworzono dedykowany kod, który implementuje poszczególne etapy przetwarzania danych oraz obliczania wyników.

Kod obsługujący klasyfikator głosujący został zorganizowany w zestaw funkcji, które realizują różnorodne zadania związane z przetwarzaniem danych, obliczaniem wyników oraz ich agregacją. Kluczową rolę w całym procesie odgrywa funkcja `get_probabilities`, która odpowiedzialna jest za ekstrakcję prawdopodobieństw predykcji z wyników modeli zapisanych w plikach `pickle` odpowiadających poszczególnym modelom wytrenowanym na pojedynczych kanałach spektralnych. Funkcja ta wczytuje dane, przetwarza je w celu uzyskania informacji o przewidywanej kategorii dla każdego obrazu wraz z przypisanym jej prawdopodobieństwem, a następnie zapisuje wyniki w formie ramki danych, co ułatwia ich dalsze przetwarzanie.

W opracowanym podejściu w przypadku agregacji predykcji i wyników wykrycia w postaci poligonów, istotne były predykcje dotyczące konkretnych roślin, a nie całych obrazów. Wszystkie predykcje zostały zagregowane na podstawie identyfikatora rośliny (`file_id` w bazie



danych). Dla każdej rośliny zarejestrowano wiele predykcji i wiele wyników wykrycia w postaci poligonów, więc należało wybrać jedną klasę dla każdej rośliny.

Wybór przewidywanej klasy z wyników wykrycia w postaci poligonów polegał na tym, że z predykcji zwracanych przez model wybierana była pojedyncza klasa, która występowała najczęściej. W przypadku remisu wybierana była klasa o największym średnim score poligonu. W przypadku wykrytych poligonów, gdy obraz był wykryty jako więcej niż jedna klasa, zastosowaliśmy podobne podejście. Wybrana została klasa, która występowała najczęściej, a w przypadku remisu wybierana była klasa o największej średniej powierzchni poligonu.

Każdy z dziesięciu modeli głosujących był wytrenowany na jednym z dziesięciu zbiorów obrazów odpowiadających poszczególnym kanałom spektralnym kamery RedEdge. Modele te zostały wytrenowane z wykorzystaniem algorytmu HTC z ResNeXt101 jako backbone, dostępnego w repozytorium MMDetection [He i in., 2016]. Wybór tego algorytmu był uzasadniony najlepszymi wynikami osiągniętymi przy jego pomocy.

Klasyfikator głosujący działa w oparciu o zasadę polegającą na tym, że dla każdego z klasyfikatorów podstawowych dokonywana jest predykcja danej rośliny, przy czym obrazy roślin są robione przez różne kamery z nieco innych kątów nachylenia, ale można odróżnić kolejne rośliny na podstawie id przypisanych do nazwy pliku z obrazem. Następnie wybierana jest klasa większościowa, czyli taka, dla której było najwięcej głosów (predykcji) z poszczególnych klasyfikatorów. Klasyfikator głosujący agreguje wyniki i zwraca klasę większościową dla każdego z modeli.

Na danym obrazie może być zarejestrowanych więcej niż jedna roślina, lecz w założeniach zadania podkreślono, że wysiew nasion kukurydzy odbywa się w tym samym czasie, więc teoretyczne różnice w rozwoju poszczególnych roślin powinny być minimalne. Celem nadrzędnym klasyfikacji jest zbieranie danych na temat faz rozwojowych kukurydzy w celu prowadzenia wizualizacji mapowej stanu rozwoju roślin kukurydzy na monitorowanym obszarze. Stąd też precyzja zapisywania wyników klasyfikacji faz rozwojowych z dokładnością do jednego obrazu (wykonywanego z częstotliwością około 1 Hz) jest w zupełności wystarczająca i zgodna z założeniami projektowymi.

Agregacja uzyskanych w ten sposób wyników dla klasyfikatora głosującego polega na tym, że każdy z pojedynczych klasyfikatorów (dla kolejnych kanałów spektralnych) zwraca wyniki w postaci:

- obraz\_1 :
  - klasa\_1:
    - polygon\_1: score

- polygon\_2: score
  - ...
- klasa\_2:
- ...
- obraz\_2
- ...

Klasyfikator głosujący agreguje wyniki tak, aby wybrać największy score dla danego obrazu i przypisuje klasę największego score'a do danego obrazu. Opracowane podejście bierze pod uwagę największy score ze wszystkich klas i to stanowi klasę wynikową dla danego obrazu. Otrzymane wyniki przyjmują formę:

- obraz\_1: klasa\_6
- obraz\_2: klasa\_5
- ...

Agregacja oznaczeń w zbiorze walidacyjnym do klas polega na tym, że dla każdego z kanałów spektralnych zostają zdefiniowane poligony z różnymi klasami. Czasami na jednym obrazku w danym kanale spektralnym może pojawić się więcej niż jeden poligon dla danej klasy, a nawet poligony dla różnych klas w różnych miejscach obrazu. Na potrzeby opracowanego eksperymentu należało uzyskać dla każdego obrazka jedną konkretną klasę. Zostało to spełnione poprzez wybranie dla każdego obrazka poligonu o największej powierzchni.

Wstępna struktura danych dla każdego z kanałów spektralnych wygląda następująco:

- obraz\_1 :
  - klasa\_1:
    - polygon\_1: area
    - polygon\_2: area
    - ...
  - klasa\_2:
    - polygon\_1: area
    - polygon\_2: area
    - ...

- ...
- obraz\_2
- ...

Zaś otrzymane wyniki przyjmują następującą formę:

- obraz\_1: klasa\_6
- obraz\_2: klasa\_5
- ...

Warto również przybliżyć przyjętą metodę porównywania predykcji do rzeczywistych klas. Predykcje są grupowane dla każdej z roślin (`file_id`) w taki sposób, że ze wszystkich kanałów spektralnych typowana jest klasa najczęściej wybierana, czyli statystyczna moda. Jeśli dwie lub więcej klas są równoliczne, to w takim przypadku porównywane są następujące wartości:

- „area” w przypadku wykrytych poligonów – wybierana jest klasa o największej średniej powierzchni poligonów
- „score” w przypadku predykcji – wybierana jest klasa o największym średnim score.

Po wyciągnięciu zagregowanych klas można wyliczyć typowe dla klasyfikacji metryki takie jak *accuracy* (stosunek poprawnie sklasyfikowanych obiektów do wszystkich obiektów), *precision* (stosunek poprawnie sklasyfikowanych obiektów danej klasy do wszystkich obiektów sklasyfikowanych jako ta klasa), *recall* (stosunek poprawnie sklasyfikowanych obiektów danej klasy do wszystkich obiektów tej klasy) oraz *F1-score* (średnia harmoniczna *precision* i *recall*) [Powers, 2011].

Opracowanie takiego podejścia pociągnęło za sobą dodatkowy cel badania, czyli porównanie jakości klasyfikacji faz rozwoju kukurydzy w odniesieniu do dwóch wariantów modeli:

- a) modelu wytrenowanego na danych pochodzących z kamery RGB,
- b) modelu opartego na klasyfikatorze głosującym, złożonym z 10 modeli wytrenowanych na danych pochodzących z kamery wielospektralnej z poszczególnych kanałów spektralnych.

Po przetworzeniu danych z obu źródeł, tj. predykcji oraz adnotacji, następuje etap porównania i oceny wyników, realizowany przez funkcję `calculate_metrics`. Jest ona odpowiedzialna za obliczenie kluczowych miar jakości klasyfikacji, takich jak dokładność

(accuracy), precyzja (precision), czułość (recall) oraz miara F1 score. Funkcja ta łączy dane o rzeczywistych i przewidywanych kategoriach, a następnie na tej podstawie oblicza wspomniane miary, co umożliwia kompleksową ocenę skuteczności klasyfikatora głosującego [Zhang & Ma, 2012].

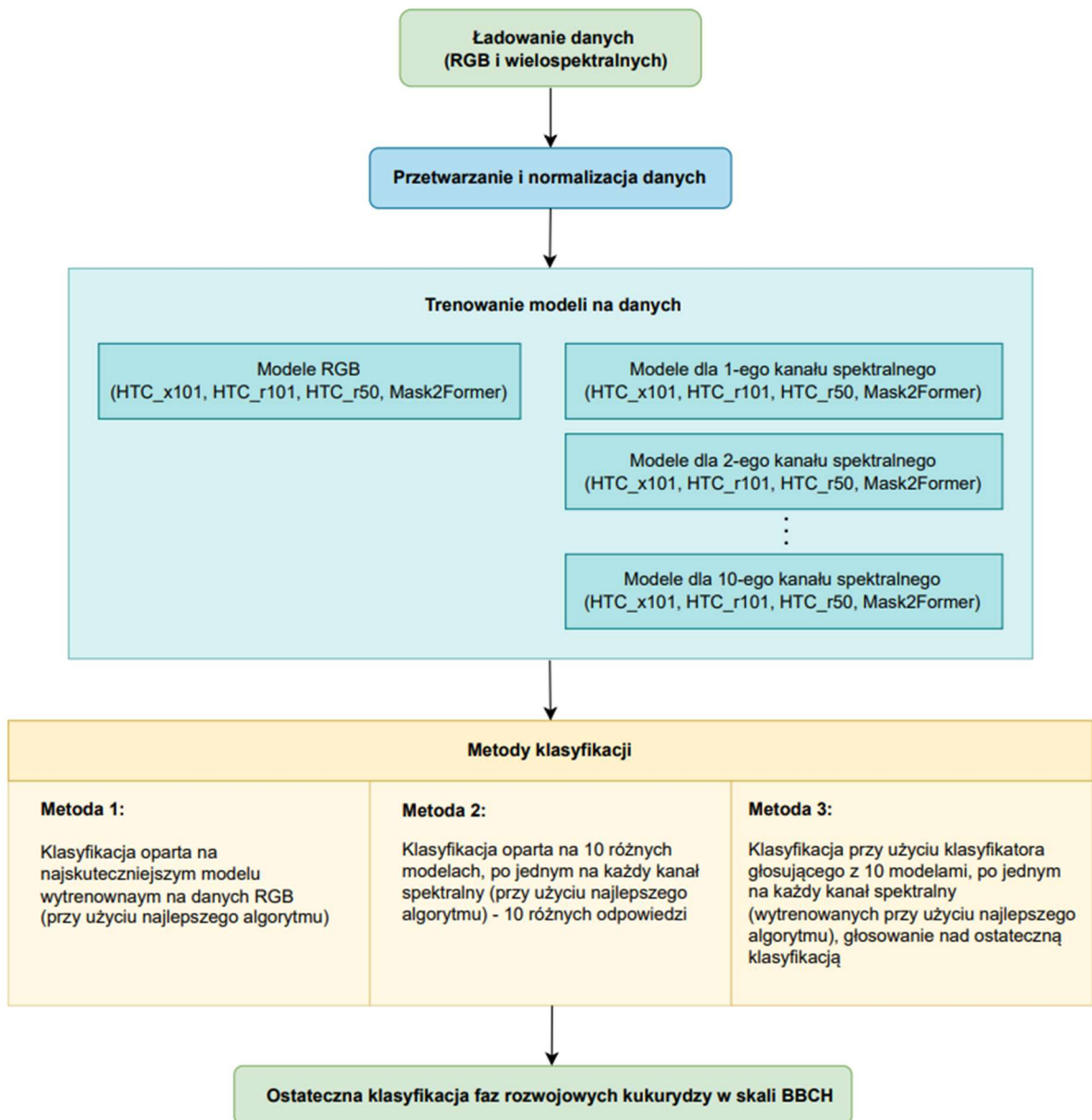
Główny proces walidacji wyników jest realizowany przez funkcję `validate`, która łączy wyniki predykcji i adnotacji, a następnie wywołuje funkcję `calculate_metrics`, aby obliczyć miary jakości. Funkcja `validate` dostosowuje sposób przetwarzania danych oraz ich ocenę w zależności od typu klasyfikatora, którym w tym przypadku jest klasyfikator głosujący oparty na modelach odpowiadających poszczególnym pojedynczym kanałom spektralnym.

## **4. Detekcja i klasyfikacja faz rozwoju kukurydzy w skali BBCH z wykorzystaniem obrazowania RGB i kamery wielospektralnej RedEdge**

Niniejszy rozdział skupia się na replikacji wyników uzyskanych w ramach zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH z zastosowaniem następujących architektur uczenia głębokiego: HTC\_r101, HTC\_r50, HTC\_x101 oraz Mask2Former. Architektury te mają optymalne parametry pod względem przeprowadzania procesu trenowania i nie wymagają zbyt dużych zasobów obliczeniowych do trenowania. Zbadana została ich efektywność i skuteczność w przypadku trenowania na zbiorze RGB oraz na zbiorze obrazów wielospektralnych z kamery MicaSense Red Edge MX Dual. Dodatkowo, badania zostały przeprowadzone dzieląc dataset danych wielospektralnych na poszczególne pojedyncze kanały spektralne, co umożliwiło uzyskanie odpowiedzi, który z badanych algorytmów radzi sobie najlepiej na wskazanych zbiorach danych oraz którego zbioru danych użyć, aby detekcja i klasyfikacja faz rozwojowych osiągnęła najlepsze efekty. Wprowadzona została także dodatkowa metoda z użyciem klasyfikatora głosującego, w której modele wytrenowane na pojedynczych kanałach spektralnych głosują nad wspólnym wynikiem wyboru klasy oznaczającej konkretną fazę rozwojową w skali BBCH. Te nowatorskie badania rozszerzają świat analizy danych o inne spektra niż popularne dotychczas zobrazowanie RGB. Prezentowane prace otwierają nowe pole do zbadania nowych zagadnień i były inspiracją do kontynuacji badań z wykorzystaniem nowego datasetu na połączonych kanałach spektralnych oraz użycia w tym celu również innej kamery wielospektralnej MapIR, co zostało opisane w rozdziale 5 niniejszej rozprawy.

### **4.1. Schemat opracowanego rozwiązania**

W sekcji tej przedstawiony zostanie szczegółowy opis rozwiązań użytych w badaniu dotyczącym detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH. Rysunek 4.1. pokazuje ogólny schemat zaproponowanego systemu.



**Rysunek 4.1.** Schemat zaproponowanego systemu klasyfikacji faz rozwojowych kukurydzy w skali BBCH.

Schemat przedstawia trzy główne podejścia zastosowane w opracowanym systemie: klasyfikacja na obrazach RGB, klasyfikacja na pojedynczych kanałach spektralnych oraz klasyfikacja z użyciem klasyfikatora głosującego. Rozwiązania te zaproponowano, aby sprawdzić i porównać wyniki otrzymane przy zastosowaniu wszystkich dostępnych typów zobrażeń, czyli zobrażenia RGB oraz zobrażenia pozyskiwanego za pomocą kamery wielospektralnej MicaSense Red Edge MX Dual. Co warto podkreślić, z uwagi na przesunięcie



i różny kształt obiektów rejestrowanych poprzez poszczególne kanały kamery wielospektralnej, wynikające z pozyskiwania obrazów z bliskiej odległości i różnych kątów patrzenia kamery, niemożliwe było połączenie tychże kanałów w jedno wspólne zobrazowanie. Konieczna była zatem analiza wyników generowanych przez modele wytrenowane przez pojedyncze kanały spektralne. Aby jednak wykorzystać wspólny potencjał, który tkwi w zobrazowaniach generowanych przez zastosowaną kamerę wielospektralną, opracowano metodę klasyfikacji opartą o klasyfikator głosujący, w którym modele wytrenowane na pojedynczych kanałach spektralnych głosują nad wspólną odpowiedzią. W ten oto sposób w etapie trenowania obecne były treningi dla danych RGB oraz treningi związane z pojedynczymi kanałami spektralnymi, z kolei w etapie klasyfikacji znalazły się trzy główne podejścia.

W pierwszym podejściu (Metoda 1) zastosowana została klasyfikacja obrazów RGB przy użyciu wybranego algorytmu, spośród puli algorytmów analizowanych podczas badań. Drugie podejście (Metoda 2) polega na klasyfikacji przy użyciu 10 modeli (również wykorzystujących wskazane w badaniu algorytmy) wytrenowanych na poszczególnych pojedynczych kanałach spektralnych - generowanych jest tutaj 10 odpowiedzi odpowiadających poszczególnym modelom wytrenowanym na pojedynczych kanałach spektralnych. W trzecim podejściu (Metoda 3) zastosowana została klasyfikacja z wykorzystaniem klasyfikatora głosującego, łączącego wyniki 10 modeli wytrenowanych na pojedynczych kanałach spektralnych, przy użyciu wybranego algorytmu, w celu uzyskania wspólnej odpowiedzi.

Finalnie wybrana metoda klasyfikacji faz rozwojowych kukurydzy w skali BBCH wiąże się z wyborem najbardziej skutecznego rozwiązania spośród zaproponowanych metod. Przeprowadzone w tym celu badania, realizowane były z wykorzystaniem pełnej puli algorytmów zastosowanych do realizacji testów z wykorzystaniem każdej z opracowanych metod. Pozwoliło to na kompleksowe porównanie uzyskanych wyników i wyłonienie takiej konfiguracji metody i wybranego algorytmu, aby osiągnięte wyniki były jak najlepsze.

W podrozdziałach 4.2. – 4.6. przedstawione zostaną wyniki eksperymentalne dotyczące klasyfikacji faz rozwojowych w skali BBCH. Zaprezentowane zostanie porównanie wyników uzyskanych na danych RGB, wyników uzyskanych na danych wielospektralnych oraz wyników uzyskanych dzięki opracowanej metodzie głosowania z wykorzystaniem danych wielospektralnych.

## 4.2. Wyniki uzyskane dla danych wielospektralnych

Podczas badań eksperymentalnych zauważalne były różnice w wynikach klasyfikacji pomiędzy poszczególnymi algorytmami wykorzystywanymi do procesu trenowania głębokich sieci neuronowych oraz pomiędzy poszczególnymi kanałami spektralnymi, na bazie których prowadzone były treningi z wykorzystaniem tychże algorytmów.

Poniżej przedstawione zostały wyniki klasyfikacji dla badanych kanałów spektralnych zarejestrowane dla każdego z badanych algorytmów oraz wyniki z wykluczeniem kanału 10. Model wytrenowany na podstawie danych z kanału nr 10 wyraźnie odbiega skutecznością klasyfikacji od modeli wytrenowanych na bazie pozostałych kanałów spektralnych. Zaprezentowane tabele ilustrują efektywność poszczególnych metod klasyfikacji na danych wielospektralnych.

| model_name | accuracy |          |          |             | precision |          |          |             | recall   |          |          |             | f1       |          |          |             |
|------------|----------|----------|----------|-------------|-----------|----------|----------|-------------|----------|----------|----------|-------------|----------|----------|----------|-------------|
|            | htc_r101 | htc_r50  | htc_x101 | mask2former | htc_r101  | htc_r50  | htc_x101 | mask2former | htc_r101 | htc_r50  | htc_x101 | mask2former | htc_r101 | htc_r50  | htc_x101 | mask2former |
| canal      |          |          |          |             |           |          |          |             |          |          |          |             |          |          |          |             |
| 01         | 0.486957 | 0.552174 | 0.547826 | 0.539130    | 0.486508  | 0.592528 | 0.582530 | 0.550819    | 0.546268 | 0.608676 | 0.582942 | 0.547557    | 0.482994 | 0.558260 | 0.543616 | 0.506817    |
| 02         | 0.568282 | 0.559471 | 0.559471 | 0.555066    | 0.594116  | 0.563848 | 0.586652 | 0.611719    | 0.619491 | 0.589024 | 0.606347 | 0.599922    | 0.563584 | 0.554428 | 0.564948 | 0.543282    |
| 03         | 0.548246 | 0.550661 | 0.530702 | 0.508772    | 0.587350  | 0.573304 | 0.570517 | 0.567900    | 0.593651 | 0.591402 | 0.566799 | 0.549762    | 0.552418 | 0.557723 | 0.521853 | 0.509900    |
| 04         | 0.567100 | 0.549784 | 0.541126 | 0.510823    | 0.603094  | 0.587433 | 0.563036 | 0.502541    | 0.626295 | 0.600151 | 0.566362 | 0.557919    | 0.566393 | 0.555305 | 0.539768 | 0.494669    |
| 05         | 0.575893 | 0.580357 | 0.558036 | 0.531250    | 0.596516  | 0.605068 | 0.578664 | 0.545954    | 0.621161 | 0.630494 | 0.587758 | 0.571906    | 0.585859 | 0.589184 | 0.552485 | 0.523784    |
| 06         | 0.547085 | 0.551570 | 0.581081 | 0.520179    | 0.570063  | 0.577274 | 0.591761 | 0.545100    | 0.571536 | 0.585762 | 0.610729 | 0.558193    | 0.546741 | 0.556209 | 0.591596 | 0.521969    |
| 07         | 0.582609 | 0.565217 | 0.569565 | 0.495652    | 0.606866  | 0.581716 | 0.593181 | 0.566532    | 0.627316 | 0.587195 | 0.602076 | 0.541441    | 0.587952 | 0.560104 | 0.570215 | 0.474963    |
| 08         | 0.538117 | 0.522321 | 0.540179 | 0.486607    | 0.575040  | 0.556559 | 0.572584 | 0.504101    | 0.561341 | 0.553343 | 0.555533 | 0.498973    | 0.542855 | 0.529349 | 0.552871 | 0.476158    |
| 09         | 0.551570 | 0.569507 | 0.549550 | 0.524664    | 0.586433  | 0.584708 | 0.561682 | 0.540334    | 0.611539 | 0.596285 | 0.565288 | 0.541253    | 0.557820 | 0.570818 | 0.544688 | 0.527224    |
| 10         | 0.219298 | 0.223684 | 0.214912 | 0.232456    | 0.161993  | 0.175143 | 0.162701 | 0.174677    | 0.215904 | 0.228801 | 0.217906 | 0.257787    | 0.175971 | 0.187777 | 0.176843 | 0.185934    |

**Tabela 4.1.** Wyniki klasyfikacji dla każdego z analizowanych kanałów spektralnych z użyciem poszczególnych algorytmów uczenia głębokiego.

| model_name | accuracy |          |          |             | precision |          |          |             | recall   |          |          |             | f1       |          |          |             |
|------------|----------|----------|----------|-------------|-----------|----------|----------|-------------|----------|----------|----------|-------------|----------|----------|----------|-------------|
|            | htc_r101 | htc_r50  | htc_x101 | mask2former | htc_r101  | htc_r50  | htc_x101 | mask2former | htc_r101 | htc_r50  | htc_x101 | mask2former | htc_r101 | htc_r50  | htc_x101 | mask2former |
| canal      |          |          |          |             |           |          |          |             |          |          |          |             |          |          |          |             |
| 01         | 0.486957 | 0.552174 | 0.547826 | 0.539130    | 0.486508  | 0.592528 | 0.582530 | 0.550819    | 0.546268 | 0.608676 | 0.582942 | 0.547557    | 0.482994 | 0.558260 | 0.543616 | 0.506817    |
| 02         | 0.568282 | 0.559471 | 0.559471 | 0.555066    | 0.594116  | 0.563848 | 0.586652 | 0.611719    | 0.619491 | 0.589024 | 0.606347 | 0.599922    | 0.563584 | 0.554428 | 0.564948 | 0.543282    |
| 03         | 0.548246 | 0.550661 | 0.530702 | 0.508772    | 0.587350  | 0.573304 | 0.570517 | 0.567900    | 0.593651 | 0.591402 | 0.566799 | 0.549762    | 0.552418 | 0.557723 | 0.521853 | 0.509900    |
| 04         | 0.567100 | 0.549784 | 0.541126 | 0.510823    | 0.603094  | 0.587433 | 0.563036 | 0.502541    | 0.626295 | 0.600151 | 0.566362 | 0.557919    | 0.566393 | 0.555305 | 0.539768 | 0.494669    |
| 05         | 0.575893 | 0.580357 | 0.558036 | 0.531250    | 0.596516  | 0.605068 | 0.578664 | 0.545954    | 0.621161 | 0.630494 | 0.587758 | 0.571906    | 0.585859 | 0.589184 | 0.552485 | 0.523784    |
| 06         | 0.547085 | 0.551570 | 0.581081 | 0.520179    | 0.570063  | 0.577274 | 0.591761 | 0.545100    | 0.571536 | 0.585762 | 0.610729 | 0.558193    | 0.546741 | 0.556209 | 0.591596 | 0.521969    |
| 07         | 0.582609 | 0.565217 | 0.569565 | 0.495652    | 0.606866  | 0.581716 | 0.593181 | 0.566532    | 0.627316 | 0.587195 | 0.602076 | 0.541441    | 0.587952 | 0.560104 | 0.570215 | 0.474963    |
| 08         | 0.538117 | 0.522321 | 0.540179 | 0.486607    | 0.575040  | 0.556559 | 0.572584 | 0.504101    | 0.561341 | 0.553343 | 0.555533 | 0.498973    | 0.542855 | 0.529349 | 0.552871 | 0.476158    |
| 09         | 0.551570 | 0.569507 | 0.549550 | 0.524664    | 0.586433  | 0.584708 | 0.561682 | 0.540334    | 0.611539 | 0.596285 | 0.565288 | 0.541253    | 0.557820 | 0.570818 | 0.544688 | 0.527224    |

**Tabela 4.2.** Wyniki klasyfikacji dla wybranych kanałów spektralnych z wykluczeniem kanału nr 10.

Analizując powyższe wyniki można wyciągnąć istotne wnioski. Przy wykorzystaniu algorytmu HTC z ResNet101 jako backbone najlepsze wyniki uzyskano dla kanału spektralnego nr 07 (red edge 705 (10)). Przy wykorzystaniu algorytmu HTC z ResNet50 jako backbone najlepsze wyniki uzyskano dla kanału spektralnego nr 05 (red 650 (16)). Przy wykorzystaniu algorytmu HTC z ResNeXt101 jako backbone najlepsze wyniki uzyskano dla kanału spektralnego nr 06 (red 668 (14)). Z kolei przy wykorzystaniu algorytmu Mask2Former najlepsze wyniki uzyskano dla kanału spektralnego nr 02 (blue 475 (32)). Widać zatem, iż dla poszczególnych algorytmów, przy zachowaniu tych samych danych treningowych najlepsze wyniki uzyskiwane są dla różnych kanałów spektralnych.

Analizując zatem wyniki dla poszczególnych kanałów spektralnych oraz wskazując, który z algorytmów dla danego kanału spektralnego pozwalał na osiągnięcie najkorzystniejszych wyników, można zauważyć, że:

- Dla kanału spektralnego nr 01 (coastal blue 444 (28)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmu HTC z ResNet50 jako backbone, zgodnie ze wszystkimi miarami użytymi do oceny wydajności modeli klasyfikacyjnych (Accuracy, Precision, Recall, F1-score).
- Dla kanału spektralnego nr 02 (blue 475 (32)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmów: HTC z ResNet101 jako backbone (według metryk Accuracy i Recall), Mask2Former (według metryki Precision) oraz HTC z ResNeXt101 jako backbone (według metryki F1-score).
- Dla kanału spektralnego nr 03 (green 531 (14)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmów: HTC z ResNet50 jako backbone (według metryk Accuracy i F1-score) oraz HTC z ResNet101 jako backbone (według metryk Precision i Recall).
- Dla kanału spektralnego nr 04 (green 560 (27)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmu HTC z ResNet101 jako backbone, zgodnie ze wszystkimi miarami użytymi do oceny wydajności modeli klasyfikacyjnych (Accuracy, Precision, F1-score).
- Dla kanału spektralnego nr 05 (red 650 (16)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmu HTC z ResNet50 jako backbone, zgodnie ze wszystkimi

miarami użytymi do oceny wydajności modeli klasyfikacyjnych (Accuracy, Precision, Recall, F1-score).

- Dla kanału spektralnego nr 06 (red 668 (14)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmu HTC z ResNeXt101 jako backbone, zgodnie ze wszystkimi miarami użytymi do oceny wydajności modeli klasyfikacyjnych (Accuracy, Precision, Recall, F1-score).
- Dla kanału spektralnego nr 07 (red edge 705 (10)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmu HTC z ResNet101 jako backbone, zgodnie ze wszystkimi miarami użytymi do oceny wydajności modeli klasyfikacyjnych (Accuracy, Precision, Recall, F1-score).
- Dla kanału spektralnego nr 08 (red edge 717 (12)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmów: HTC z ResNet101 jako backbone (według metryk Precision i Recall) oraz HTC z ResNeXt101 jako backbone (według metryk Accuracy i F1-score).
- Dla kanału spektralnego nr 09 (red edge 740 (18)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmów: HTC z ResNet50 jako backbone (według metryk Accuracy i F1-score) oraz HTC z ResNet101 jako backbone (według metryk Precision i Recall).
- Dla kanału spektralnego nr 10 (NIR 842 (57)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmów: Mask2Former jako backbone (według metryk Accuracy i Recall) oraz HTC z ResNet50 jako backbone (według metryk Precision i F1-score).

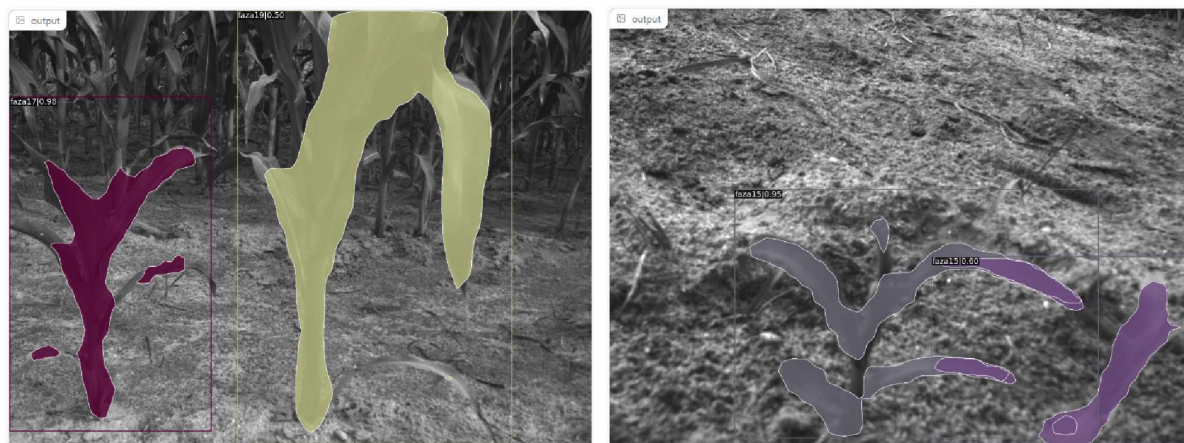
W Tabeli 4.3. przedstawiono wyniki klasyfikacji dla każdego badanego algorytmu. Tabela ta ilustruje dokładność poszczególnych metod klasyfikacyjnych na danych RGB w oparciu o podstawowe metryki takie jak: Accuracy, Precision, Recall oraz F1-score.

| Spectral Channel  | Wavelength (nm) | Best Algorithm   | Metrics                               |
|-------------------|-----------------|------------------|---------------------------------------|
| 01 (coastal blue) | 444 (28)        | HTC (ResNet50)   | Accuracy, Precision, Recall, F1-score |
| 02 (blue)         | 475 (32)        | HTC (ResNet101)  | Accuracy, Recall                      |
|                   |                 | Mask2Former      | Precision                             |
|                   |                 | HTC (ResNeXt101) | F1-score                              |
| 03 (green)        | 531 (14)        | HTC (ResNet50)   | Accuracy, F1-score                    |
|                   |                 | HTC (ResNet101)  | Precision, Recall                     |
| 04 (green)        | 560 (27)        | HTC (ResNet101)  | Accuracy, Precision, Recall, F1-score |
| 05 (red)          | 650 (16)        | HTC (ResNet50)   | Accuracy, Precision, Recall, F1-score |
| 06 (red)          | 668 (14)        | HTC (ResNeXt101) | Accuracy, Precision, Recall, F1-score |
| 07 (red edge)     | 705 (10)        | HTC (ResNet101)  | Accuracy, Precision, Recall, F1-score |
| 08 (red edge)     | 717 (12)        | HTC (ResNet101)  | Precision, Recall                     |
|                   |                 | HTC (ResNeXt101) | Accuracy, F1-score                    |
| 09 (red edge)     | 740 (18)        | HTC (ResNet50)   | Accuracy, F1-score                    |
|                   |                 | HTC (ResNet101)  | Precision, Recall                     |
| 10 (NIR)          | 842 (57)        | Mask2Former      | Accuracy, Recall                      |
|                   |                 | HTC (ResNet50)   | Precision, F1-score                   |

**Tabela 4.3.** Wyniki klasyfikacji dla każdego badanego algorytmu.

Wyniki klasyfikacji przeprowadzone na kanale spektralnym nr 10 (NIR 842 (57)) znacznie odbiegają pod kątem poprawności klasyfikacji od wyników dla pozostałych kanałów. To daje jasny wniosek, iż klasyfikacja przeprowadzona tylko i wyłącznie na tym kanale spektralnym wiąże się z najniższą skutecznością klasyfikacji obiektów spośród analizowanych pasm spektralnych.

Poniżej przedstawiono kilka dodatkowych obrazów, na których widać wyniki działania głębokiej sieci neuronowej opracowanej w ramach zadania. Widać, że sieć w sposób całkowicie poprawny wykryła obiekty w postaci kukurydzy i prawidłowo zaklasyfikowała fazy rozwojowe w skali BBCH.



**Rysunek 4.2.** Wynik predykcji modelu opartego o uczenie na bazie obrazów wielospektralnych.



### 4.3. Wyniki uzyskane dla danych RGB

Oprócz badań z wykorzystaniem obrazów wielospktralnych przeprowadzone zostały również badania z użyciem obrazów RGB. Podczas prowadzonych eksperymentów zaobserwować można było różnice w wynikach klasyfikacji pomiędzy poszczególnymi algorytmami wykorzystywanymi do trenowania głębokich sieci neuronowych w oparciu o obrazy RGB na bazie których prowadzone były treningi.

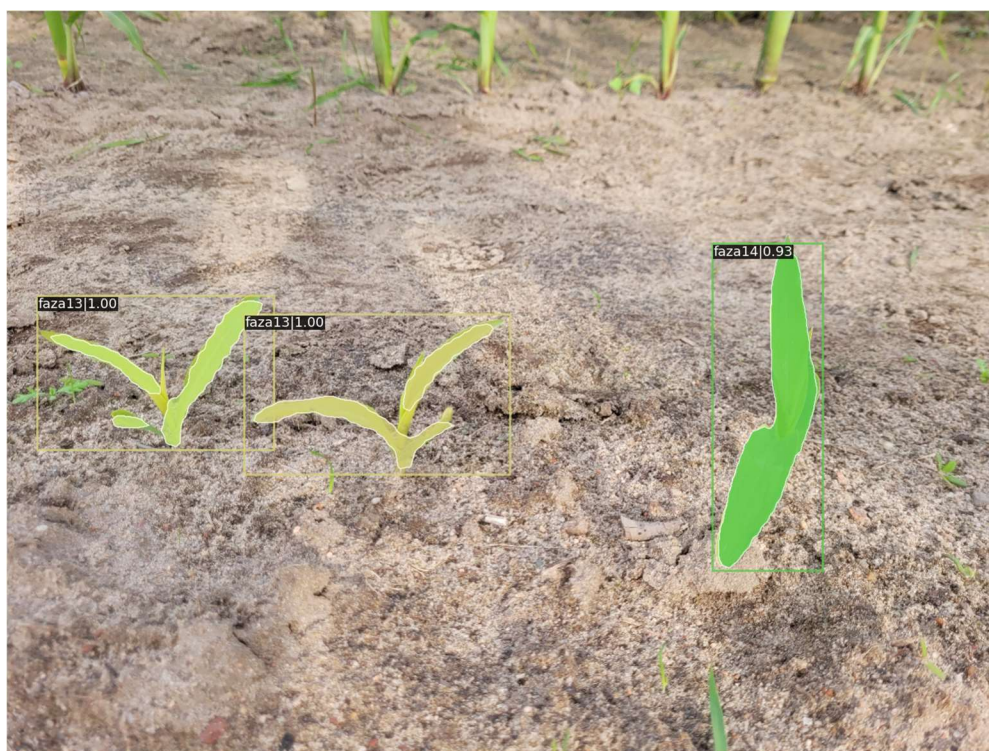
Najlepsze wyniki klasyfikacji obrazów RGB uzyskał algorytm HTC z ResNeXt101 jako backbone. Fakt ten pozwolił wyciągnąć istotny wniosek badawczy oraz projektowy, wskazując, że do procesu klasyfikacji obrazów RGB spośród badanych modeli najlepiej stosować model wytrenowany z użyciem tego algorytmu.

Na Rysunku 4.3. widać wyniki pierwszych eksperymentów, które prowadzone były na nieco mniejszym zbiorze treningowym i przy mniejszej liczbie testowanych algorytmów. Algorytm już wtedy poprawnie wykrywał kukurydzę i klasyfikował jej fazę rozwojową, lecz obwiednia będąca oznaczeniem wykrytych obiektów nie była jeszcze idealnym odzwierciedleniem roślin.



**Rysunek 4.3.** Wynik predykcji modelu – pierwsze eksperymenty.

Po przeprowadzeniu dalszych treningów, po rozszerzeniu zbioru treningowego oraz po przeanalizowaniu większej ilości algorytmów opartych o głębokie sieci neuronowe uzyskano poprawę wyników, a przetestowanie wielu różnego rodzaju obrazów kukurydzy pokazało, że opracowane rozwiązanie poprawnie wykrywa kukurydzę oraz poprawnie klasyfikuje wykryte na obrazach obiekty i przypisuje im klasy zgodnie ze zbiorem: *faza10*, *faza11*, *faza12*, *faza13*, *faza14*, *faza15*, *faza16*, *faza17*, *faza18*, *faza19*. Na Rysunku 4.4. zaprezentowano wynik predykcji modelu wytrenowanego na większej liczbie obrazów. Kukurydza została prawidłowo wykryta, algorytm prawidłowo zaklasyfikował ją, a wykryty obszar praktycznie idealnie pokrywa się z obszarem obrazu, na którym jest kukurydza. Jak widać najlepsze wyniki uzyskano przy wykorzystaniu algorytmu Hybrid Task Cascade (HTC) z ResNeXt101 jako backbone.



**Rysunek 4.4.** Wynik predykcji modelu.

Poniżej przedstawiono kilka dodatkowych obrazów, na których widać wyniki działania głębokiej sieci neuronowej opracowanej w ramach zadania.





**Rysunek 4.5.** Wynik predykcji modelu wykrywania kukurydzy i klasyfikacji jej faz rozwojowych.



**Rysunek 4.6.** Wynik predykcji modelu wykrywania kukurydzy i klasyfikacji jej faz rozwojowych.





**Rysunek 4.7.** Wynik predykcji modelu wykrywania kukurydzy i klasyfikacji jej faz rozwojowych.

#### 4.4. Wyniki uzyskane przy zastosowaniu klasyfikatora głosującego

W Tabeli 4.4. zebrane zostały wyniki dotyczące dokładności klasyfikacji dla wszystkich badanych algorytmów uczenia maszynowego, z podziałem na model wytrenowany wyłącznie na danych RGB oraz model oparty na zaproponowanej metodzie głosowania zestawu modeli wytrenowanych na poszczególnych kanałach spektralnych. Do oceny skuteczności poszczególnych metod posłużyły metryki oceny: Accuracy, Precision, Recall oraz F1-score. Dodatkowo, w przypadku danych RGB użyto do badań zarówno zbioru analogicznego ze zbiorem obrazów wielospektralnych pod względem liczności obiektów w poszczególnych klasach oraz zbioru powiększonego, czyli takiego, w którym ilość oznaczonych obiektów w danych klasach jest większa niż w każdym z pojedynczych kanałów spektralnych.

Analizując poszczególne wyniki, można wyciągnąć istotne wnioski. W przypadku algorytmu HTC z ResNet101 jako backbone, model wytrenowany na podstawie zbioru obrazów RGB okazał się lepszy od zaproponowanej metody głosowania pojedynczych modeli wytrenowanych na podstawie obrazów z pojedynczych kanałów spektralnych pod względem metryki Accuracy. Metryki Precision, Recall, F1-score wskazują natomiast na lepsze wyniki w przypadku klasyfikatora głosującego. Wyniki dla powiększonego zbioru obrazów RGB wskazują natomiast, że model wytrenowany na tym zbiorze jest lepszy od modelu

wytrenowanego na mniejszym zbiorze obrazów RGB oraz osiąga również lepsze wyniki niż zastosowany klasyfikator głosujący. Wniosek ten jest poparty wszystkimi analizowanymi metrykami.

|             |                     | accuracy | precision | recall   | f1       |
|-------------|---------------------|----------|-----------|----------|----------|
| model_name  | canal               |          |           |          |          |
| htc_r101    | all_spectral_canals | 0.651466 | 0.684678  | 0.686132 | 0.643477 |
|             | rgb                 | 0.706667 | 0.460286  | 0.496550 | 0.465737 |
|             | rgb_bigger_dataset  | 0.707143 | 0.737807  | 0.696962 | 0.675608 |
| htc_r50     | all_spectral_canals | 0.661238 | 0.690290  | 0.699036 | 0.659793 |
|             | rgb                 | 0.720000 | 0.611607  | 0.708640 | 0.629196 |
|             | rgb_bigger_dataset  | 0.678571 | 0.712847  | 0.678563 | 0.657851 |
| htc_x101    | all_spectral_canals | 0.657980 | 0.663664  | 0.676181 | 0.652508 |
|             | rgb                 | 0.760000 | 0.525599  | 0.580694 | 0.524339 |
|             | rgb_bigger_dataset  | 0.714286 | 0.746534  | 0.714740 | 0.699089 |
| mask2former | all_spectral_canals | 0.625407 | 0.615539  | 0.630647 | 0.592380 |
|             | rgb                 | 0.800000 | 0.596212  | 0.660516 | 0.600132 |
|             | rgb_bigger_dataset  | 0.528571 | 0.583843  | 0.524615 | 0.468124 |

**Tabela 4.4.** Porównanie wyników klasyfikacji osiąganych z użyciem poszczególnych modeli wytrenowanych na zbiorze obrazów RGB z wynikami klasyfikacji osiąganymi z użyciem tychże modeli wytrenowanych na powiększonym zbiorze obrazów RGB oraz z wynikami klasyfikacji osiąganymi przez te modele po ich wytrenowaniu z użyciem obrazów wielospektralnych oraz zastosowaniu opracowanej metody klasyfikatora głosującego. Do analizy wykorzystano metryki: Accuracy, Precision, Recall oraz F1-score.

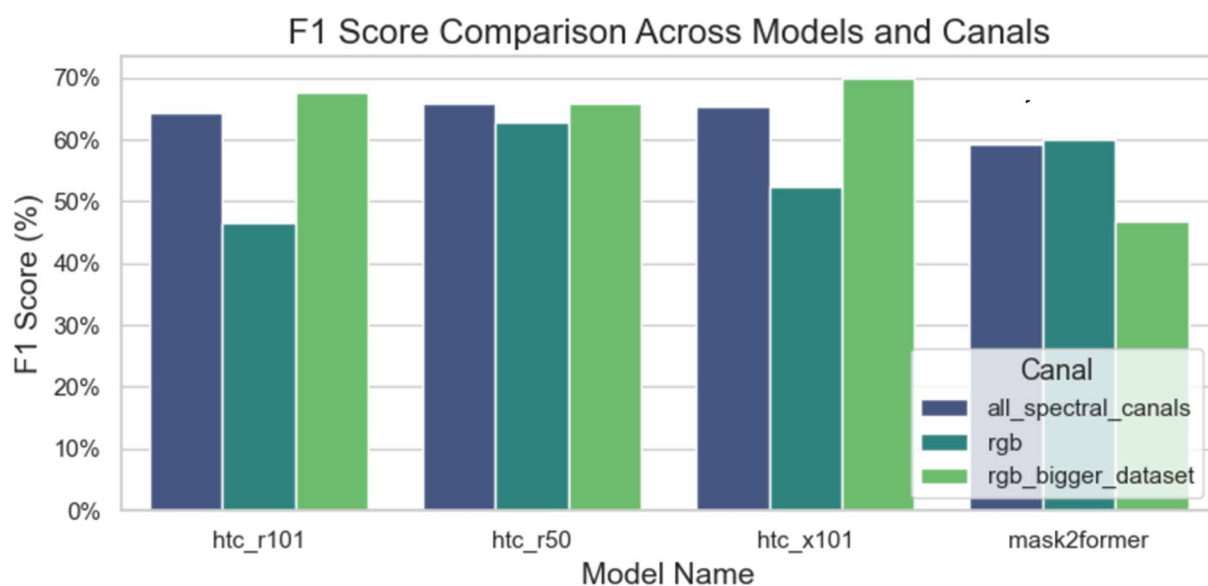
W przypadku algorytmu HTC z ResNet50 jako backbone, model wytrenowany na podstawie zbioru obrazów RGB okazał się lepszy od zaproponowanej metody głosowania opartej na modelach wytrenowanych na obrazach z pojedynczych kanałów spektralnych jedynie na podstawie metryk Accuracy i Recall. Natomiast według metryk Precision i F1-score, to zaproponowana metoda głosowania pozwala uzyskać lepsze wyniki. Wyniki dla powiększonego zbioru obrazów RGB wskazują natomiast, że model wytrenowany na tym

zbiorze osiąga wyższe wartości metryk Precision i F1-score niż model wytrenowanego na mniejszym zbiorze obrazów RGB oraz osiąga wyższe wartości metryk Accuracy i Precision niż zastosowany klasyfikator głosujący.

W przypadku algorytmu HTC z ResNeXt101 jako backbone, model wytrenowany na podstawie zbioru obrazów RGB tylko pod względem metryki Accuracy okazał się lepszy od klasyfikatora głosującego, natomiast metryki Precision, Recall i F1-score pokazały, że jest on gorszy od zaproponowanej przez nas metody głosowania pojedynczych modeli wytrenowanych na podstawie obrazów z pojedynczych kanałów spektralnych. Wyniki uzyskane dla powiększonego zbioru obrazów RGB wskazują zaś, iż model wytrenowany na tym zbiorze osiąga wyższe wartości metryk Accuracy, Precision, Recall i F1-score niż model wytrenowany na mniejszym zbiorze obrazów RGB oraz osiąga wyższe wartości metryk Precision, Recall i F1-score, niż zastosowany klasyfikator głosujący.

W przypadku algorytmu Mask2Former, zaproponowana metoda głosowania pojedynczych modeli wytrenowanych na obrazach z pojedynczych kanałów spektralnych okazała się bardziej skuteczna, niż model wytrenowany na powiększonym zbiorze obrazów RGB. Wniosek ten jest poparty wszystkimi analizowanymi metrykami. W przypadku analogicznego, mniejszego zbioru obrazów RGB, to model wytrenowany na tym zbiorze osiągnął wyniki lepsze niż klasyfikator głosujący według metryk Accuracy, Recall i F1-score, zaś według metryki Precision osiągnął gorsze wyniki niż zastosowana metoda klasyfikatora głosującego.

Rysunek 4.8. przedstawia wykres podsumowujący wyniki dla modelu wytrenowanego na obrazach RGB oraz metody z użyciem klasyfikatora głosującego, w której modele wytrenowane na pojedynczych kanałach spektralnych głosują nad wynikiem klasyfikacji. Do porównania użyta została metryka F1-score.



**Rysunek 4.8.** Wizualizacja porównująca wyniki klasyfikacji poszczególnych algorytmów na obrazach RGB z wynikami uzyskanymi przez te algorytmy z użyciem obrazów wielospktralnych i zastosowaniem klasyfikatora głosującego.

Jak widać na bazie z wykresu, model wytrenowany na obrazach RGB oraz metoda z użyciem klasyfikatora głosującego, w którym modele wytrenowane na pojedynczych kanałach spektralnych głosują nad wynikiem klasyfikacji są porównywalne w przypadku algorytmu HTC z ResNet50 jako backbone. W przypadku algorytmów HTC z ResNet101 oraz ResNeXt101 jako backbone, model wytrenowany na obrazach RGB okazał się lepszy od metody z użyciem klasyfikatora głosującego. W przypadku algorytmu Mask2Former metoda z użyciem klasyfikatora głosującego okazała się zdecydowanie lepsza, niż model wytrenowany na obrazach RGB.

W przypadku obrazów RGB i pojedynczego klasyfikatora uzyskano 89 wyników w postaci poligonów i 89 wyników w postaci predykcji, natomiast dla klasyfikatora głosującego uzyskano 157 wyników w postaci poligonów i 104 wyniki w postaci predykcji. Różnica w liczbie wyników w postaci poligonów i liczbie wyników w postaci predykcji dla klasyfikatora głosującego opartego o analizę obrazów z pojedynczych kanałów spektralnych może wynikać z tego, że dla niektórych obrazów model nie był w stanie wykryć roślin z powodu słabej widoczności w pojedynczych kanałach spektralnych.

Na podstawie pełnej analizy z użyciem poszczególnych algorytmów, po wyliczeniu dokładności w postaci parametru accuracy, uzyskano następujące wyniki:

- w przypadku algorytmu HTC\_x101 dla obrazów RGB i pojedynczego klasyfikatora uzyskano dokładność na poziomie 71,4 %, zaś dla klasyfikatora głosującego opartego o analizę obrazów z pojedynczych kanałów spektralnych uzyskano dokładność na poziomie 65.8%
- w przypadku algorytmu HTC\_r101 dla obrazów RGB i pojedynczego klasyfikatora uzyskano dokładność na poziomie 70,7 %, zaś dla klasyfikatora głosującego opartego o analizę obrazów z pojedynczych kanałów spektralnych uzyskano dokładność na poziomie 65.1%
- w przypadku algorytmu HTC\_r50 dla obrazów RGB i pojedynczego klasyfikatora uzyskano dokładność na poziomie 67,9 %, zaś dla klasyfikatora głosującego opartego o analizę obrazów z pojedynczych kanałów spektralnych uzyskano dokładność na poziomie 66,1 %
- w przypadku algorytmu Mask2Former dla obrazów RGB i pojedynczego klasyfikatora uzyskano dokładność na poziomie 52,9 %, zaś dla klasyfikatora głosującego opartego o analizę obrazów z pojedynczych kanałów spektralnych uzyskano dokładność na poziomie 62.5%.

Na podstawie tychże wyników można wyciągnąć ważne wnioski. W przypadku zastosowania architektur HTC klasyfikator głosujący oparty o analizę obrazów z pojedynczych kanałów spektralnych uzyskał gorszy wynik, niż pojedynczy klasyfikator oparty o analizę obrazów RGB, przy czym w przypadku algorytmu HTC z ResNet50 jako backbone różnica w wynikach była minimalna. Z kolei, przy zastosowaniu architektury Mask2Former sytuacja jest odwrotna i to klasyfikator głosujący poradził sobie lepiej, niż pojedynczy klasyfikator oparty o analizę obrazów RGB, a różnica ta była wyraźna. Patrząc jednak na wynik globalny, najwyższą jakość wyników uzyskano przy zastosowaniu obrazowania RGB i użyciu algorytmu HTC\_x101.

Pojedyncze modele oparte na danych z pojedynczych kanałów spektralnych mają generalnie gorszą jakość, ponieważ zostały wytrenowane na jednokanałowych obrazach, podczas gdy klasyfikator RGB został wytrenowany na obrazach RGB, które są trójkanałowe. Zastosowanie techniki głosowania często poprawia jakość sumarycznej predykcji, lecz nie w tym przypadku ta poprawa nie była na tyle duża, aby przewyższyć wyniki uzyskane z użyciem obrazowania RGB. Problem w danych polega na tym, że obrazy dla różnych kanałów spektralnych robione



były z różnych kątów. Dla pojedynczych modeli mogło to spowodować znaczne różnice w klasyfikacji, które nie zostały skorygowane przez zastosowanie techniki klasyfikatora głosującego.

Warto w tym momencie zaproponować dalsze kroki postępowania. Aby zaadresować problem, który leży u podłoża niskich wyników uzyskanych przy użyciu klasyfikatora głosującego, należy wykonać poprawiony dataset, co można osiągnąć na dwa sposoby:

- pozyskiwanie obrazów z wykorzystaniem obrotowej kamery wielospektralnej lub kamery spektralnej jednooczkowej, aby każdy obraz było wykonane dokładnie pod tym samym kątem,
- pozyskiwanie obrazów z większej odległości oraz zoomowanie obrazów, aby różnice w różnym kącie patrzenia poszczególnych kamer były jak najmniejsze.

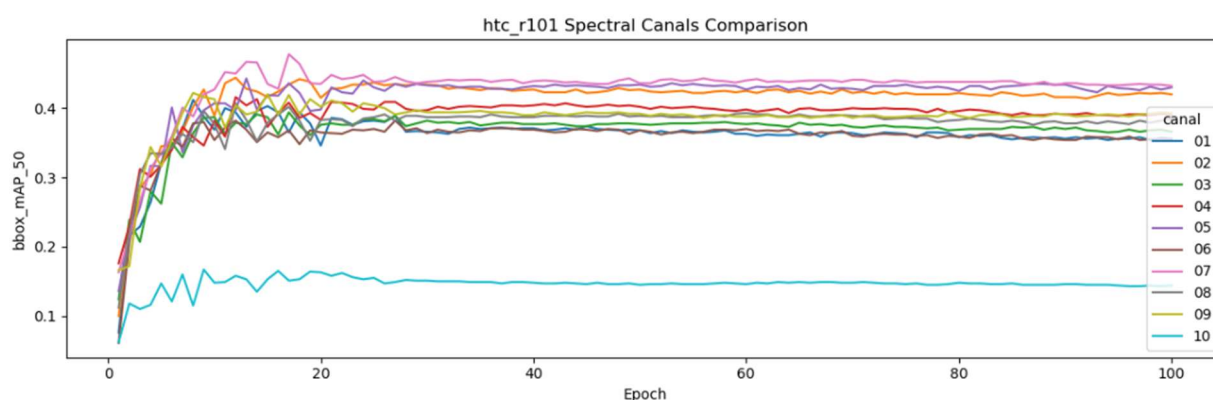
W dalszych badaniach zajęto się pierwszym z tych podejść, aby sprawdzić jak wpłynie to na poprawę wyników klasyfikacji. Uzyskane wyniki zaprezentowano w rozdziale 5. Detekcja i klasyfikacja faz rozwoju kukurydzy w skali BBCH przy wykorzystaniu obrazowania RGB i kamery wielospektralnej MapIR. Co prawda wykorzystywana kamera wielospektralna ma inne parametry niż kamera MicaSense RedEdge MX-Dual, lecz nadal możliwa jest konfrontacja rozwiązania uzyskanego z użyciem obrazowania RGB i wielospektralnego.

#### 4.5. Krzywe uczenia dla danych wielospektralnych i RGB

W niniejszym rozdziale zaprezentowane zostały krzywe uczenia zarejestrowane podczas trenowania poszczególnych modeli. Poniżej zamieszczono krzywe uczenia dla poszczególnych algorytmów, przedstawiające wyniki przebiegu uczenia zarejestrowane dla różnych kanałów spektralnych.

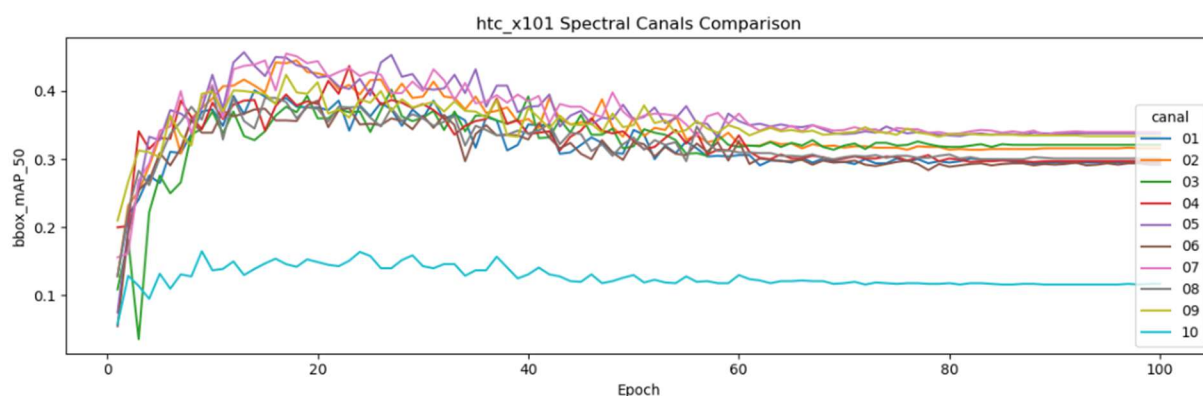
Rysunek 4.9. przedstawia porównanie wyników modelu HTC\_r101 trenowanego na poszczególnych kanałach spektralnych w trakcie treningu, który trwał 100 epok. Oś pozioma reprezentuje epoki, a oś pionowa przedstawia miarę dokładności wykrywania obiektów. Wyniki dla kanału nr 10 są najniższe w porównaniu do innych kanałów, gdyż wartość bbox\_mAP\_50 oscyluje wokół 0.1 po pierwszych 20 epokach i utrzymuje się na tym poziomie do końca całego zakresu. Pozostałe kanały wykazują znacznie lepsze wyniki. Wartości bbox\_mAP\_50 dla tych kanałów oscylują wokół 0.4 po pierwszych 20 epokach i utrzymują się

na tym poziomie do końca obserwowanego zakresu, z niewielkimi wahaniami, co wskazuje na lepszą efektywność w wykrywaniu obiektów. Najlepszą skutecznością charakteryzuje się model trenowany na podstawie danych z kanału nr 7. Osiąga on maksymalną spośród wszystkich kanałów wartość bbox\_mAP\_50 w granicach około 0.47, po czym lekko spada, utrzymując się na najwyższej pozycji spośród wszystkich modeli wytrenowanych na poszczególnych kanałach spektralnych.



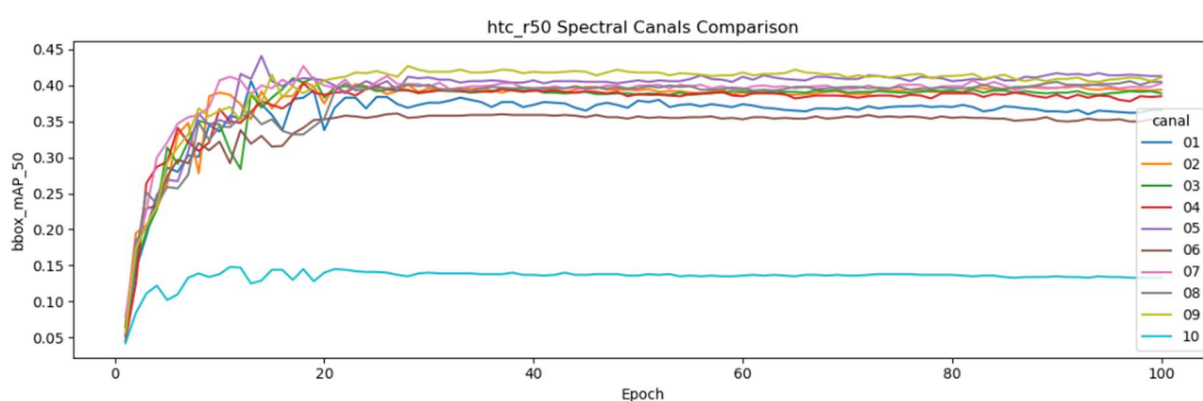
**Rysunek 4.9.** Porównanie wyników modelu HTC\_r101 trenowanego na poszczególnych kanałach spektralnych.

Warto również przyjrzeć się rysunkowi 4.10. zawierającemu krzywe uczenia modelu opartego o algorytm HTC\_x101 dla poszczególnych kanałów spektralnych. Również w tym przypadku wyniki dla kanału nr 10 są najniższe w porównaniu do innych kanałów. Wartości bbox\_mAP\_50 dla tych kanałów oscylują wokół 0.4 po pierwszych 15-40 epokach, po czym delikatnie spadają do poziomu 0.35 w zakresie 40-60 epok i utrzymują się na poziomie ponad 0.3 w dalszym etapie trenowania. Najlepsze wyniki można zaobserwować dla kanału nr 05 przy bardzo zbliżonych wynikach również dla kanału nr 07.



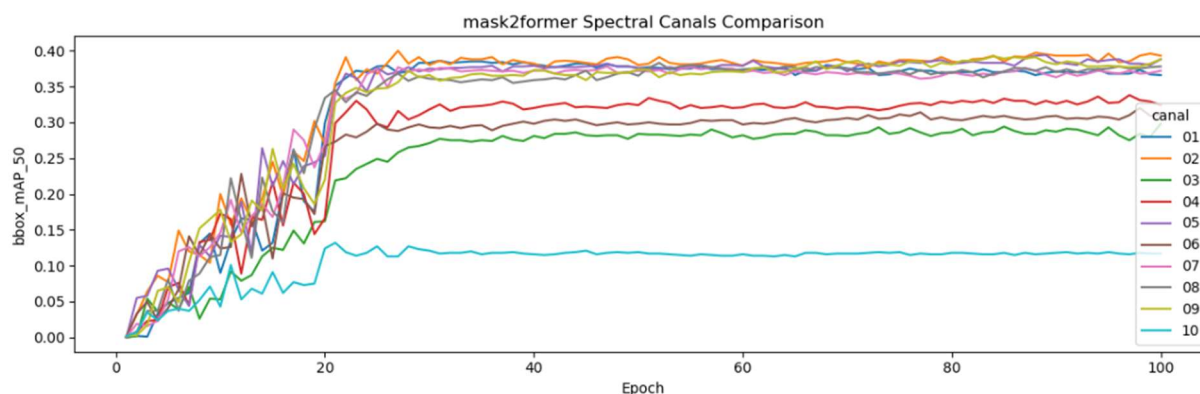
**Rysunek 4.10.** Porównanie wyników modelu HTC\_x101 trenowanego na poszczególnych kanałach spektralnych.

W przypadku krzywych uczenia dla algorytmu HTC\_r50 zaobserwowane wyniki są podobne do krzywych uczenia dla algorytmu HTC\_r101. W tym przypadku również wyniki dla kanału nr 10 są najniższe w porównaniu do innych kanałów. Pozostałe modele trenowane na innych kanałach wykazują znacznie lepsze wyniki. Wartości bbox\_mAP\_50 dla tych kanałów oscylują wokół 0.4 po pierwszych 10 epokach i utrzymują się na tym poziomie, z niewielkimi wahaniami, do końca zakresu obserwacji, czyli 100 epok. Wartości dla kanałów 01-09 są bardziej stabilne po osiągnięciu maksimum, co wskazuje na ich lepszą efektywność w wykrywaniu obiektów za pomocą modelu HTC\_r50.



**Rysunek 4.11.** Porównanie wyników modelu HTC\_r50 trenowanego na poszczególnych kanałach spektralnych.

Dla algorytmu Mask2Former również można zaobserwować najniższe wyniki dla kanału nr 10, gdzie bbox\_mAP\_50 oscyluje wokół 0.125 po pierwszych 20 epokach i utrzymuje się na tym poziomie do końca obserwacji, czyli do końca 100 epok. Pozostałe kanały osiągają wyższe wartości bbox\_mAP\_50, lecz nie stanowią już tak zwartej grupy jak w przypadku poprzednich algorytmów. Spośród wszystkich kanałów najwyższe wartości bbox\_mAP osiąga kanał nr 02.



**Rysunek 4.12.** Porównanie wyników modelu Mask2Former trenowanego na poszczególnych kanałach spektralnych.

Przedstawione wyniki pokazują, iż dla różnych algorytmów zupełnie różne kanały spektralne osiągają najwyższe wartości bbox\_mAP\_50, zdecydowana większość stabilizuje się po 20 epokach (z wyjątkiem algorytmu HTC\_x101, który osiąga stabilizację po 60 epokach), a we wszystkich przypadkach użytych algorytmów kanał nr 10 daje najgorsze wyniki uczenia.

Warto prześledzić również krzywe uczenia dla poszczególnych kanałów. Zaczniemy od modeli trenowanych na obrazach RGB. Wykres 6 przedstawia porównanie wyników czterech modeli na zbiorze danych RGB w ciągu 100 epok. Oś pozioma reprezentuje epoki, a oś pionowa przedstawia wartość bbox\_mAP\_50, czyli miarę dokładności wykrywania obiektów.

Model HTC\_r101 osiąga dobre wyniki już od początku treningu, a wartość bbox\_mAP\_50 szybko rośnie osiągając poziom około 0.8 już po około 20 epokach i stabilizuje się na tym poziomie do końca treningu.

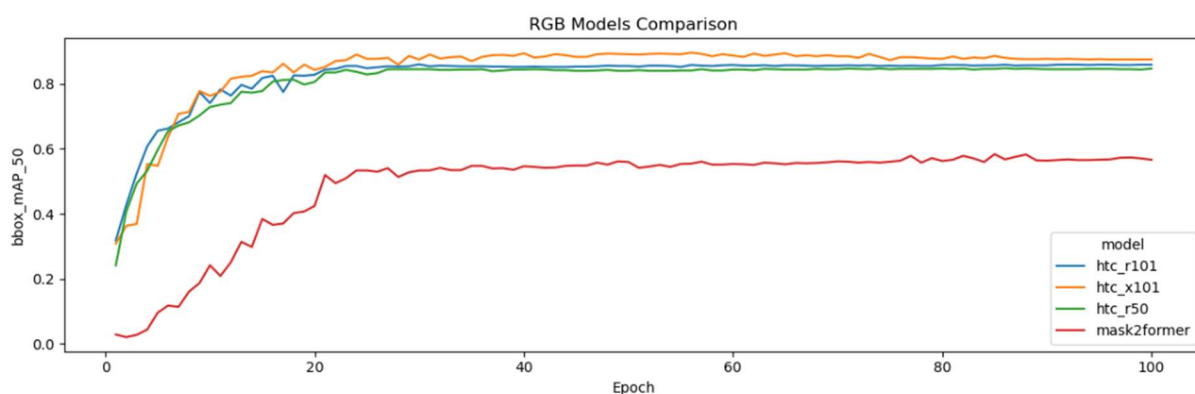
Model wytrenowany na podstawie algorytmu HTC\_x101 osiąga najlepsze wyniki spośród wszystkich porównywanych modeli. Wartości bbox\_mAP\_50 również szybko rosną, osiągając

poziom nieco powyżej 0.8 po około 20 epokach i utrzymując się na tym poziomie do końca treningu.

Model wytrenowany na bazie algorytmu HTC\_r50 osiąga wyniki zbliżone do HTC\_r101 i HTC\_x101. Wartości bbox\_mAP\_50 szybko rosną do poziomu około 0.8 po około 20 epokach i utrzymują się na tym poziomie do końca treningu.

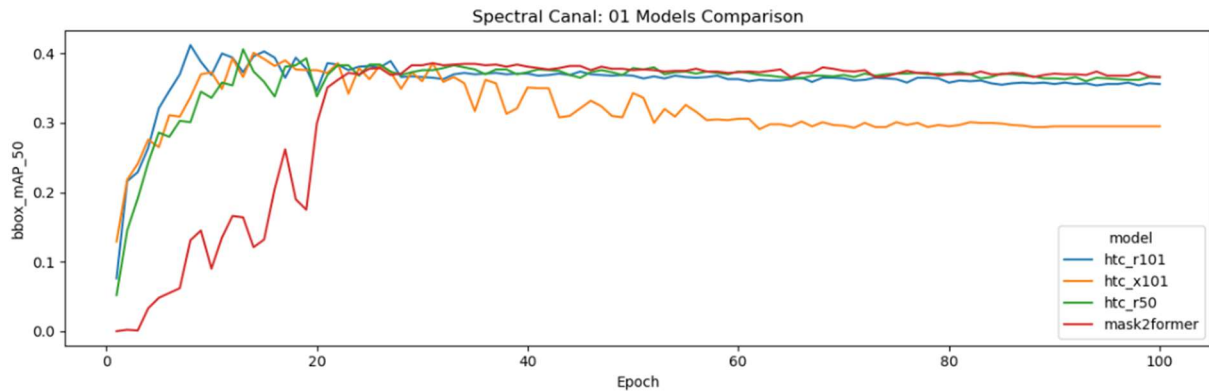
Model Mask2Former osiąga zaś najniższe wyniki spośród porównywanych modeli. Wartości bbox\_mAP\_50 rosną wolniej, osiągając poziom około 0.6 po około 60 epokach i utrzymują się na tym poziomie do końca treningu, wykazując większe wahania w porównaniu do innych modeli.

Można wywnioskować zatem, iż modele HTC\_r101, HTC\_x101 i HTC\_r50 są bardziej efektywne w wykrywaniu obiektów na zbiorze RGB niż model Mask2Former.



**Rysunek 4.13.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych RGB.

Można teraz przejść do modeli trenowanych na obrazach z pojedynczych kanałów spektralnych. Poniższe wykresy przedstawiają porównanie wyników czterech modeli na zbiorze danych odpowiadającym dwóm wybranym kanałom spektralnym w ciągu 100 epok. Pozostałe wykresy dla innych kanałów spektralnych dostępne są w repozytorium [wstawię link]. Na wszystkich wykresach oś pozioma reprezentuje poszczególne epoki, a oś pionowa przedstawia wartość bbox\_mAP\_50, czyli miarę dokładności wykrywania obiektów.

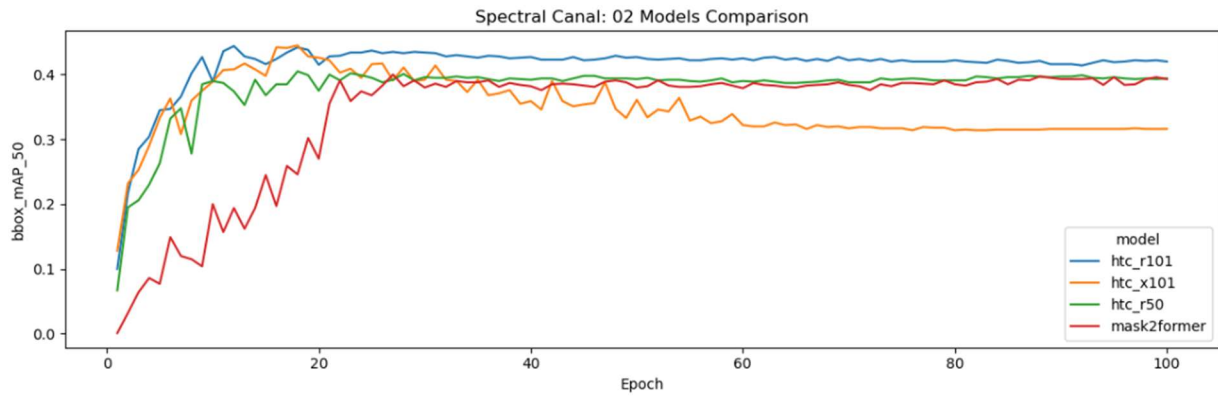


**Rysunek 4.14.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z pierwszego kanału spektralnego.

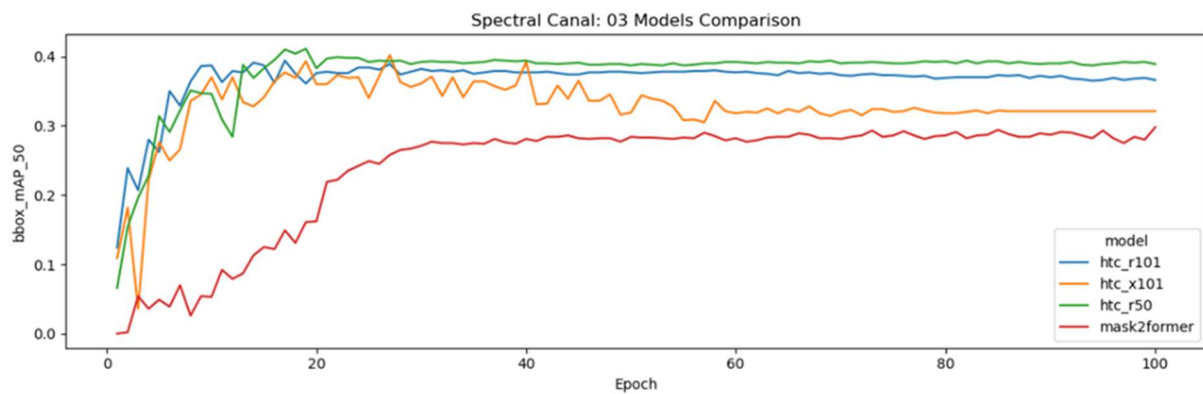
Rysunek 4.14 przedstawia wyniki dla poszczególnych algorytmów zarejestrowane dla kanału spektralnego nr 01. Model Mask2Former po 20 epokach osiąga najwyższe i stabilne wyniki, z wartością `bbox_mAP_50` około 0.4. Modele HTC\_r101 i HTC\_r50 osiągają wysokie i stabilne wyniki z wartościami `bbox_mAP_50` około 0.4, ale są nieco niższe niż wyniki Mask2Former po 20 epokach. Model HTC\_x101 początkowo osiąga dobre wyniki, ale po 20 epokach następuje spadek i stabilizacja na poziomie około 0.3, co wskazuje na niższą efektywność w porównaniu do pozostałych modeli.

Analizując wyniki dla kanału spektralnego nr 02 (Rysunek 4.15) można zauważyć, że wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie `bbox_mAP_50` równym 0.4. Wykres dla algorytmu Mask2Former ma szybką tendencję wzrostową w pierwszych 20 epokach, po czym również stabilizuje się dla wartości `bbox_mAP` na poziomie około 0.4. Z kolei wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 20 epok osiągając poziom `bbox_mAP_50` około 0.4, po czym spada i stabilizuje się na poziomie `bbox_mAP_50` równym około 0.3 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów osiąga model oparty na HTC\_r101, a najniższe model oparty o algorytm HTC\_x101.



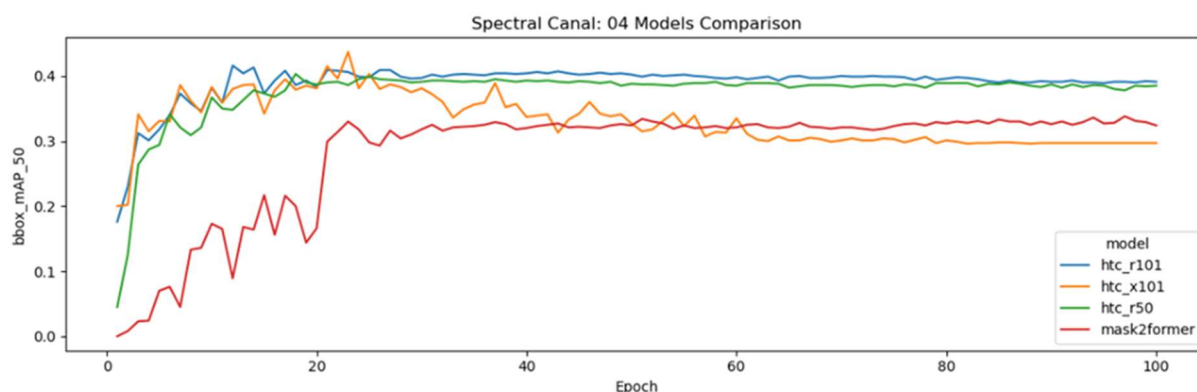


**Rysunek 4.15.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z drugiego kanału spektralnego.



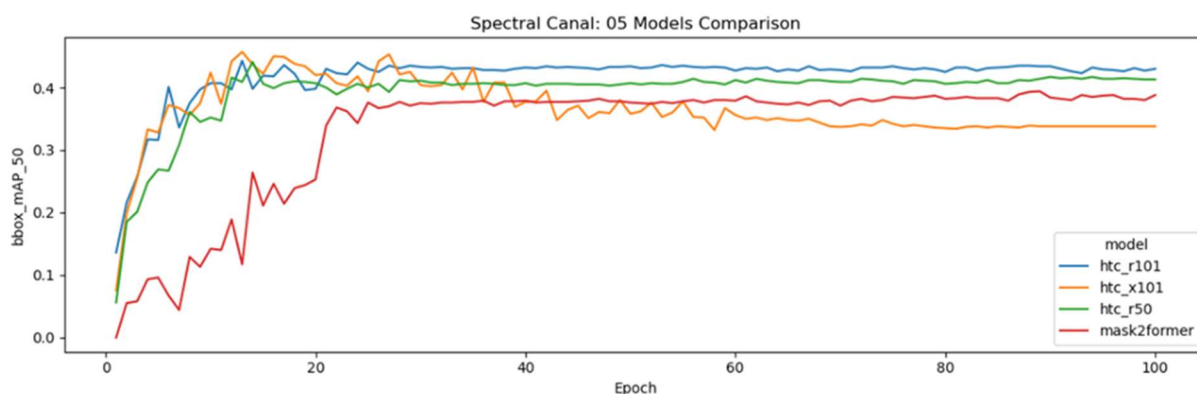
**Rysunek 4.16.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z trzeciego kanału spektralnego.

W wynikach dla kanału spektralnego nr 03 (Rysunek 4.16.) również można zauważyć, że wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox\_mAP\_50 równym 0.4. Wykres dla algorytmu Mask2Former ma dosyć szybką tendencję wzrostową w pierwszych 20 epokach, po czym stabilizuje się dla wartości bbox\_mAP na poziomie około 0.3. Z kolei wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 10 epok osiągając poziom bbox\_mAP\_50 około 0.4, po czym spada i stabilizuje się na poziomie bbox\_mAP\_50 równym około 0.3 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów osiąga model oparty na HTC\_r50, a najniższe model oparty o algorytm Mask2Former.



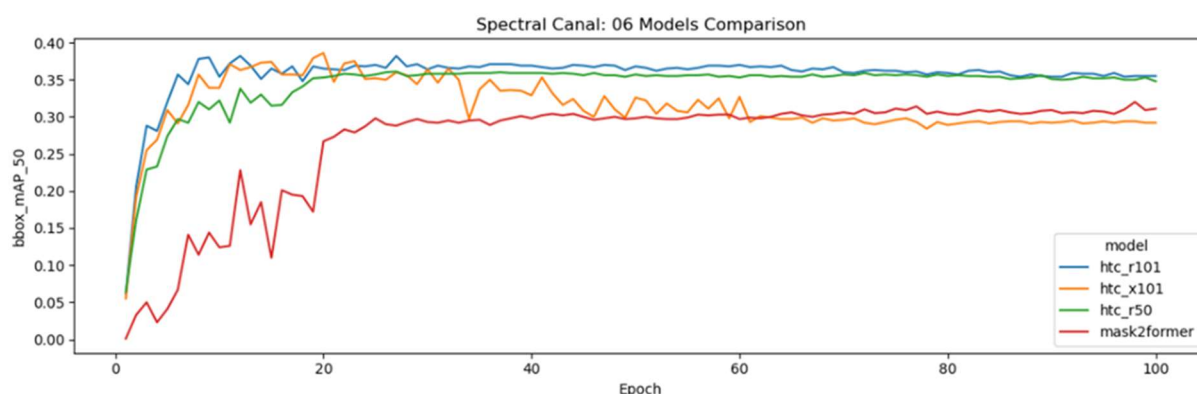
**Rysunek 4.17.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z czwartego kanału spektralnego.

Przechodząc do analizy wyników dla kanału spektralnego nr 04 (Rysunek 4.17) również można zauważyć, że wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox\_mAP\_50 równym 0.4. Wykres dla algorytmu Mask2Former ma szybką i dynamicznie zmieniającą się tendencję wzrostową w pierwszych 20 epokach, po czym stabilizuje się dla wartości bbox\_mAP na poziomie około 0.3. Z kolei wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 10 epok osiągając poziom bbox\_mAP\_50 około 0.4, po czym spada i stabilizuje się na poziomie bbox\_mAP\_50 równym około 0.3 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów osiąga model oparty na HTC\_r101, tuż za nim jest HTC\_r50, a najniższe wyniki można zaobserwować dla modelu opartego o algorytm HTC\_x101.



**Rysunek 4.18.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z piątego kanału spektralnego.

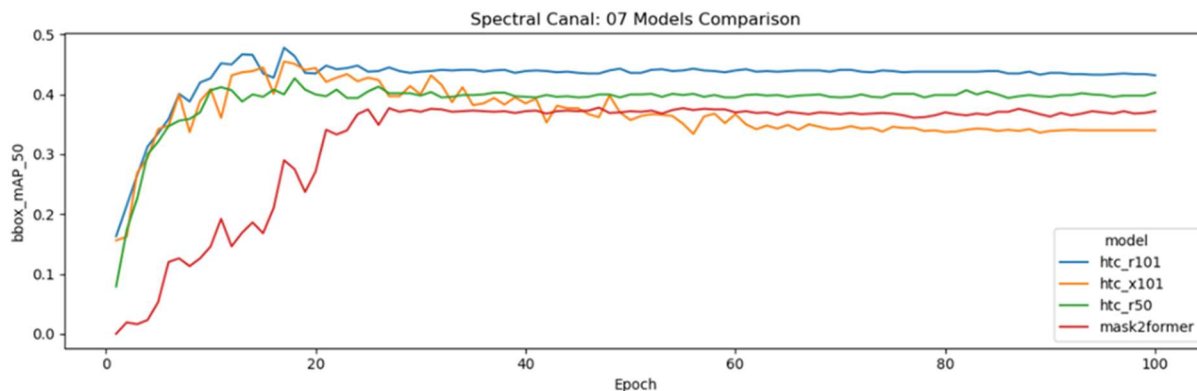
Poddając analizie wyniki dla kanału spektralnego nr 05 (Rysunek 4.18) można zauważyć, że wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox\_mAP\_50 równym 0.4. Wykres dla algorytmu Mask2Former ma szybką tendencję wzrostową w pierwszych 20 epokach, po czym stabilizuje się dla wartości bbox\_mAP na poziomie około 0.4. Z kolei wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 10 epok osiągając poziom bbox\_mAP\_50 około 0.4, po czym spada i stabilizuje się na poziomie bbox\_mAP\_50 równym około 0.3 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów osiąga model oparty na HTC\_r101, tuż za nim jest HTC\_r50, a najniższe wyniki można zaobserwować dla modelu opartego o algorytm HTC\_x101.



**Rysunek 4.19.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z szóstego kanału spektralnego.

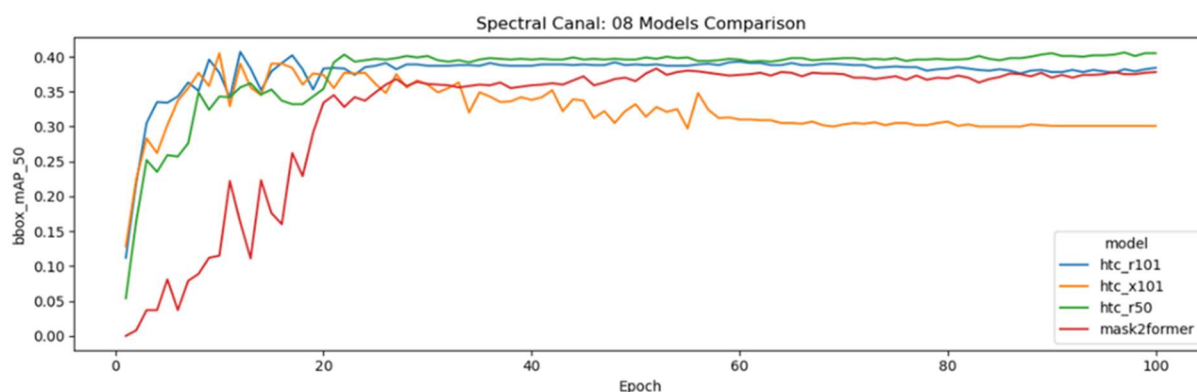
Analizując wyniki dla kanału spektralnego nr 06 (Rysunek 4.19) można zauważyć, że wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox\_mAP\_50 równym 0.4. Wykres dla algorytmu Mask2Former ma szybką tendencję wzrostową w pierwszych 20 epokach, po czym stabilizuje się dla wartości bbox\_mAP na poziomie około 0.3. Wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 10 epok osiągając poziom bbox\_mAP\_50 około 0.38, po czym spada i stabilizuje się na poziomie bbox\_mAP\_50 równym około 0.3 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów osiąga model

oparty na HTC\_r101, tuż za nim jest HTC\_r50, a najniższe wyniki można zaobserwować dla modelu opartego o algorytm HTC\_x101.



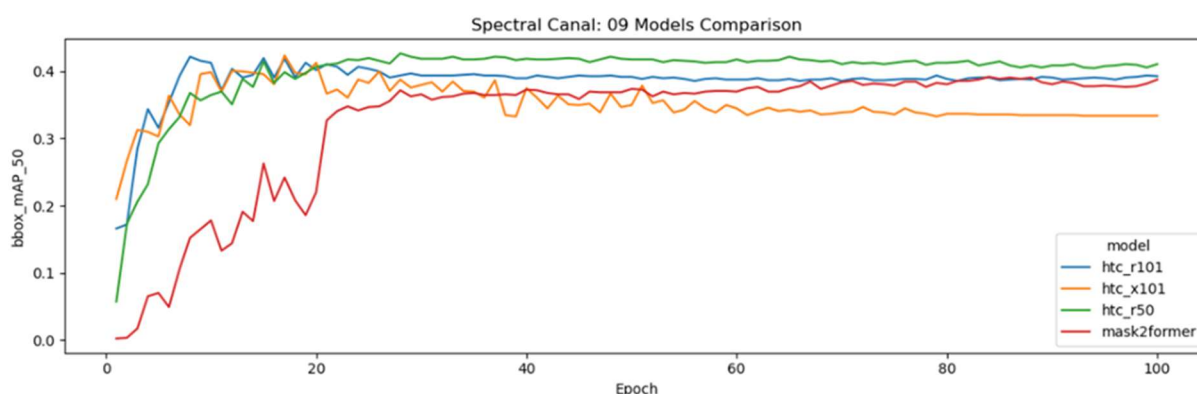
**Rysunek 4.20.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z siódmego kanału spektralnego.

Analizując wyniki dla kanału spektralnego nr 07 (Rysunek 4.20) można zauważyć, że wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox\_mAP\_50 równym 0.4 w przypadku HTC\_r50 oraz 0.45 w przypadku HTC\_r101. Wykres dla algorytmu Mask2Former ma szybką tendencję wzrostową w pierwszych 20 epokach, po czym stabilizuje się dla wartości bbox\_mAP na poziomie około 0.4. Wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 10 epok osiągając poziom bbox\_mAP\_50 około 0.45, po czym spada i stabilizuje się na poziomie bbox\_mAP\_50 równym około 0.3 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów zdecydowanie osiąga model oparty na HTC\_r101, dalej jest algorytm HTC\_r50, a najniższe wyniki można zaobserwować dla modelu opartego o algorytm HTC\_x101.



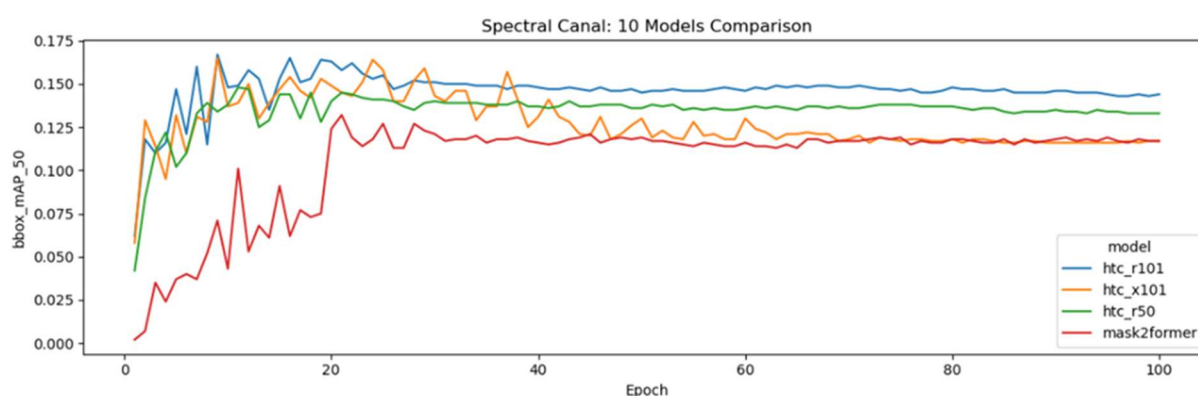
**Rysunek 4.21.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z ósmego kanału spektralnego.

Analizując wyniki dla kanału spektralnego nr 08 (Rysunek 4.21) można zauważyć, że wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox\_mAP\_50 równym około 0.4. Wykres dla algorytmu Mask2Former ma szybką tendencję wzrostową w pierwszych 20 epokach, po czym stabilizuje się również dla wartości bbox\_mAP na poziomie około 0.4. Wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 10 epok osiągając poziom bbox\_mAP\_50 około 0.4, po czym spada i stabilizuje się na poziomie bbox\_mAP\_50 równym około 0.3 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów zdecydowanie osiąga model oparty na HTC\_r50, a najniższe wyniki można zaobserwować dla modelu opartego o algorytm HTC\_x101.



**Rysunek 4.22.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z dziewiątego kanału spektralnego.

Poddając analizie wyniki dla kanału spektralnego nr 09 (Rysunek 4.22) widać, że wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox\_mAP\_50 równym około 0.4. Wykres dla algorytmu Mask2Former ma szybką tendencję wzrostową w pierwszych 20 epokach, po czym stabilizuje się również dla wartości bbox\_mAP na poziomie około 0.4. Wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 10 epok osiągając poziom bbox\_mAP\_50 około 0.4, po czym spada i stabilizuje się na poziomie bbox\_mAP\_50 równym około 0.3 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów osiąga model oparty na HTC\_r50, tuż za nim są algorytmy HTC\_r101 oraz Mask2Former, a najniższe wyniki można zaobserwować dla modelu opartego o algorytm HTC\_x101.



**Rysunek 4.23.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z dziesiątego kanału spektralnego.

Analizując zaś wyniki dla kanału spektralnego nr 10 (Rysunek 4.23) można zaobserwować, iż wykresy dla algorytmów HTC\_r101 i HTC\_r50 mają szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox\_mAP\_50 równym około 0.15. Wykres dla algorytmu Mask2Former ma szybką tendencję wzrostową w pierwszych 20 epokach, po czym stabilizuje się również dla wartości bbox\_mAP na poziomie około 0.125. Wykres dla algorytmu HTC\_x101 dynamicznie rośnie w przedziale pierwszych 10 epok osiągając poziom bbox\_mAP\_50 około 0.4, po czym spada i stabilizuje się na poziomie bbox\_mAP\_50 równym około 0.125 do końca treningu. Najwyższe wyniki spośród wszystkich algorytmów osiąga model oparty na HTC\_r101, tuż za nim są algorytmy HTC\_r101 oraz



Mask2Former, a najniższe wyniki można zaobserwować dla modeli opartych o algorytmy HTC\_x101 oraz Mask2Former.

Jak widać na podstawie krzywych uczenia, w przeważającej liczbie kanałów spektralnych najslabsze wyniki można zaobserwować w przypadku modelu wytrenowanego z użyciem algorytmu HTC\_x101, czyli tego, który osiągnął najlepsze wyniki w przypadku trenowania na zbiorze obrazów RGB. Co więcej model, który miał najniższe wyniki w przypadku trenowania na zbiorze obrazów RGB, czyli Mask2Former, w przypadku poszczególnych kanałów spektralnych osiąga wyniki porównywalne do wyników otrzymanych z użyciem innych analizowanych algorytmów, a w przypadku kanału spektralnego nr 01 osiągnął najwyższe wyniki spośród analizowanych algorytmów. Warto również podkreślić, że dla kanału spektralnego nr 10 każdy z analizowanych algorytmów osiągnął odpowiednio niższe wyniki, niż dla pozostałych kanałów spektralnych.

#### 4.6. Porównanie wyników i wybór najlepszych rozwiązań

Wynikiem przeprowadzonych prac jest zaproponowane nowatorskie podejście do automatyzacji precyzyjnego monitorowania wzrostu kukurydzy, które wykorzystuje zaawansowane technik analizy obrazów oraz sztucznej inteligencji, polegające na zastosowaniu głębokich sieci neuronowych do analizy zarówno obrazów RGB, jak i wielospektralnych, z dodatkowym wykorzystaniem klasyfikatora głosującego. Przeprowadzone w tym kierunku zostały badania mające na celu detekcję i klasyfikację faz rozwoju kukurydzy zgodnie z międzynarodową skalą rozwoju roślin BBCH, przy użyciu obrazowania wielospektralnego i RGB. Wiodącym celem było opracowanie efektywnej metody, która umożliwi automatyczne monitorowanie faz rozwoju roślin i generowanie wyników, aby możliwa była ich późniejsza prezentacja dla większych obszarów, np. w postaci wizualizacji mapowej.

Uzyskane wyniki pokazują innowacyjne podejście do tematu automatycznej detekcji i klasyfikacji faz rozwojowych roślin, w tym przypadku kukurydzy, i osiągnięta została wysoka precyzja określania aktualnego poziomu rozwoju roślin z dokładnością do pojedynczych obiektów. Opracowane podejście umożliwiło zwiększenie szybkości klasyfikacji faz rozwoju kukurydzy w porównaniu do dotychczasowych ręcznych metod. Dzięki zastosowaniu zaawansowanych technik analizy obrazów i algorytmów głębokiego uczenia, udało się osiągnąć oczekiwany poziom automatyzacji procesu określania faz rozwojowych kukurydzy, a w oparciu o analizę literaturową można stwierdzić, iż opracowane podejście jest innowacyjne.

Modele wykorzystywane w opracowanym rozwiązaniu zostały wytrenowane na stworzonych w tym celu autorskich zbiorach danych składających się z oznaczonych obrazów kukurydzy znajdującej się w różnych etapach rozwoju odpowiadających poszczególnym fazom rozwoju kukurydzy w skali BBCH (fazy 10-19). Obrazy te były rejestrowane zarówno w spektrum widzialnym (RGB), jak i w spektrum wielospektralnym. To umożliwiło przeprowadzenie kompleksowych badań i trenowanie modeli na danych RGB oraz na danych z poszczególnych kanałów spektralnych. Co więcej, do procesu trenowania sztucznych sieci neuronowych wykorzystane zostały różne algorytmy uczenia głębokiego, co pozwoliło na obserwacje wyników zarówno pod kątem skuteczności detekcji i klasyfikacji w zależności od użytego zbioru obrazów treningowych, a także w zależności od zastosowanego algorytmu.

Na bazie obserwacji można zauważyć, że lepsze wyniki niż model wytrenowany na zbiorze obrazów RGB, osiągnął klasyfikator głosujący, w którym modele wytrenowane na pojedynczych kanałach spektralnych głosowały nad wynikiem dotyczącym wyboru konkretnej fazy rozwojowej wykrytych roślin kukurydzy.

Pojedyncze modele jednokanałowych danych mają w sumie gorszą jakość, ponieważ zostały wytrenowane na jednokanałowych obrazach, podczas gdy modele RGB zostały wytrenowane na trójkanałowych danych. Zastosowanie techniki głosowania poprawiło jakość sumarycznej predykcji. Problem jaki występuje w danych jest taki, że obrazy dla różnych kanałów były robione z różnej perspektywy i z innych kątów patrzenia kamery. Dla pojedynczych modeli mogło to spowodować znaczne różnice w klasyfikacji, które zostały poprawione przez zastosowanie techniki głosowania.

Aby zaadresować problem który leży u podłoża niskich wyników na pojedynczych kanałach spektralnych, należy zasugerować wykonanie poprawionego datasetu i można to osiągnąć na dwa sposoby: po pierwsze - pozyskiwanie obrazów z wykorzystaniem jednoobiektywowej kamery, aby zobrazowanie z każdego kanału spektralnego było wykonane z takiej samej perspektywy i z tego samego położenia obiektywu kamery i po drugie - pozyskiwanie obrazów z daleka oraz zoomowanie obrazy w taki sposób, aby różnice w różnym kącie patrzenia były jak najmniejsze. Tak wykonane obrazy można następnie złączyć do jednego wielokanałowego obrazu, który zostanie wykorzystany do wytrenowania uaktualnionego modelu. PACE te zostały zrealizowane również w ramach niniejszej rozprawy, a ich wyniki opisane zostały w kolejnym rozdziale.

Dodatkowymi cennymi wnioskami z przeprowadzonych badań jest obserwacja zachowań różnych algorytmów i uzyskiwanych przez nie wyników w zależności od zastosowanego zbioru treningowego. Spośród wszystkich badanych algorytmów trenowanych na zbiorze danych RGB najlepsze wyniki uzyskał algorytm HTC\_x101. Z kolei spośród wszystkich badanych algorytmów trenowanych na zbiorze danych wielospektralnych i przy zastosowaniu klasyfikatora głosującego najlepsze wyniki uzyskał algorytm HTC\_r50. Porównując zaś wyniki poszczególnych algorytmów w zależności od zastosowanego zbioru treningowego można zauważyć, że algorytmy HTC\_r101 oraz HTC\_x101 uzyskują lepsze wyniki dla zbioru treningowego składającego się z obrazów RGB. Z kolei algorytmy HTC\_r50 oraz Mask2Former uzyskują wyższe wyniki przy trenowaniu na zbiorach składających się z pojedynczych kanałów spektralnych i zastosowaniu następnie klasyfikatora głosującego (przy czym w przypadku algorytmu Mask2Former różnica ta jest bardzo wyraźna). Są to istotne informacje z punktu widzenia przyszłych badań oraz wkładu do obecnego stanu nauki w tej kwestii.

Analizując poszczególne pojedyncze kanały spektralne można zauważyć, że:

- Algorytm HTC\_r101 osiągnął najlepsze wyniki spośród badanych algorytmów dla zbiorów treningowych odpowiadających następującym kanałom spektralnym: canal 02, canal 04, canal 05, canal 06, canal 07, canal 10 i w żadnym z analizowanych kanałów nie osiągnął najgorszych wyników spośród analizowanych algorytmów.
- Algorytm HTC\_x101 osiągnął najlepsze wyniki spośród badanych algorytmów dla zbioru obrazów RGB oraz najgorsze wyniki spośród analizowanych algorytmów dla następujących zbiorów treningowych odpowiadających poszczególnym kanałom spektralnym: canal 01, canal 02, canal 04, canal 05, canal 06, canal 07, canal 08, canal 09, canal 10.
- Algorytm HTC\_r50 osiągnął najlepsze wyniki spośród badanych algorytmów dla następujących zbiorów treningowych odpowiadających poszczególnym kanałom spektralnym: canal 03, canal 08, canal 09 i w żadnym z analizowanych kanałów nie osiągnął najgorszych wyników spośród analizowanych algorytmów.
- Algorytm Mask2Former osiągnął najlepsze wyniki spośród badanych algorytmów dla zbioru treningowego odpowiadającego kanałowi spektralnemu canal 01. Algorytm ten osiągnął zaś najgorsze wyniki spośród analizowanych algorytmów dla zbioru

treningowego obrazów RGB oraz dla zbiorów treningowych odpowiadających następującym kanałom spektralnym: canal 03, canal 10.

Dzięki opracowanemu podejściu osiągnięta została wysoka dokładność klasyfikacji dzięki zastosowaniu głębokich sztucznych sieci neuronowych, które są w stanie efektywnie uczyć się istotnych cech potrzebnych do klasyfikacji. Dodatkowo, użycie klasyfikatora głosującego pozwoliło na poprawę wyników dla obrazów wielospektralnych poprzez łączenie wyników z kilku różnych modeli odpowiadających pojedynczym kanałom spektralnym.

Kluczowym wnioskiem jest także sprawdzenie, który z pojedynczych kanałów spektralnych najlepiej nadaje się do detekcji faz rozwojowych kukurydzy w skali BBCH z wykorzystaniem badanych algorytmów. Dla algorytmu HTC\_r101 najlepszym kanałem okazał się canal 07, dla algorytmu HTC\_x101 najlepszymi kanałami okazały się canal 05 i canal 07, dla algorytmu HTC\_r50 najlepszym kanałem okazał się canal 05, zaś dla algorytmu Mask2Former najlepszym kanałem okazał się canal 02.

Zaprezentowane rozwiązanie jest unikatowe w kontekście automatycznego i precyzyjnego monitorowania faz rozwoju kukurydzy przy użyciu obrazów wielospektralnych i obrazów RGB. Uzyskane wyniki zostały szczegółowo przeanalizowane, uwzględniając szerokie spektrum widzenia kamery wielospektralnej. To pozwoliło na uchwycenie więcej informacji o stanie roślin, co czyni opracowany model bardziej kompleksowym i dokładnym w porównaniu do modeli wykorzystujących jedynie obrazowanie RGB.

Dzięki prezentowanej metodzie, możliwe jest dokładne śledzenie stanu roślin i podejmowanie trafnych decyzji w rolnictwie precyzyjnym, gdzie dokładna identyfikacja etapów wzrostu roślin może pomóc w optymalizacji nawadniania, nawożenia oraz ochrony roślin. Możliwe jest również potencjalne rozszerzenie opracowanych metod na inne typy upraw i wiąże się to ze stworzeniem odpowiednich zbiorów uczących oraz trenowaniem na nich wskazanych algorytmów uczenia głębokiego.

W rzeczywistych warunkach, model wytrenowany na obrazach RGB może być łatwo wdrożony przy użyciu standardowych kamer zamontowanych np. na robotach polowych. Dzięki temu rolnicy mogą na bieżąco monitorować etapy rozwoju kukurydzy i podejmować odpowiednie działania w czasie rzeczywistym, co przyczyni się do zwiększenia efektywności i wydajności upraw.

## **5. Detekcja i klasyfikacja faz rozwoju kukurydzy w skali BBCH przy wykorzystaniu obrazowania RGB i kamery wielospektralnej MapIR**

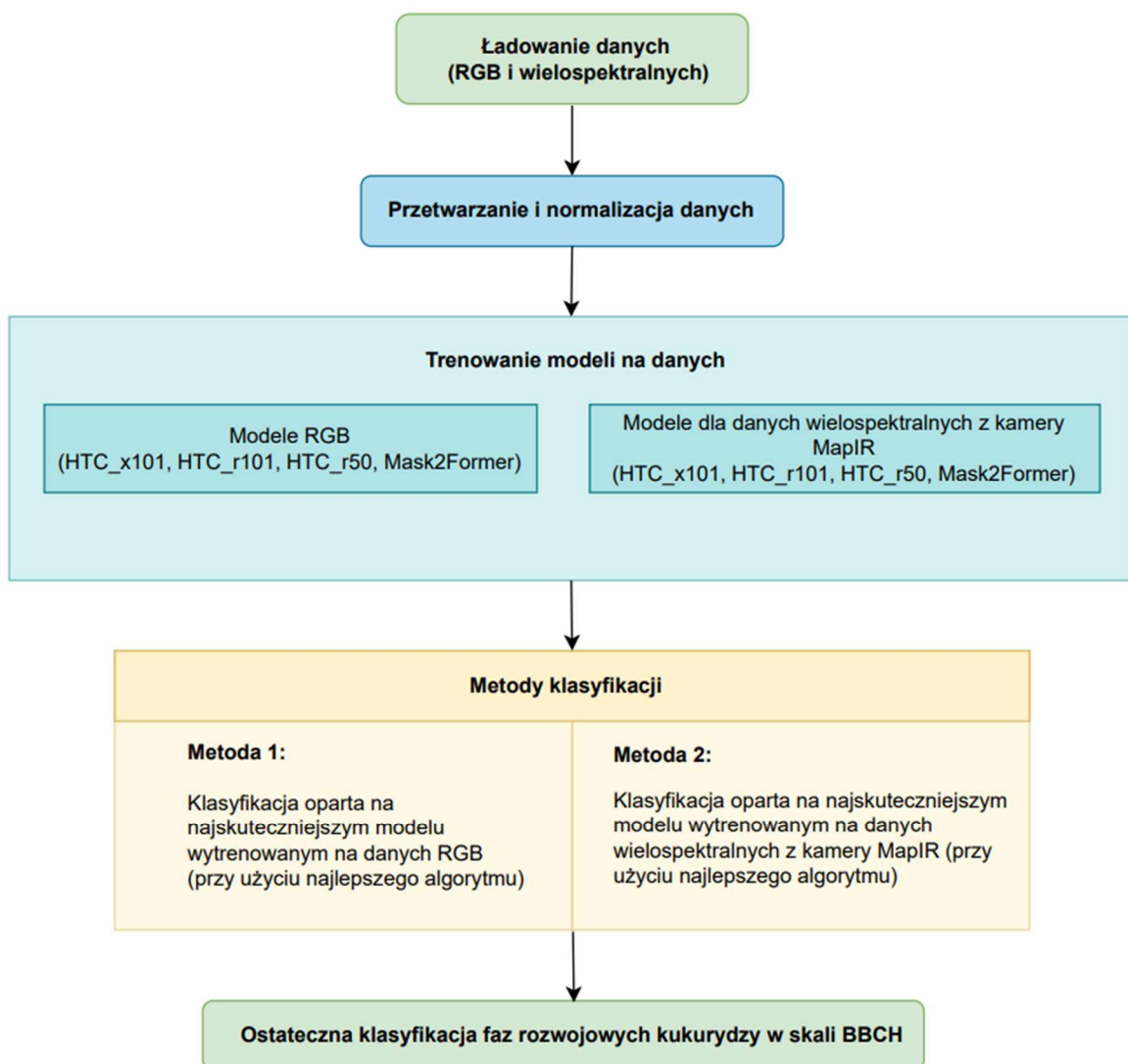
Niniejszy rozdział skupia się na replikacji wyników uzyskanych w ramach zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH z zastosowaniem następujących architektur uczenia głębokiego: HTC\_r101, HTC\_r50, HTC\_x101 oraz Mask2Former.

Do realizacji pracach została wykorzystana kamera wielospektralna MapIR Survey3 Red+Green+NIR (RGN), która umożliwiła jednoczesne pozyskiwanie obrazów z poszczególnych kanałów spektralnych za pośrednictwem jednego obiektywu. To pozwoliło na uniknięcie problemu zauważonego podczas badań z wykorzystaniem kamery wielospektralnej MicaSense RedEdge, gdzie ta sama roślina rejestrowana przez różne obiektywy kamery wielospektralnej miała inny kształt w każdym z rejestrowanych kanałów spektralnych. Pozyskiwanie obrazów z kamery MapIR pozwoliło na utworzenie na ich bazie zbioru treningowego, który zawiera zestaw informacji z innych pasm spektrum światła, niż standardowe zobrazowanie RGB. Cały eksperyment został przeprowadzony również z wykorzystaniem kamery RGB. Obrazy były pozyskiwane z tej samej pozycji i w tym samym czasie, co zobrazowania z kamery MapIR, aby zachować identyczne warunki badawcze oraz aby jedyną zmienną podczas eksperymentu był rodzaj kamery oraz rodzaj pozyskiwanych zobrazowań.

W kolejnych podrozdziałach zaprezentowano schemat opracowanego rozwiązania, wyniki uzyskane dla danych wielospektralnych, wyniki uzyskane dla danych RGB, krzywe uczenia dla danych wielospektralnych i RGB oraz porównanie wyników i wybór najlepszych rozwiązań.

### **5.1. Schemat opracowanego rozwiązania**

W tej sekcji przedstawiony zostanie szczegółowy opis rozwiązań użytych w badaniu dotyczącym detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH. Rysunek 4.1. pokazuje ogólny schemat zaproponowanego systemu.



**Rysunek 5.1.** Schemat zaproponowanego systemu klasyfikacji faz rozwojowych kukurydzy w skali BBCH z użyciem zobrazowania RGB oraz zobrazowania z kamery MapIR.

Schemat przedstawia dwa główne podejścia zastosowane w opracowanym systemie: klasyfikacja na obrazach RGB oraz klasyfikacja na obrazach pochodzących z kamery wielospektralnej MapIR. Rozwiązania te zaproponowano, aby sprawdzić i porównać wyniki otrzymane przy zastosowaniu standardowego obrazowania RGB oraz zobrazowania pozyskiwanego za pomocą kamery wielospektralnej MapIR, która należy do wyposażenia Działu Teledetekcji – Sieci Badawczej Łukasiewicz. Konfrontacja obu podejść mogła przynieść odpowiedź na pytanie o skuteczność procesu detekcji i klasyfikacji faz rozwojowych kukurydzy z wykorzystaniem tych samych analogicznych modeli uczenia głębokiego, lecz różnych typów zobrazowań. Prace miały więc charakter badawczo-wdrożeniowy z potencjałem



na wykorzystanie w innych zadaniach, gdzie ważna jest skuteczność procesu detekcji i klasyfikacji. W ten oto sposób opracowano dwa główne podejścia.

W pierwszym podejściu (Metoda 1) zastosowana została klasyfikacja obrazów RGB przy użyciu wybranego algorytmu, spośród puli algorytmów analizowanych podczas badań. Drugie podejście (Metoda 2) polega na analogicznym wykorzystaniu tych samych algorytmów, z tym, że z wykorzystaniem obrazów z kamery MapIR. Finalna klasyfikacja jest wynikiem wyboru najskuteczniejszej metody opartej o jedno ze zobrazowań: RGB lub z kamery MapIR oraz o algorytm, który osiągnął najlepsze wyniki spośród analizowanych algorytmów.

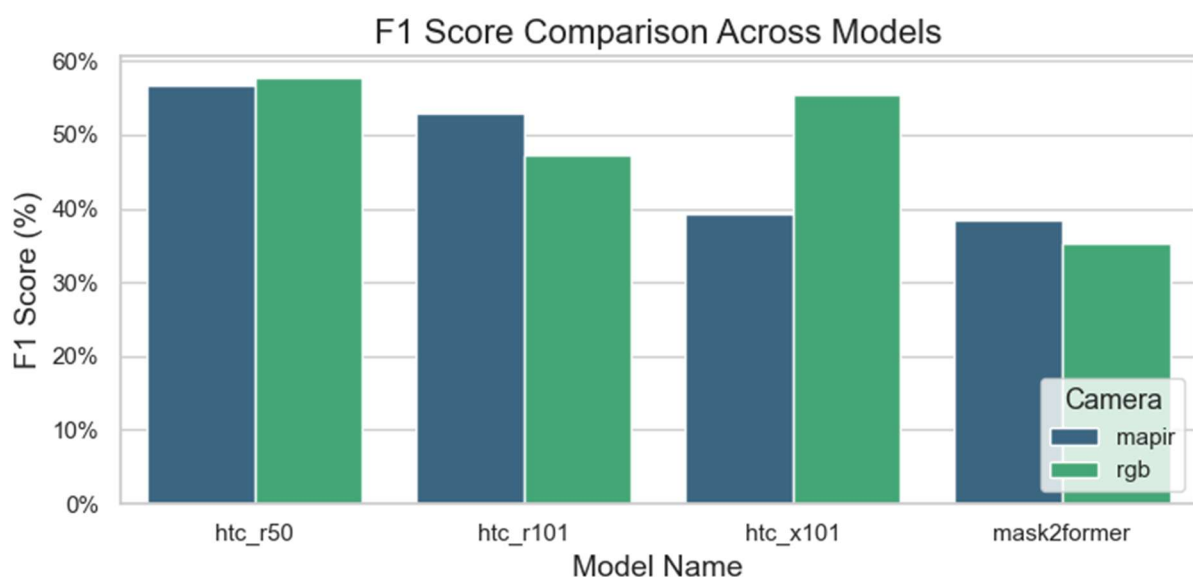
## 5.2. Wyniki uzyskane dla danych wielospektralnych i RGB

Podczas badań zaobserwowano różnice w wynikach klasyfikacji pomiędzy poszczególnymi algorytmami wykorzystywanymi do trenowania głębokich sieci neuronowych oraz pomiędzy tymi samymi algorytmami, ale różnymi typami zobrazowania. W tabeli 5.1. przedstawiono wyniki klasyfikacji dla każdego z algorytmów oraz dla obu typów zobrazowania. Najlepsze wyniki klasyfikacji obrazów RGB uzyskał algorytm HTC\_r50, zaraz za nim był algorytm HTC\_x101, następnie HTC\_r101, a najgorsze wyniki uzyskano przy zastosowaniu algorytmu Mask2Former. Najlepsze wyniki klasyfikacji obrazów wielospektralnych z kamery MapIR uzyskał algorytm HTC\_r50, zaraz za nim był algorytm HTC\_r101, następnie HTC\_x101, a najgorsze wyniki uzyskano przy zastosowaniu algorytmu Mask2Former. Analizując wyniki pod kątem najskuteczniejszej kombinacji użytego algorytmu oraz typu zobrazowania można wywnioskować, iż najlepsze wyniki osiągnięto dzięki zastosowaniu zobrazowania RGB oraz algorytmu HTC\_r50. W Tabeli 5.1. przedstawiono wyniki poszczególnych algorytmów z uwzględnieniem analizowanego rodzaju zobrazowania, wyrażone za pomocą metryk oceny algorytmów Accuracy, Precision, Recall oraz F1-score.

|             |                 | accuracy | precision | recall   | f1       |
|-------------|-----------------|----------|-----------|----------|----------|
| model_name  | experiment_name |          |           |          |          |
| htc_r101    | mapir           | 0.547445 | 0.555562  | 0.545144 | 0.529451 |
|             | rgb             | 0.562044 | 0.520592  | 0.478862 | 0.472016 |
| htc_r50     | mapir           | 0.587591 | 0.586635  | 0.577213 | 0.568128 |
|             | rgb             | 0.682482 | 0.574147  | 0.592729 | 0.577468 |
| htc_x101    | mapir           | 0.485401 | 0.414784  | 0.409376 | 0.391897 |
|             | rgb             | 0.562044 | 0.543993  | 0.619409 | 0.554181 |
| mask2former | mapir           | 0.463504 | 0.413548  | 0.390541 | 0.384978 |
|             | rgb             | 0.470803 | 0.399994  | 0.380418 | 0.353583 |

**Tabela 5.1.** Wyniki klasyfikacji dla poszczególnych algorytmów oraz analizowanych typów zobrazowania RGB oraz wielospektralnego z kamery MapIR.

Rysunek 5.2 przedstawia z kolei wykres wartości parametru F1-score dla poszczególnych zobrazowań oraz analizowanych typów zobrazowania: RGB oraz zobrazowania wielospektralnego z kamery MapIR.



**Rysunek 5.2.** Wykres przedstawiający wartości parametru F1-score dla poszczególnych zobrazowań oraz analizowanych typów zobrazowania: RGB oraz zobrazowania wielospektralnego z kamery MapIR.

Na Rysunkach 5.3. - 5.5. zaprezentowano wyniki działania modelu HTC\_r50 wytrenowanego na obrazach RGB. Z kolei na Rysunkach 5.6. – 5.8. zaprezentowano wyniki działania modelu HTC\_r50 wytrenowanego na danych wielospektralnych z kamery MapIR.

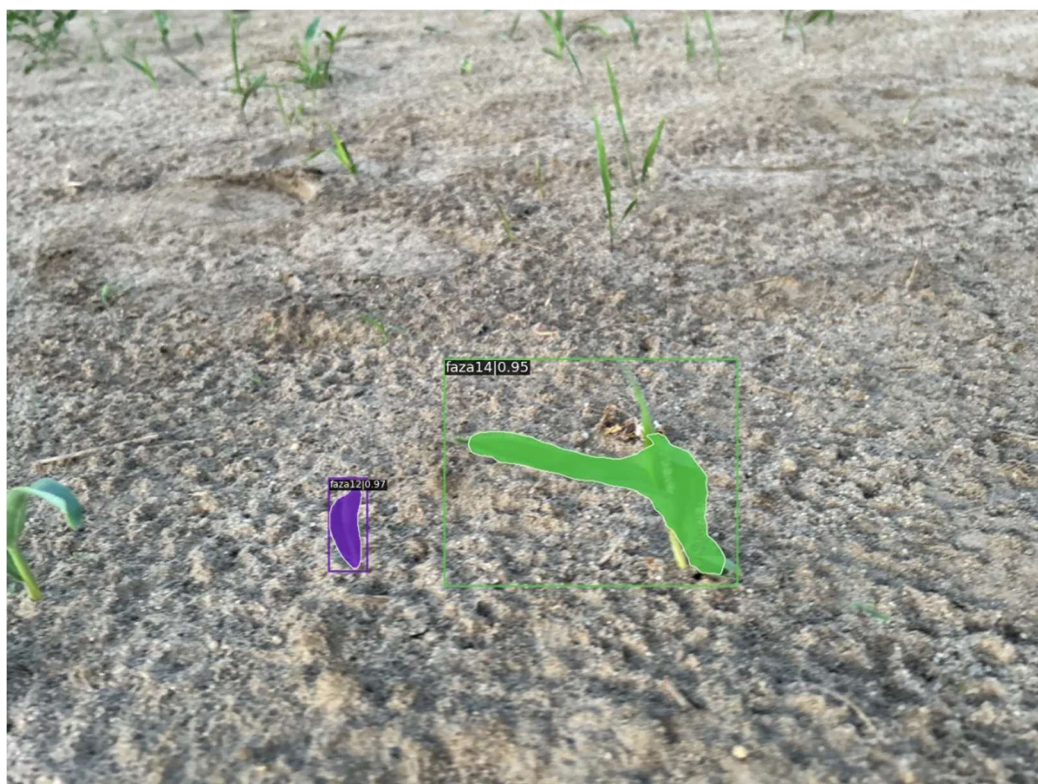


**Rysunek 5.3.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach RGB.

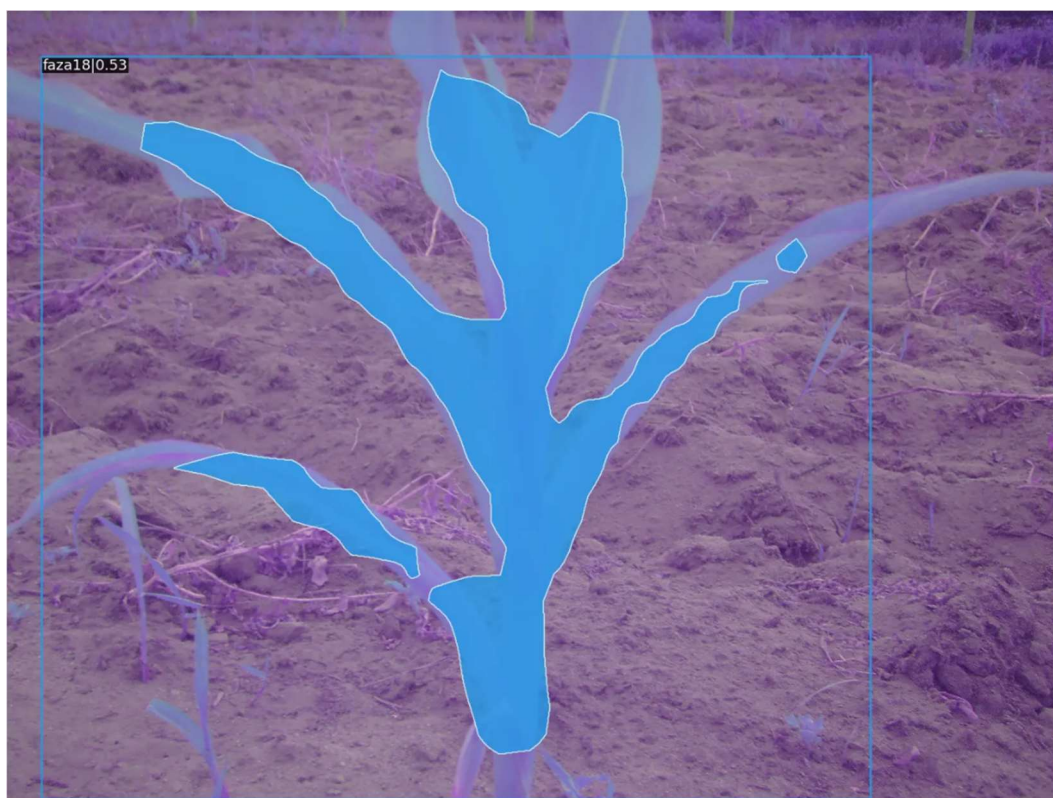


**Rysunek 5.4.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach RGB.





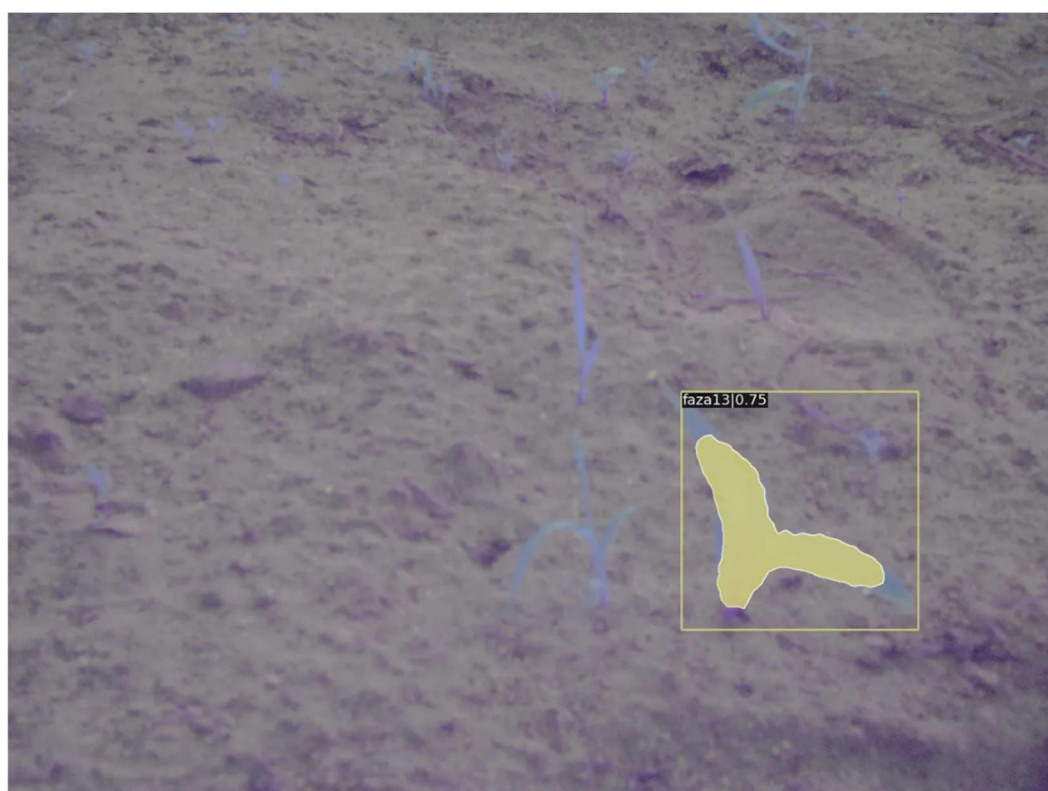
**Rysunek 5.5.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach RGB.



**Rysunek 5.6.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR.



**Rysunek 5.7.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR.

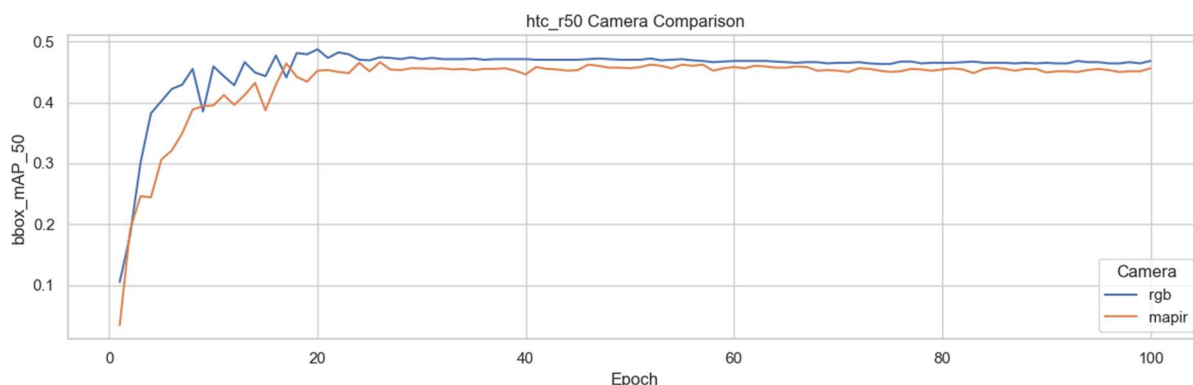


**Rysunek 5.8.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR.



### 5.3. Krzywe uczenia dla danych wielospektralnych i RGB

W niniejszym podrozdziale zaprezentowane zostały krzywe uczenia zarejestrowane podczas trenowania poszczególnych modeli. Poniższe wykresy przedstawiają przebieg uczenia dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.



**Rysunek 5.9.** Przebieg uczenia modelu opartego o algorytm HTC\_r50 dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.

Rysunek 5.9. przedstawia porównanie wyników mAP (w postaci bbox\_mAP\_50) dla modelu HTC\_r50 trenowanego na dwóch rodzajach danych wejściowych, pochodzących z kamer RGB oraz MapIR. Na obu krzywych można zaobserwować szybki wzrost wartości mAP w początkowych etapach treningu, co wskazuje, że model w pierwszych kilkunastu epokach przyswaja kluczowe informacje z danych. W obu przypadkach, w ciągu pierwszych dziesięciu epok, następuje gwałtowny wzrost mAP, gdzie wartości początkowe na poziomie 0.1-0.2 rosną odpowiednio do 0.45 dla danych RGB i około 0.42 dla danych z kamery MapIR. Szybka konwergencja w początkowych fazach nauki sugeruje, że model skutecznie uczy się z danych, co może być wynikiem dobrze zbalansowanych danych wejściowych oraz odpowiednio dobranych hiperparametrów.

Po osiągnięciu tych maksymalnych wartości mAP, krzywe wykazują wyraźną stabilizację. Od około 30. epoki widać, że model nie wykazuje już istotnych wahań wydajności, co sugeruje, że proces uczenia się osiągnął punkt zbieżności. Wartość mAP utrzymuje się na stabilnym poziomie zarówno dla RGB, jak i MapIR, co oznacza, że model nie poprawia swojej wydajności w kolejnych epokach, ale także nie ulega spadkowi, co jest pozytywnym sygnałem świadczącym o braku nadmiernego przetrenowania.

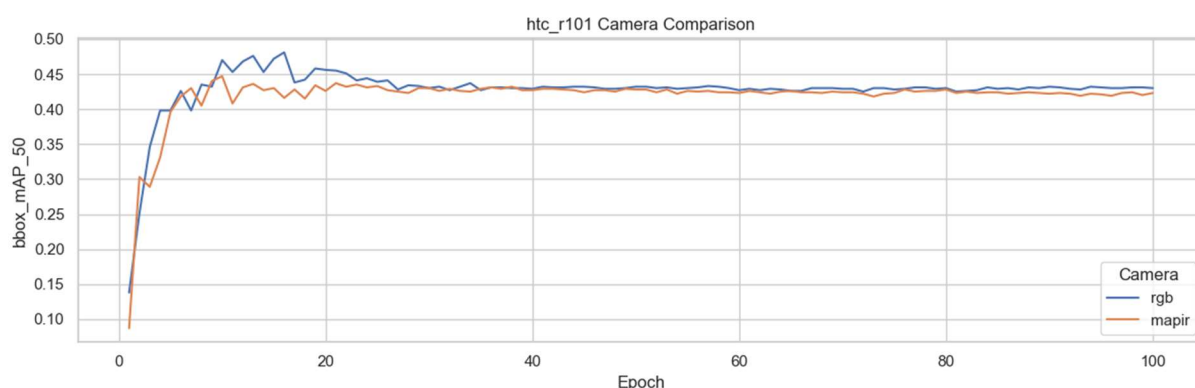
Porównując oba rodzaje danych, zauważalna jest pewna przewaga danych z kamery RGB nad danymi z kamery MapIR, choć różnica ta nie jest duża. Model lepiej radzi sobie z



obrazami RGB, osiągając nieco wyższą wartość mAP, co może sugerować, że obrazy te dostarczają więcej informacji istotnych dla zadania detekcji obiektów. Obrazy z kamery MapIR, choć stabilne, wykazują mniejszą wydajność, co może wynikać z ograniczeń spektralnych tych danych lub większej specyficzności informacji, jakie one zawierają.

Pomimo tych różnic, obie krzywe pozostają stosunkowo stabilne po osiągnięciu punktu zbieżności, co wskazuje na skuteczne uogólnienie przez model, bez oznak overfittingu. Brak znaczących fluktuacji po osiągnięciu optymalnej wartości mAP sugeruje, że model generalizuje dobrze, a stabilność krzywych świadczy o odpowiednio przeprowadzonym procesie treningowym.

Podsumowując, model HTC\_r50 wykazał się wysoką stabilnością i skutecznością w klasyfikacji obiektów zarówno na podstawie obrazów RGB, jak i MapIR, przy czym dane RGB zapewniły nieco lepsze wyniki. Wyniki te sugerują, że model osiągnął optymalny poziom wydajności na tym zestawie danych, nie wykazując oznak przetrenowania ani spadku wydajności w miarę trwania treningu.



**Rysunek 5.10.** Przebieg uczenia modelu opartego o algorytm HTC\_r101 dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.

Rysunek 5.10. przedstawia porównanie wyników mAP (bbox\_mAP\_50) dla modelu HTC\_r101 trenowanego na obrazach pochodzących z dwóch rodzajów kamer: RGB oraz MapIR. Oba zestawy danych prowadzą do podobnych wyników, jednak istnieją subtelne różnice w dynamice uczenia się modelu w zależności od typu danych wejściowych.

Na początku treningu (pierwsze 10-15 epok) widać gwałtowny wzrost wartości mAP dla obu typów kamer. Krzywa dla danych RGB w pierwszych epokach wzrasta szybciej i osiąga nieco wyższą wartość maksymalną (~0.47) w porównaniu do danych z kamery MapIR (~0.43).

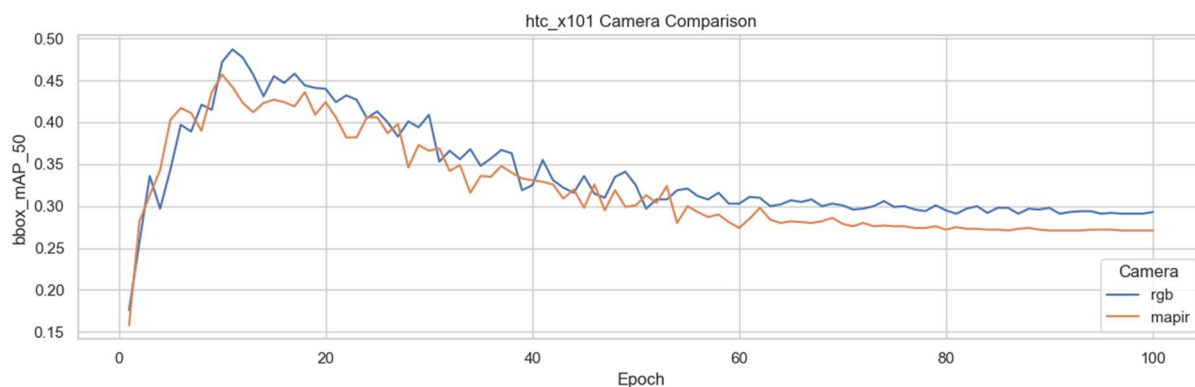
Szybka konwergencja na obu krzywych sugeruje, że model szybko uczy się kluczowych cech z danych, co wskazuje na efektywność trenowania.

Po początkowej fazie wzrostu następuje stabilizacja wartości mAP w obu przypadkach. Od około 20. epoki krzywe te wykazują jedynie drobne fluktuacje i stabilizują się na poziomach zbliżonych do 0.45 dla RGB i 0.42-0.43 dla MapIR. Oznacza to, że model osiągnął punkt zbieżności i nie poprawia dalej swojej wydajności, co jest typowe dla dobrze trenowanego modelu, który nie cierpi na nadmierne dopasowanie (overfitting).

Niewielkie różnice między danymi RGB a MapIR wskazują, że choć model jest w stanie skutecznie uczyć się z obu typów obrazów, to jednak dane RGB zapewniają nieco bogatszy zestaw cech, co pozwala na osiągnięcie wyższych wyników. Może to wynikać z pełniejszego spektrum informacji dostępnego w obrazach RGB, które lepiej reprezentują charakterystyki obiektów, podczas gdy dane MapIR mogą być bardziej specyficzne, a przez to mniej różnorodne.

Stabilność obu krzywych w późniejszych epokach sugeruje, że model generalizuje dobrze na danych walidacyjnych i nie ma problemów z niestabilnością procesu uczenia. Brak wyraźnych oznak overfittingu, w postaci spadków wydajności, świadczy o skutecznie dobranych hiperparametrach, takich jak odpowiednia wartość learning rate czy liczba epok.

Podsumowując, model HTC\_r101 pokazuje stabilne i wysokie wyniki w detekcji obiektów na podstawie danych RGB i MapIR, przy czym obrazy RGB dostarczają nieco wyższej jakości informacji dla tego modelu. Pomimo tych różnic, obie krzywe sugerują, że model dobrze uogólnia informacje, co świadczy o skuteczności zastosowanego podejścia.



**Rysunek 5.11.** Przebieg uczenia modelu opartego o algorytm HTC\_x101 dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.

Rysunek 5.11. przedstawia porównanie wyników mAP (bbox\_mAP\_50) dla modelu HTC\_x101 trenowanego na dwóch typach danych: obrazach z kamer RGB oraz MapIR. Na podstawie krzywych można zaobserwować wyraźnie różną dynamikę uczenia się modelu oraz pewne problemy z utrzymaniem wysokiej wydajności w późniejszych fazach treningu.

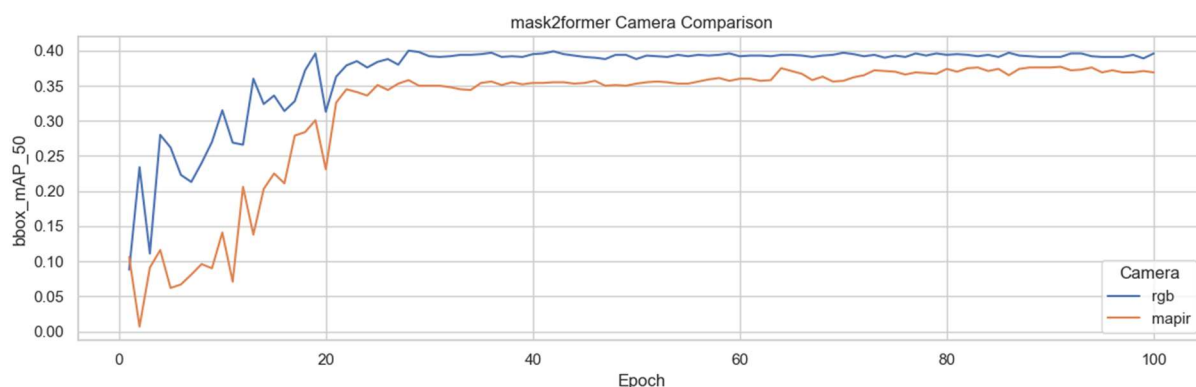
Początkowo, zarówno dla danych RGB, jak i MapIR, obserwujemy szybki wzrost mAP w ciągu pierwszych kilkunastu epok. Model osiąga maksymalną wartość mAP około 0.45 dla RGB i około 0.40 dla MapIR po około 20 epokach. Ta początkowa faza gwałtownego wzrostu sugeruje, że model szybko uczy się kluczowych cech z danych, jednak już na tym etapie można zauważyć, że dane RGB prowadzą do lepszych wyników niż dane MapIR.

Po osiągnięciu wartości szczytowych, zarówno dla danych RGB, jak i MapIR, następuje powolny spadek mAP. W przypadku danych RGB krzywa opada wolniej, utrzymując się na poziomie około 0.35 pod koniec treningu. Natomiast dla danych MapIR spadek jest bardziej wyraźny, a końcowa wartość mAP wynosi około 0.30. Taki spadek wydajności może sugerować, że model zaczyna tracić zdolność do generalizowania lub występują problemy z przetrenowaniem (overfitting).

Fluktuacje widoczne na krzywych mAP dla obu typów danych w środkowej fazie treningu mogą świadczyć o niestabilności modelu w zakresie rozpoznawania cech, co również może być związane z nadmiernym dopasowaniem modelu do danych treningowych. Stabilizacja pod koniec treningu, choć na niższym poziomie niż w początkowej fazie, sugeruje, że model zdołał częściowo uogólnić nauczane cechy, jednak nie osiągnął pełnej stabilności.

Porównując oba zestawy danych, wyniki RGB są wyraźnie lepsze niż dla MapIR, co wskazuje, że model lepiej radzi sobie z obrazami RGB. Różnica ta może wynikać z faktu, że obrazy RGB dostarczają więcej różnorodnych informacji, które są łatwiej przetwarzane przez model, w przeciwieństwie do bardziej specyficznych danych MapIR.

Podsumowując, model HTC\_x101 początkowo osiąga dobre wyniki zarówno na danych RGB, jak i MapIR, jednak w miarę postępu treningu wydajność modelu spada, co może sugerować konieczność dostrojenia hiperparametrów lub zastosowania technik zapobiegających przetrenowaniu. Warto zwrócić uwagę na lepszą stabilność i wyższą skuteczność modelu przy pracy z danymi RGB, co sugeruje, że obrazy RGB dostarczają bogatszych informacji dla zadania detekcji.



**Rysunek 5.12.** Przebieg uczenia modelu opartego o algorytm Mask2Former dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.

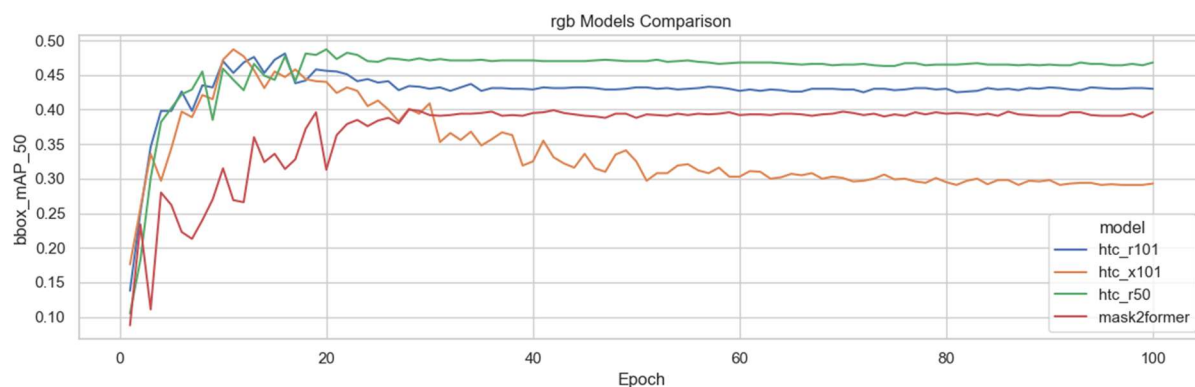
Rysunek 5.12 przedstawia porównanie wyników mAP (bbox\_mAP\_50) dla modelu Mask2Former trenowanego na obrazach pochodzących z kamer RGB oraz MapIR. Analizując krzywe dla obu typów kamer, można zauważyć wyraźne różnice w wydajności modelu, a także stabilizację wyników po początkowych fazach treningu.

Dla obrazów RGB model bardzo szybko zwiększa swoją skuteczność, osiągając wartość mAP na poziomie 0.35 już po około 20 epokach. Po tym szybkim wzroście wydajność modelu ulega stabilizacji, a krzywa oscyluje wokół wartości 0.38 przez resztę treningu. Ta stabilność świadczy o dobrym uogólnieniu i skutecznym przyswajaniu kluczowych cech z danych RGB przez model Mask2Former. Ponadto, brak wyraźnych spadków mAP po osiągnięciu szczytowych wartości sugeruje, że model nie ulega przetrenowaniu.

W przypadku obrazów z kamery MapIR widzimy wolniejszy wzrost wartości mAP w początkowych epokach, a model osiąga maksymalną wartość około 0.30 dopiero po 20 epokach. Po tym okresie następuje stabilizacja wyników na poziomie nieco poniżej 0.30, co jest wyraźnie niższą wartością w porównaniu do danych RGB. Krzywa dla MapIR jest bardziej niestabilna w pierwszej fazie, co może sugerować, że model potrzebował więcej czasu na przyswojenie kluczowych cech z danych o specyfice spektralnej MapIR.

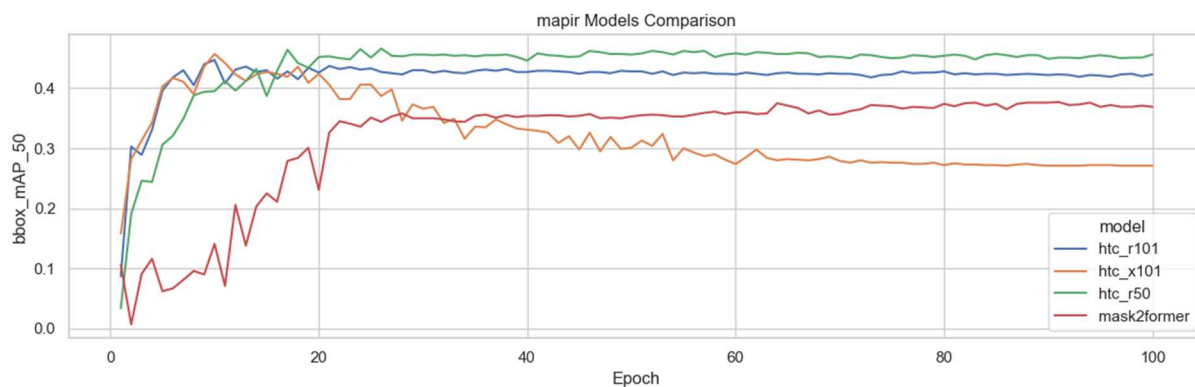
Porównując oba zestawy danych, model wyraźnie lepiej radzi sobie z obrazami RGB, osiągając wyższe wartości mAP oraz stabilniejszą krzywą treningu. Wyniki dla MapIR są niższe, co może wynikać z ograniczonej różnorodności informacji w obrazach MapIR w porównaniu do pełniejszego spektrum dostarczanego przez dane RGB.

Podsumowując, model Mask2Former wykazuje lepszą wydajność przy analizie obrazów RGB, co wskazuje, że tego typu dane dostarczają więcej przydatnych informacji dla zadania detekcji obiektów. Stabilizacja wyników po wczesnej fazie treningu dla obu typów danych świadczy o dobrej konwergencji modelu, jednak wyższa wartość mAP dla obrazów RGB sugeruje, że są one bardziej skuteczne w kontekście trenowania modelu Mask2Former.



**Rysunek 5.13.** Przebieg uczenia modeli opartych o algorytmy HTC\_r50, HTC\_r101, HTC\_x101 oraz Mask2Former dla zobrazenia RGB.

Na Rysunku 5.13 przedstawiono porównanie wyników mAP (bbox\_mAP\_50) dla czterech modeli trenowanych na danych RGB: HTC\_r50, HTC\_r101, HTC\_x101 oraz Mask2Former. Najlepsze wyniki osiąga model HTC\_r50, który stabilizuje się na poziomie około 0.45 mAP, przewyższając pozostałe modele. HTC\_r101 również osiąga dobre wyniki, jednak pozostaje poniżej HTC\_r50. Model HTC\_x101 wykazuje wyraźnie gorszą wydajność, a Mask2Former ma najniższą wartość mAP, z trudnościami w stabilizacji po początkowym wzroście. W efekcie, HTC\_r50 okazuje się najbardziej skuteczny w analizie danych RGB, osiągając najwyższe wyniki i wykazując stabilność w procesie uczenia.



**Rysunek 5.14.** Przebieg uczenia modeli opartych o algorytmy HTC\_r50, HTC\_r101, HTC\_x101 oraz Mask2Former dla zobrazenia wielospektralnego z kamery MapIR.

Na tym wykresie przedstawiono porównanie wyników mAP (bbox\_mAP\_50) dla modeli trenowanych na danych z kamery MapIR: HTC\_r50, HTC\_r101, HTC\_x101 oraz Mask2Former. HTC\_r50 ponownie osiąga najlepsze wyniki, stabilizując się na poziomie około 0.42 mAP, co czyni go najskuteczniejszym modelem na tym zestawie danych. Model HTC\_r101 radzi sobie nieco gorzej, ale utrzymuje stabilną wartość mAP blisko 0.40. Model HTC\_x101 wykazuje niższą wydajność z wartością mAP poniżej 0.35, natomiast Mask2Former osiąga najniższe wyniki, stabilizując się na poziomie około 0.30. W efekcie, podobnie jak na danych RGB, HTC\_r50 okazuje się najbardziej efektywny również na obrazach z kamery MapIR.

#### 5.4. Porównanie wyników i wybór najlepszych rozwiązań

Na podstawie przeprowadzonych eksperymentów nad klasyfikacją faz rozwoju kukurydzy w skali BBCH przy użyciu różnych modeli głębokich sieci neuronowych oraz obrazów z kamer RGB i MapIR, można sformułować kilka kluczowych wniosków. Przedstawione wyniki jasno pokazują, że model HTC\_r50 jest najbardziej efektywną architekturą zarówno w przypadku danych RGB, jak i multispektralnych MapIR. Uzyskane wyniki mAP, F1-score oraz accuracy wyraźnie wskazują, że ten model najlepiej radzi sobie z zadaniem automatycznej oceny faz rozwoju kukurydzy. Jego zdolność do generalizacji na obu typach danych sprawia, że jest on wyjątkowo skuteczny w wykrywaniu subtelnych różnic między różnymi etapami wzrostu roślin.

Dane RGB okazały się bardziej informatywne i lepiej nadają się do trenowania modeli w kontekście klasyfikacji faz BBCH. Wszystkie testowane modele, bez wyjątku, osiągają wyższe wyniki na obrazach RGB w porównaniu do obrazów z kamery MapIR. Jest to szczególnie widoczne w przypadku modelu HTC\_r50, który stabilizuje się na poziomie blisko 0.45 mAP w przypadku danych RGB, co świadczy o jego zdolności do skutecznej analizy bogatego spektrum informacji dostarczanych przez obrazy RGB. Dane MapIR, choć nadal użyteczne, nie dostarczają tak dużej ilości informacji wizualnych, co skutkuje niższymi wartościami metryk oceny dla wszystkich modeli.

Model HTC\_r101 również osiąga wysokie wyniki, szczególnie na obrazach RGB, gdzie stabilizuje się na poziomie zbliżonym do HTC\_r50, choć nieznacznie niższym. HTC\_r101 może być traktowany jako skuteczna alternatywa, zwłaszcza w przypadku konieczności dostosowania modelu do nieco innych ograniczeń, na przykład zasobów obliczeniowych. Choć wyniki tego modelu są bliskie HTC\_r50, nie osiąga on jednak takich samych poziomów



skuteczności, co wskazuje, że HTC\_r50 lepiej nadaje się do analizy obrazów w kontekście faz rozwoju kukurydzy.

Modele HTC\_x101 oraz Mask2Former okazały się mniej efektywne w porównaniu do HTC\_r50 i HTC\_r101. Choć HTC\_x101 ma nieco bardziej zaawansowaną architekturę, jego wydajność na obu zestawach danych jest zauważalnie niższa. Podobnie, model Mask2Former, który jest zoptymalizowany pod kątem zadań segmentacji, nie wykazuje dużej skuteczności w kontekście klasyfikacji faz wzrostu kukurydzy. Jest to wynik, który można przypisać specyficznej charakterystyce tych modeli, które mogą być mniej optymalne w zadaniach związanych z klasyfikacją, gdzie precyzja detekcji cech wizualnych jest kluczowa.

Stabilność wyników po pierwszych 20-30 epokach dla większości modeli sugeruje, że proces uczenia został przeprowadzony prawidłowo, bez oznak przetrenowania. Krzywe mAP wskazują, że modele szybko przyswajają informacje z danych w początkowych etapach treningu, a następnie stabilizują się na różnych poziomach w zależności od efektywności architektury. Stabilne wartości metryk takich jak mAP i F1-score dowodzą, że modele te potrafią generalizować nauczone cechy i skutecznie klasyfikować różne fazy rozwoju roślin.

Podsumowując, wyniki eksperymentów jednoznacznie wskazują, że wybór odpowiedniego modelu ma kluczowe znaczenie w kontekście automatycznej klasyfikacji faz wzrostu kukurydzy w skali BBCH. Model HTC\_r50 okazał się najbardziej efektywny w obu analizowanych przypadkach, jednak jego przewaga jest szczególnie widoczna na obrazach RGB. Te dane dostarczają więcej informacji wizualnych, co przekłada się na wyższe wartości mAP i F1-score, kluczowych dla skutecznej klasyfikacji faz rozwoju. W zadaniach takich jak to, gdzie liczy się detekcja i klasyfikacja subtelnych różnic między fazami wzrostu, model HTC\_r50 w połączeniu z danymi RGB okazuje się najlepszym rozwiązaniem, co czyni go wiodącym narzędziem w analizie faz rozwojowych kukurydzy dla potrzeb rolnictwa precyzyjnego.

## 6. Detekcja i klasyfikacja poziomów nawodnienia kukurydzy

Zadanie związane z detekcją i klasyfikacją poziomów nawodnienia kukurydzy miało na celu określenie, w jakim przedziale nawodnienia znajduje się widoczna na wskazanym obrazie roślina. Jako podstawową skalę, dla której prowadzono badania, przyjęto trzy poziomy nawodnienia: *niski*, *średni* i *wysoki poziom nawodnienia*, gdzie *niski poziom nawodnienia* oznaczał wystawienie roślin na warunki znacznie ograniczonej dostępności do wody, wręcz w warunkach suszy fizjologicznej, *średni poziom nawodnienia* wiązał się z umiarkowanym dostępem roślin do wody, a *wysoki poziom nawodnienia* wiązał się z utrzymywaniem stałego optymalnego poziomu wilgotności. Badania prowadzono w warunkach glebowych charakterystycznych dla obszaru centralnej Polski, a dokładnie dla miejscowości Borowiec (gmina Tarczyn, województwo mazowieckie). Celem nadrzędnym zadania było zaś zapewnienie dodatkowej funkcjonalności opracowanego systemu monitoringu kondycji kukurydzy polegającej na identyfikacji poziomów nawodnienia badanych roślin i przypisaniu im odpowiednich prawdopodobieństw detekcji na poszczególnych częściach pola, z dokładnością do częstotliwości pobierania jednego obrazu.

W kolejnych sekcjach tego rozdziału zaprezentowany zostanie schemat zaproponowanego rozwiązania, wyniki uzyskane dla danych wielospektralnych i dla danych RGB, przebieg krzywych uczenia dla obu typów danych przy zastosowaniu badanych algorytmów uczenia głębokich sztucznych sieci neuronowych oraz porównanie wyników i wybór najlepszych rozwiązań.

### 6.1. Schemat zaproponowanego rozwiązania

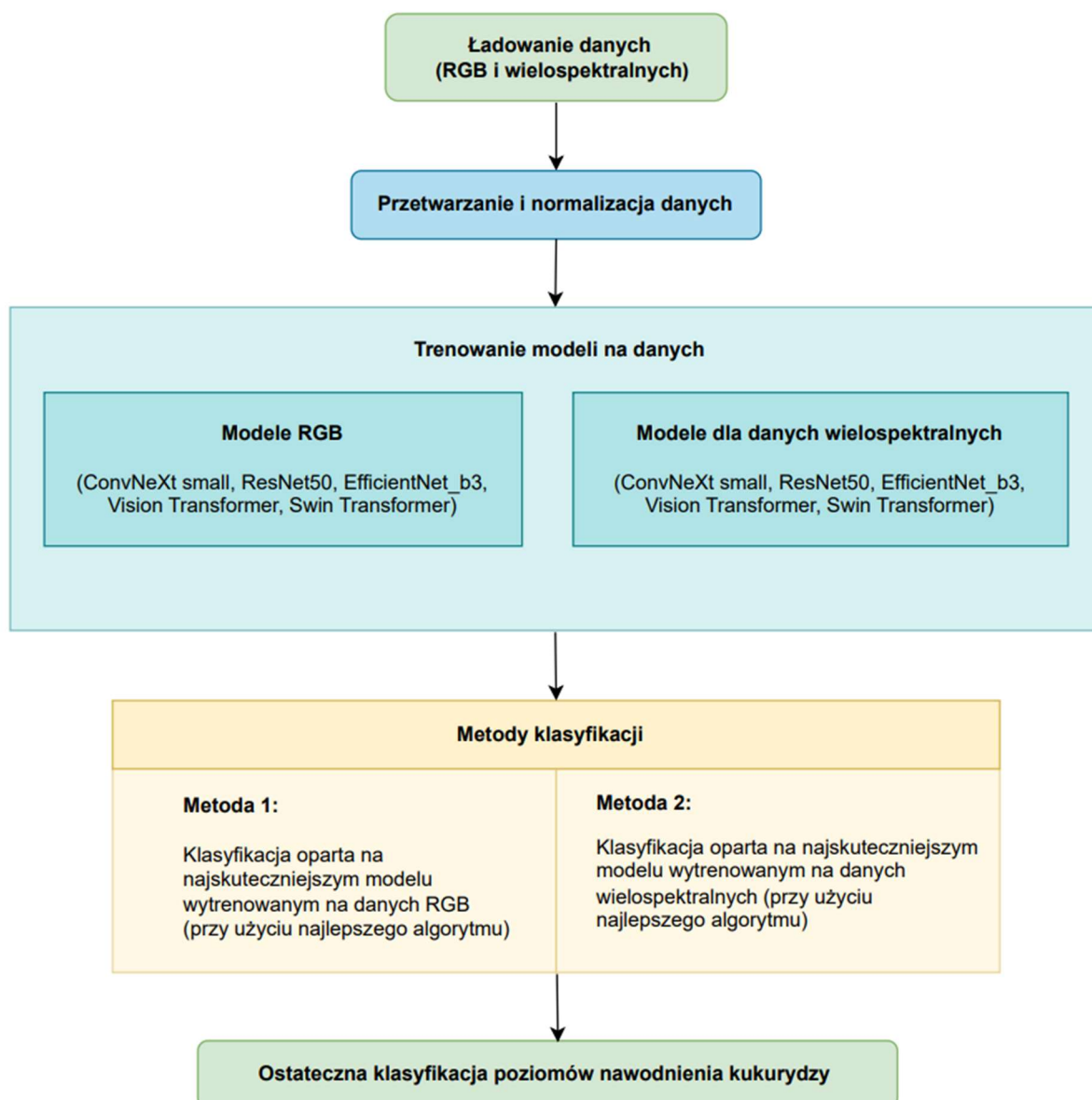
W sekcji tej przedstawiony zostanie szczegółowy opis rozwiązań użytych podczas realizacji badania dotyczącego detekcji i klasyfikacji poziomów nawodnienia kukurydzy. Rysunek 6.1. pokazuje ogólny schemat zaproponowanego systemu.

Schemat ten przedstawia dwa główne podejścia zastosowane w opracowanym systemie: klasyfikacja z użyciem obrazów RGB oraz klasyfikacja z użyciem obrazów wielospektralnych z kamery MapIR. Rozwiązania te zaproponowano, aby sprawdzić i porównać wyniki otrzymane przy zastosowaniu standardowego obrazowania RGB oraz zobrazowania pozyskiwanego za pomocą kamery wielospektralnej MapIR. Porównanie obu podejść pozwoliło na uzyskanie odpowiedzi na pytanie który typ zobrazowań zapewni większą

skuteczność procesu detekcji i klasyfikacji poziomów nawodnienia kukurydzy z wykorzystaniem tych samych analogicznych modeli uczenia głębokiego. Prace te miały typowy charakter badawczo-wdrożeniowy z potencjałem na wykorzystanie uzyskanych wyników również podczas podobnych zadań prowadzonych w innych projektach.

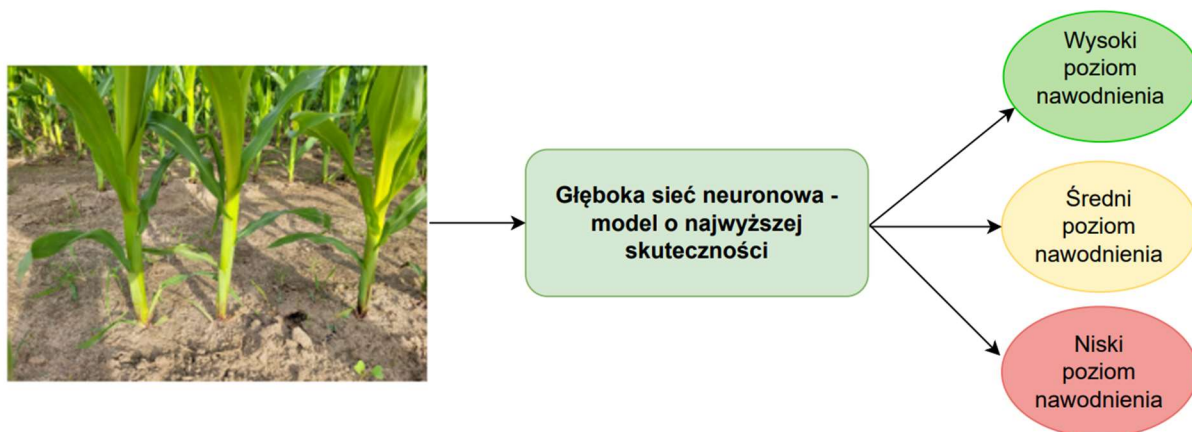
Zgodnie ze schematem, najpierw następuje proces ładowania danych (tutaj należy rozważyć konkretny typ danych: RGB lub wielospektralnych), po czym dane te są poddawane odpowiedniemu przetwarzaniu i normalizacji. W następnym kroku następuje etap trenowania głębokich sztucznych sieci neuronowych na bazie wczytanych obrazów z użyciem zaimplementowanych architektur, którymi są: ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer, a każda z tych architektur traktowana jest oddzielnie i prowadzi do stworzenia osobnego modelu. Po etapie trenowania, analizowaniu poszczególnych modeli i wyborze najlepszego algorytmu odpowiedniego dla danego typu zobrazowania, następnym kluczowym procesem jest klasyfikacja z użyciem najlepszego modelu. W tym momencie analizowane są dwa główne podejścia.

W pierwszym podejściu (Metoda 1) została zastosowana klasyfikacja obrazów RGB z wykorzystaniem wybranego algorytmu, spośród puli algorytmów analizowanych podczas badań. Drugie podejście (Metoda 2) polega na analogicznym wykorzystaniu tych samych algorytmów, ale przy zastosowaniu obrazów z kamery MapIR. Finalna klasyfikacja opiera się o wybór najskuteczniejszej metody wykorzystującej jedno ze zobrazowań: RGB lub z kamery MapIR oraz o algorytm, który osiągnął najlepsze wyniki spośród wszystkich analizowanych algorytmów.



**Rysunek 6.1.** Schemat zaproponowanego rozwiązania w ramach realizacji zadania detekcji i klasyfikacji poziomów nawodnienia kukurydzy.

Rysunek 6.2. pokazuje z kolei schemat przepływu danych w rozwiązaniu opracowanym dla problemu określania poziomów nawodnienia kukurydzy. Na wejściu podawany jest obraz, który następnie jest przetwarzany przez model oparty o głęboką sieć neuronową, a na wyjściu następuje przypisanie do poszczególnych klas z uwzględnieniem odpowiedniej wartości prawdopodobieństwa.



**Rysunek 6.2.** Schemat przepływu danych w rozwiązaniu opracowanym dla problemu określania poziomów nawodnienia kukurydzy.

Wyniki uzyskane dla poszczególnych zobrażeń i wytrenowanych z ich użyciem modeli uczenia głębokiego zostaną omówione w kolejnych podrozdziałach, aby prześledzić krok po kroku proces wyboru najlepszego rozwiązania.

## 6.2. Wyniki uzyskane dla danych wielospektralnych

W przypadku analizy obrazów wielospektralnych z kamery MapIR podczas prowadzenia badań zastosowano dane, które zostały omówione we wcześniejszej sekcji niniejszej rozprawy (podrozdział 3.2.5. *Dane wielospektralne z kamery MapIR*). Użyte zaś algorytmy głębokich sztucznych sieci neuronowych to: ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer, a implementacja tychże architektur i szczegóły dotyczące procesu trenowania omówione zostały w podrozdziałach 3.3.4. *Implementacja modeli ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer, Swin Transformer* oraz 3.4.2. *Trenowanie modeli ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer, Swin Transformer*.

Poniżej znajduje się tabela z porównaniem wyników uzyskanych przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer. Do porównania zastosowano typowe metryki: Accuracy, Precision, Recall i F1-Score.

| Model              | Accuracy | Precision | Recall | F1-Score |
|--------------------|----------|-----------|--------|----------|
| ConvNeXt Small     | 0.8548   | 0.8358    | 0.8409 | 0.8372   |
| ResNet50           | 0.7635   | 0.7852    | 0.7298 | 0.7301   |
| EfficientNet B3    | 0.7718   | 0.7469    | 0.7410 | 0.7426   |
| Vision Transformer | 0.8257   | 0.8363    | 0.8085 | 0.8105   |
| Swin Transformer   | 0.8631   | 0.8527    | 0.8493 | 0.8506   |

**Tabela 6.1.** Porównanie wyników uzyskanych dla danych wielospektralnych z kamery MapIR przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer.

Jak widać na bazie analizy zaprezentowanych wyników, najlepiej z detekcją i klasyfikacją poziomów nawodnienia kukurydzy na bazie obrazów z kamery wielospektralnej MapIR poradził sobie algorytm Swin Transformer, który uzyskał wartość parametru Accuracy rzędu 0.8631. Tuż za nim jest algorytm ConvNeXt Small z wartością Accuracy rzędu 0.8548. Dalej znajdują się kolejno algorytmy Vision Transformer, EfficientNet\_B3 i ResNet50 z wartościami metryki Accuracy kolejno: 0.8257, 0.7718 i 0.7635.

### 6.3. Wyniki uzyskane dla danych RGB

W przypadku analizy obrazów RGB podczas prowadzenia badań zastosowano dane, które zostały omówione we wcześniejszej sekcji niniejszej rozprawy (podrozdział 3.2.3. *Dane RGB*). Użyte zaś algorytmy głębokich sztucznych sieci neuronowych to identycznie jak w przypadku danych wielospektralnych z kamery MapIR: ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer, a implementacja tychże architektur i szczegóły dotyczące procesu trenowania omówione zostały w podrozdziałach 3.3.4. *Implementacja modeli ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer, Swin Transformer* oraz 3.4.2. *Trenowanie modeli ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer, Swin Transformer*.

Poniżej znajduje się tabela z porównaniem wyników uzyskanych przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer. Do porównania zastosowano typowe metryki: Accuracy, Precision, Recall i F1-Score.



| Model              | Accuracy | Precision | Recall | F1-Score |
|--------------------|----------|-----------|--------|----------|
| ConvNeXt Small     | 0.8880   | 0.8751    | 0.8752 | 0.8747   |
| ResNet50           | 0.7552   | 0.8125    | 0.7282 | 0.7208   |
| EfficientNet B3    | 0.8091   | 0.7926    | 0.7899 | 0.7911   |
| Vision Transformer | 0.8506   | 0.8345    | 0.8373 | 0.8298   |
| Swin Transformer   | 0.8382   | 0.8191    | 0.8122 | 0.8149   |

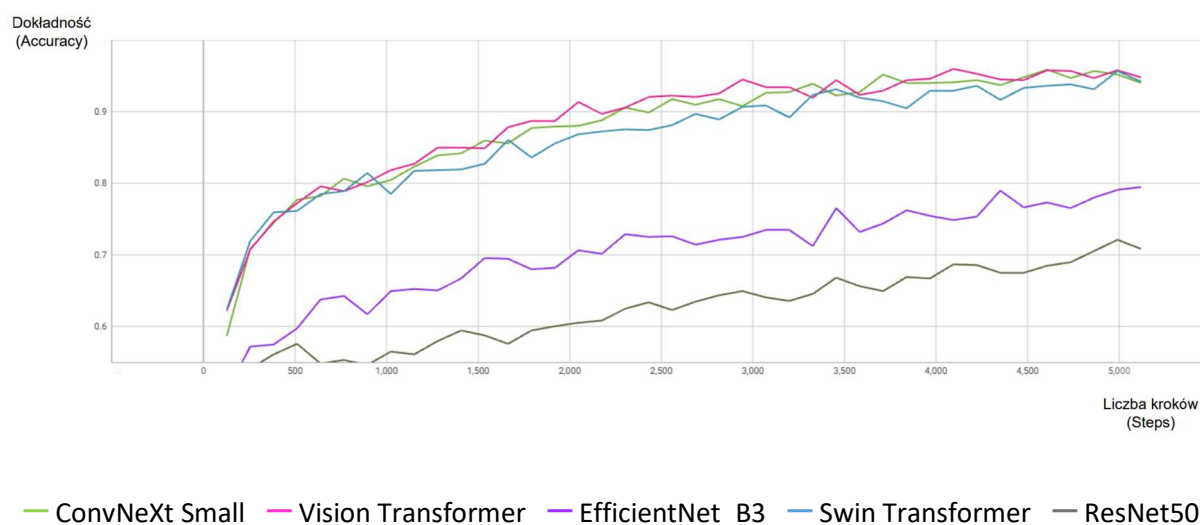
**Tabela 6.2.** Porównanie wyników uzyskanych dla danych RGB przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer.

Jak widać na bazie analizy zaprezentowanych wyników, najlepiej z detekcją i klasyfikacją poziomów nawodnienia kukurydzy na bazie obrazów RGB poradził sobie algorytm ConvNeXt Small, który uzyskał wartość parametru Accuracy rzędu 0.8880. Tuż za nim jest algorytm Vision Transformer z wartością Accuracy rzędu 0.8506. Dalej znajdują się kolejno algorytmy Swin Transformer, EfficientNet\_B3 i ResNet50 z wartościami metryki Accuracy kolejno: 0.8382, 0.8091 i 0.7552.

#### 6.4. Krzywe uczenia dla danych wielospektralnych i RGB

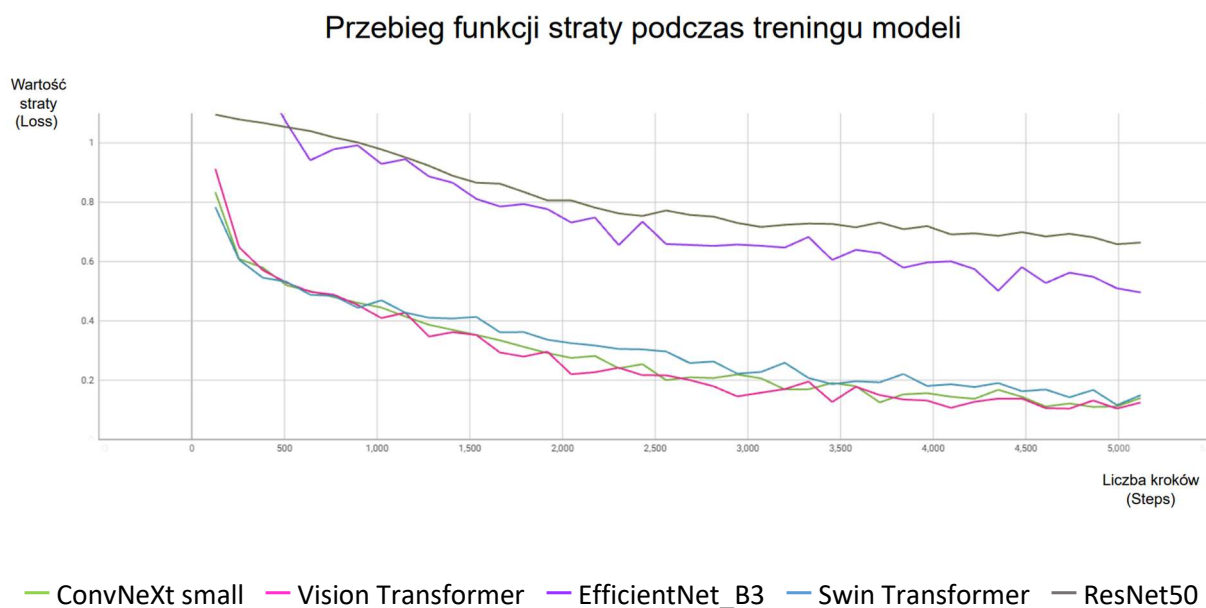
W tej części niniejszej rozprawy zaprezentowane zostaną krzywe uczenia zarejestrowane dla poszczególnych algorytmów służących do trenowania głębokich sztucznych sieci neuronowych w celu realizacji zadania polegającego na detekcji i klasyfikacji poziomów nawodnienia kukurydzy. Poniżej krzywych zamieszczone zostało podsumowanie z analizą porównawczą tychże krzywych.

### Dokładność modeli podczas treningu



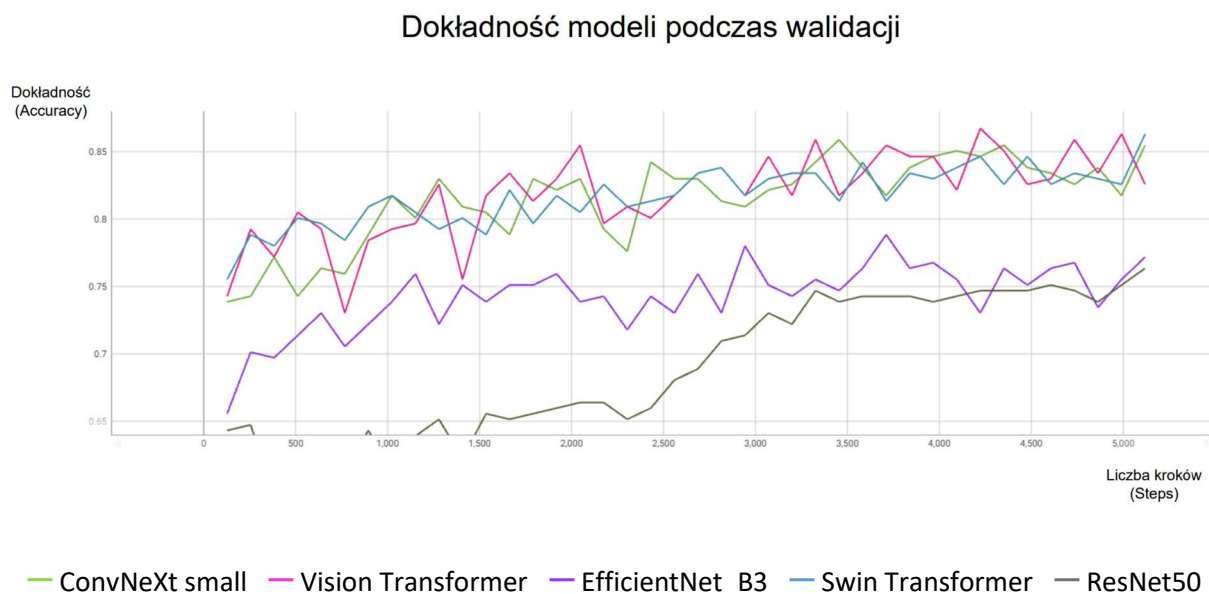
**Rysunek 6.3.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas treningu na danych z kamery MapIR.

Rysunek 6.3. przedstawia zmianę dokładności (Accuracy) modeli podczas treningu w zależności od liczby kroków (Steps). Wykres obrazuje, jak poszczególne algorytmy osiągały postępy w trakcie procesu uczenia. Algorytm Vision Transformer osiągnął najwyższą dokładność, przekraczając wartość 0.9 już po około 1000 krokach, a następnie utrzymywał wysoką dokładność przez resztę treningu, osiągając stabilizację w okolicach 0.92. Algorytm ConvNeXt Small również uzyskał wysoką dokładność, zbliżając się do wyniku różowego algorytmu. Ostatecznie jego dokładność ustabilizowała się nieco poniżej 0.9. Algorytm Swin Transformer osiągnął podobne wyniki do zielonego, jednak z nieco większymi wahaniami w trakcie treningu. Ostatecznie jego dokładność również ustabilizowała się w okolicach 0.9. Algorytm EfficientNet\_B3 początkowo osiągał umiarkowane wyniki, jednak w dalszej części treningu wykazywał stopniowy wzrost dokładności, kończąc z wynikiem powyżej 0.8. Algorytm ResNet50 osiągnął najniższą dokładność spośród wszystkich algorytmów, z wartością nieprzekraczającą 0.7 przez cały okres treningu. Z wykresu wynika, że algorytm Vision Transformer wykazał się największą stabilnością i szybkością w osiągnięciu wysokiej dokładności, podczas gdy czarny algorytm radził sobie najslabiej.



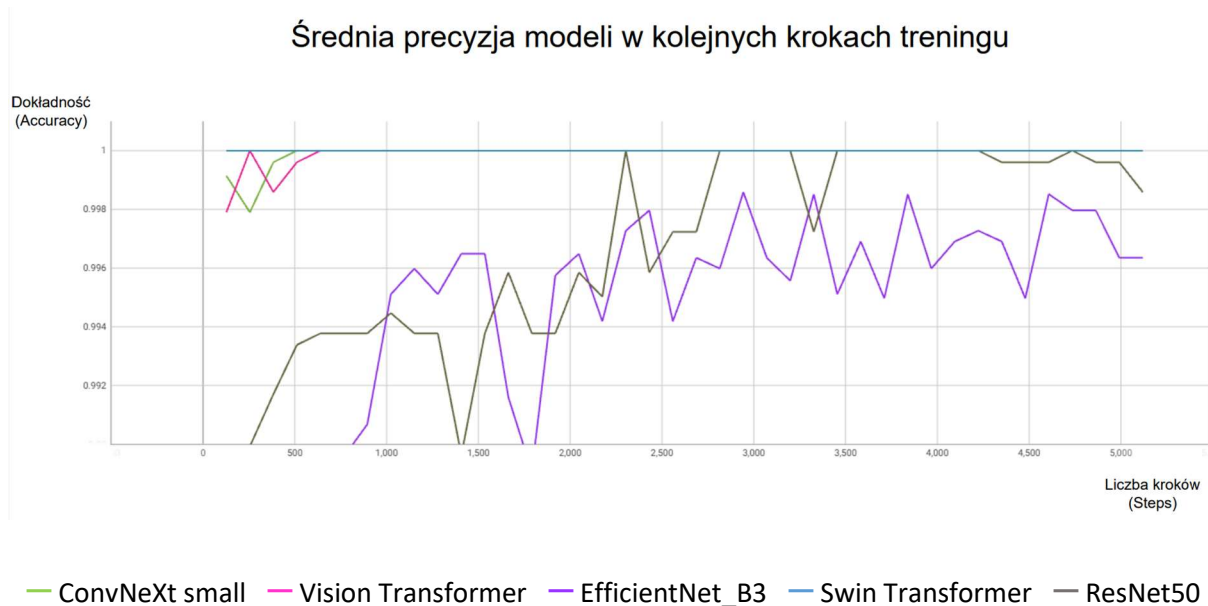
**Rysunek 6.4.** Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu na danych z kamery MapIR.

Na Rysunku 6.4. przedstawiono przebieg funkcji straty (Loss) w trakcie treningu modeli w zależności od liczby kroków (Steps). Wartość straty jest odwrotnie proporcjonalna do dokładności modeli, a jej zmniejszenie wskazuje na poprawę wydajności modelu. Algorytm Vision Transformer charakteryzował się najszybszym spadkiem wartości straty na początku treningu, szybko osiągając wartości poniżej 0.2, a następnie utrzymując tę wartość na niskim poziomie do końca treningu. Algorytm ConvNeXt Small również wykazywał szybki spadek straty, osiągając wartości podobne do różowego algorytmu, z końcową stratą oscylującą wokół 0.2. Algorytm Swin Transformer miał nieco większe wahania wartości straty, ale ostatecznie zbliżył się do poziomu zielonego i różowego algorytmu, osiągając podobną końcową wartość straty. Algorytm EfficientNet\_B3 wykazywał umiarkowany spadek wartości straty, utrzymując się na poziomie powyżej 0.4 przez większość treningu, ale również wykazał tendencję spadkową. Algorytm ResNet50 algorytm miał najwyższą wartość straty przez cały czas trwania treningu, z początkową stratą powyżej 1.0 i stopniowym, ale wolnym jej spadkiem do poziomu około 0.6 na końcu treningu. Z wykresu wynika, że algorytmy Vision Transformer i ConvNeXt Small uzyskały najlepsze wyniki, osiągając najniższe wartości straty, co sugeruje, że były one najbardziej efektywne w procesie uczenia się w porównaniu do pozostałych algorytmów. Algorytm ResNet50 wykazał się najniższą efektywnością, co znajduje odzwierciedlenie w jego najwyższej wartości straty.



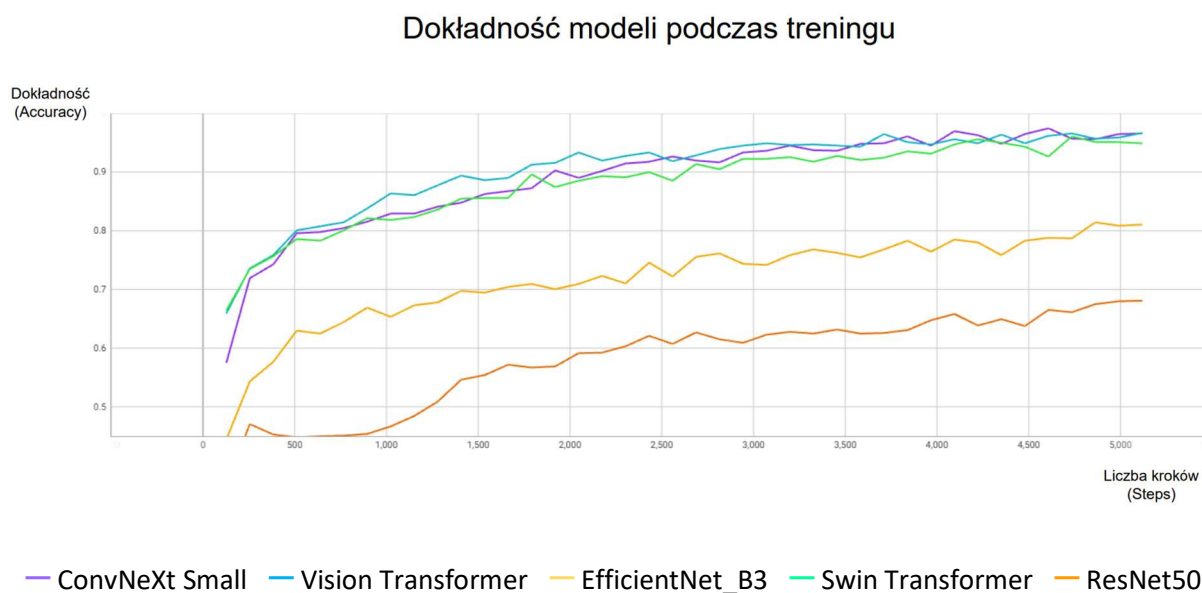
**Rysunek 6.5.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas procesu walidacji na danych z kamery MapIR.

Rysunek 6.5 przedstawia dokładność (Accuracy) modeli podczas walidacji w zależności od liczby kroków (Steps). Algorytm Vision Transformer charakteryzował się największymi wahaniami dokładności, jednak w większości przypadków uzyskiwał najwyższe wartości, osiągając około 0.85-0.88 na końcu procesu walidacji. Algorytmy ConvNeXt Small i Swin Transformer miały podobny przebieg walidacji, również wykazując pewne wahania, ale kończąc na stabilnym poziomie około 0.82-0.85. Algorytm EfficientNet\_B3 miał niższą dokładność na początku walidacji, lecz w miarę postępu treningu osiągał stabilny wzrost, kończąc na poziomie około 0.75. Algorytm ResNet50 wykazywał najniższą dokładność spośród wszystkich, z trudnościami w przekroczeniu wartości 0.7 przez większość czasu walidacji. Dopiero pod koniec walidacji jego dokładność zaczęła rosnąć, osiągając wartość około 0.7 na ostatnich krokach. Wnioskując z wykresu, algorytm Vision Transformer uzyskał najlepsze wyniki w walidacji, co wskazuje na jego wysoką skuteczność. Algorytmy ConvNeXt Small i Swin Transformer również uzyskały dobre wyniki, natomiast algorytm ResNet50 radził sobie najgorzej.



**Rysunek 6.6.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich średniej precyzji podczas treningu na danych z kamery MapIR.

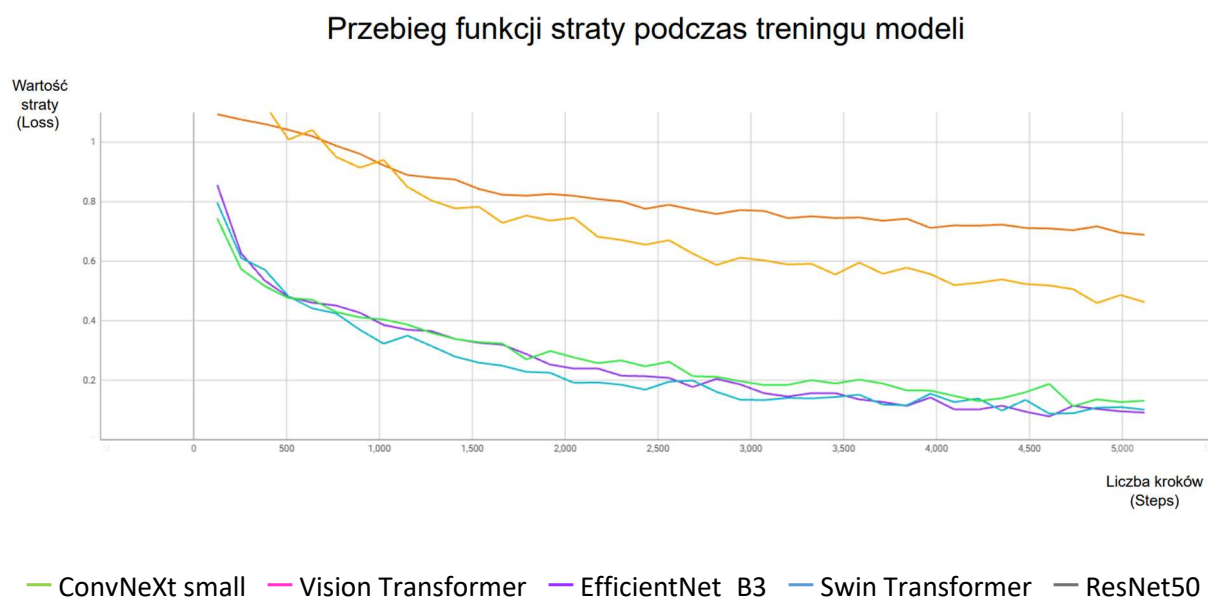
Rysunek 6.6 przedstawia średnią precyzję modeli w kolejnych krokach treningu, mierzoną jako dokładność (Accuracy) w zależności od liczby kroków (Steps). Algorytm Swin Transformer wykazał stałą i wysoką precyzję przez cały proces treningu, utrzymując wartość bardzo bliską 1.0, co oznacza, że model ten był niezwykle dokładny od samego początku. Algorytmy Vision Transformer i ConvNeXt Small osiągnęły bardzo wysoką precyzję już na początku treningu, z wynikami bliskimi 1.0, i utrzymywały ten poziom przez pierwsze 500 kroków. Algorytmy ResNet50 i EfficientNet\_B3 wykazały znacznie większe wahania precyzji. ResNet50 początkowo uzyskał wysoką precyzję, jednak z biegiem treningu wartości te spadły, wykazując duże wahania. EfficientNet\_B3 miał najniższą precyzję spośród wszystkich modeli, z wyraźnymi wahaniami i brakiem stabilności w wynikach, które nigdy nie osiągnęły wartości powyżej 0.995. Z wykresu wynika, że algorytm Swin Transformer był najbardziej stabilny i precyzyjny w całym procesie treningu, natomiast algorytmy ResNet50 i EfficientNet\_b3 miały największe problemy z utrzymaniem wysokiej precyzji.



**Rysunek 6.7.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas treningu na danych z kamery RGB.

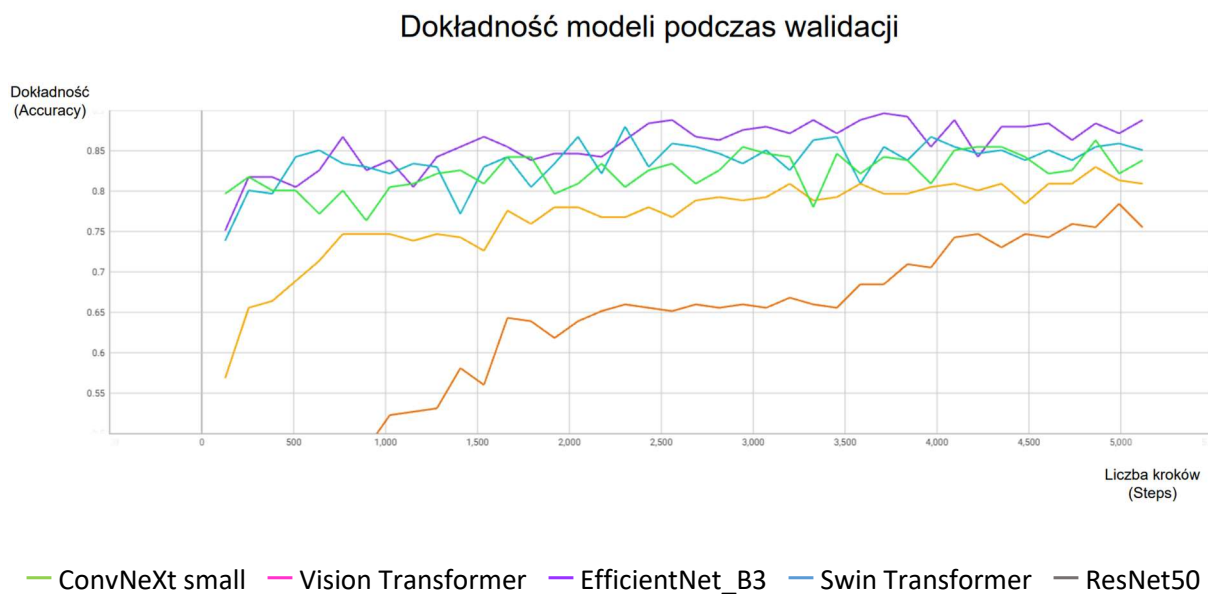
Na rysunku 6.7 przedstawiono dokładność (Accuracy) modeli podczas treningu w zależności od liczby kroków (Steps). Algorytm Vision Transformer osiągnął najwyższą dokładność, stabilizując się na poziomie bliskim 0.9 po około 2000 krokach i utrzymując tę wartość do końca treningu. Algorytm Swin Transformer również osiągnął wysoką dokładność, bardzo zbliżoną do Vision Transformer, z niewielkimi wahaniami, stabilizując się także w okolicach 0.9. Algorytm ConvNeXt Small początkowo miał większe wahania, ale ostatecznie osiągnął bardzo zbliżone wyniki do Vision Transformer i Swin Transformer, kończąc trening z dokładnością również bliską 0.9. Algorytm EfficientNet\_B3 wykazał umiarkowany wzrost dokładności, kończąc trening z wartością około 0.75, co oznacza, że jego skuteczność była średnia w porównaniu z pozostałymi modelami. Algorytm ResNet50 miał najniższą dokładność spośród wszystkich modeli, osiągając na końcu treningu wartość poniżej 0.7, co sugeruje, że jego skuteczność była zdecydowanie niższa. Z wykresu wynika, że Vision Transformer, Swin Transformer i ConvNeXt Small uzyskały najlepsze wyniki, podczas gdy ResNet50 radził sobie najgorzej w procesie uczenia się.





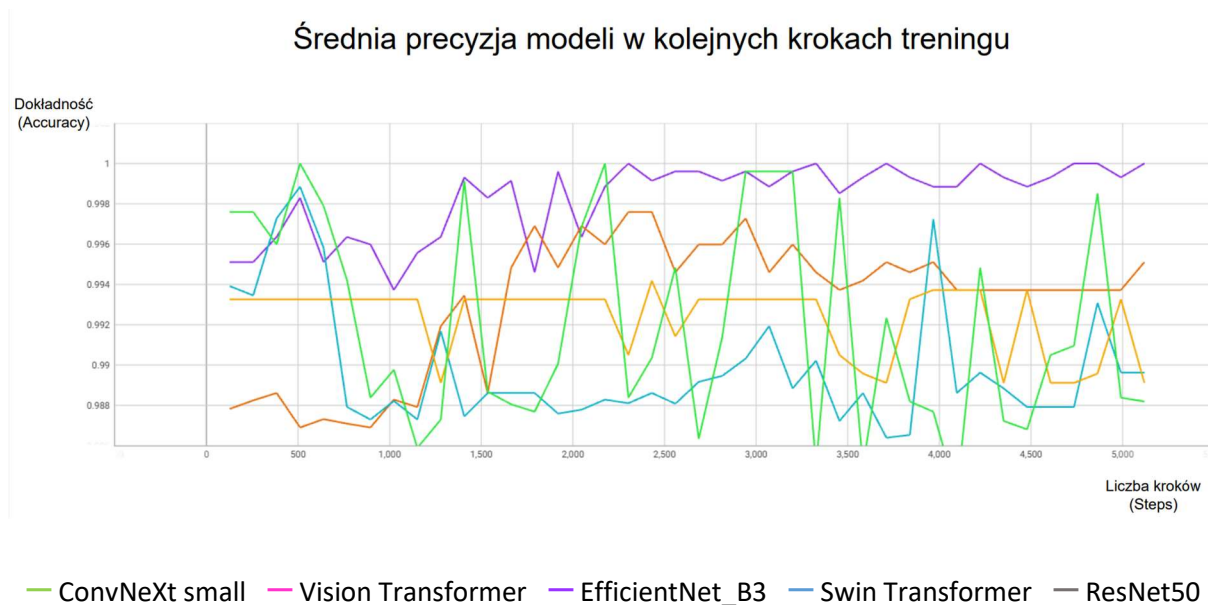
**Rysunek 6.8.** Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu na danych z kamery RGB.

Rysunek 6.8 przedstawia przebieg funkcji straty (Loss) podczas treningu modeli w zależności od liczby kroków (Steps). Algorytmy ConvNeXt Small, Vision Transformer i Swin Transformer wykazały najszybszy spadek wartości straty, osiągając najniższe wartości już po około 1000 krokach i utrzymując je na niskim poziomie do końca treningu. Wartość straty dla tych algorytmów oscylowała w okolicach 0.1-0.2 na końcowych etapach treningu. Algorytm EfficientNet\_B3 miał umiarkowany spadek wartości straty, ale jego wyniki były wyraźnie gorsze od ConvNeXt Small, Vision Transformer i Swin Transformer, z wartością straty oscylującą w okolicach 0.5-0.6 na końcu treningu. Algorytm ResNet50 wykazał najwolniejszy spadek wartości straty, utrzymując ją na stosunkowo wysokim poziomie przez większość treningu. Pod koniec treningu wartość straty dla tego algorytmu wynosiła około 0.8, co sugeruje, że model miał trudności z efektywnym uczeniem się. Z wykresu wynika, że ConvNeXt Small, Vision Transformer i Swin Transformer były najbardziej efektywne w minimalizowaniu straty podczas treningu, natomiast ResNet50 miał największe trudności z redukcją straty, co może przekładać się na jego niższą dokładność.



**Rysunek 6.9.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas procesu walidacji na danych z kamery RGB.

Rysunek 6.9 przedstawia wykres dokładności (Accuracy) modeli podczas walidacji w zależności od liczby kroków (Steps). Algorytm ConvNeXt Small osiągnął najwyższą dokładność podczas walidacji, stabilizując się w okolicach 0.85-0.87. Jest to najwyższa wartość spośród wszystkich analizowanych modeli. Algorytmy Vision Transformer i Swin Transformer osiągnęły bardzo podobne wyniki, z dokładnością oscylującą w okolicach 0.82-0.85. Oba modele wykazały się stabilnością i niewielkimi wahaniami. Algorytm EfficientNet\_B3 miał średnią dokładność w walidacji, stabilizując się na poziomie około 0.75. Choć jego wyniki były gorsze od trzech poprzednich algorytmów, były stabilne przez większość czasu. Algorytm ResNet50 osiągnął najniższą dokładność, z wynikami oscylującymi wokół 0.65-0.7. Jego wartość była wyraźnie niższa od pozostałych modeli, co sugeruje, że miał trudności z prawidłową klasyfikacją podczas walidacji. Z wykresu wynika, że ConvNeXt Small był najskuteczniejszy podczas walidacji, a ResNet50 miał największe trudności z osiągnięciem wysokiej dokładności.



**Rysunek 6.10.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich średniej precyzji podczas treningu na danych z kamery RGB.

Rysunek 6.10 pokazuje średnią precyzję modeli w kolejnych krokach treningu, mierzoną jako dokładność (Accuracy) w zależności od liczby kroków (Steps). Algorytm ConvNeXt Small wykazał się największą stabilnością, utrzymując wysoką precyzję przez cały czas trwania treningu. Jego dokładność była bardzo bliska wartości 1.0 i wykazywała minimalne wahania, co wskazuje na jego wyjątkową skuteczność w uczeniu się. Algorytm Swin Transformer miał duże wahania precyzji na początku treningu, ale z czasem stabilizował się na poziomie bliskim 0.995, chociaż wykazywał sporadyczne spadki i wzrosty. Algorytm Vision Transformer również wykazał się wysoką precyzją, jednak z nieco większymi wahaniami w porównaniu do ConvNeXt Small. Jego dokładność oscylowała w okolicach 0.995, z okresowymi spadkami. Algorytm EfficientNet\_B3 charakteryzował się niestabilną precyzją, z dużymi wahaniami w trakcie treningu. Jego dokładność często spadała poniżej 0.99, co wskazuje na trudności w stabilnym uczeniu się. Algorytm ResNet50 miał najniższą precyzję spośród wszystkich, wykazując znaczne wahania oraz liczne spadki precyzji, często poniżej wartości 0.99. Z wykresu wynika, że ConvNeXt Small był najbardziej efektywny i stabilny w treningu, podczas gdy ResNet50 miał największe problemy z utrzymaniem wysokiej precyzji, co sugeruje jego gorszą jakość w procesie uczenia się.

## 6.5. Porównanie wyników i wybór najlepszych rozwiązań

W oparciu o analizę krzywych uczenia oraz wartości poszczególnych metryk oceny algorytmów można wywnioskować, iż dla zobrazowania RGB najlepsze wyniki osiągnął algorytm EfficientNet\_B3, z kolei dla danych wielospektralnych pochodzących z kamery MapIR najlepsze wyniki pozwolił osiągnąć algorytm Vision Transformer. Jeżeli chodzi o wyniki globalne, to zostały one osiągnięte dla danych RGB, przy zastosowaniu algorytmu fioletowego. Daje to jasny sygnał, iż w przypadku analizowanej puli algorytmów najlepsze wyniki można uzyskać przy użyciu kamery RGB.

## 7. Detekcja i klasyfikacja porażenia kukurydzy

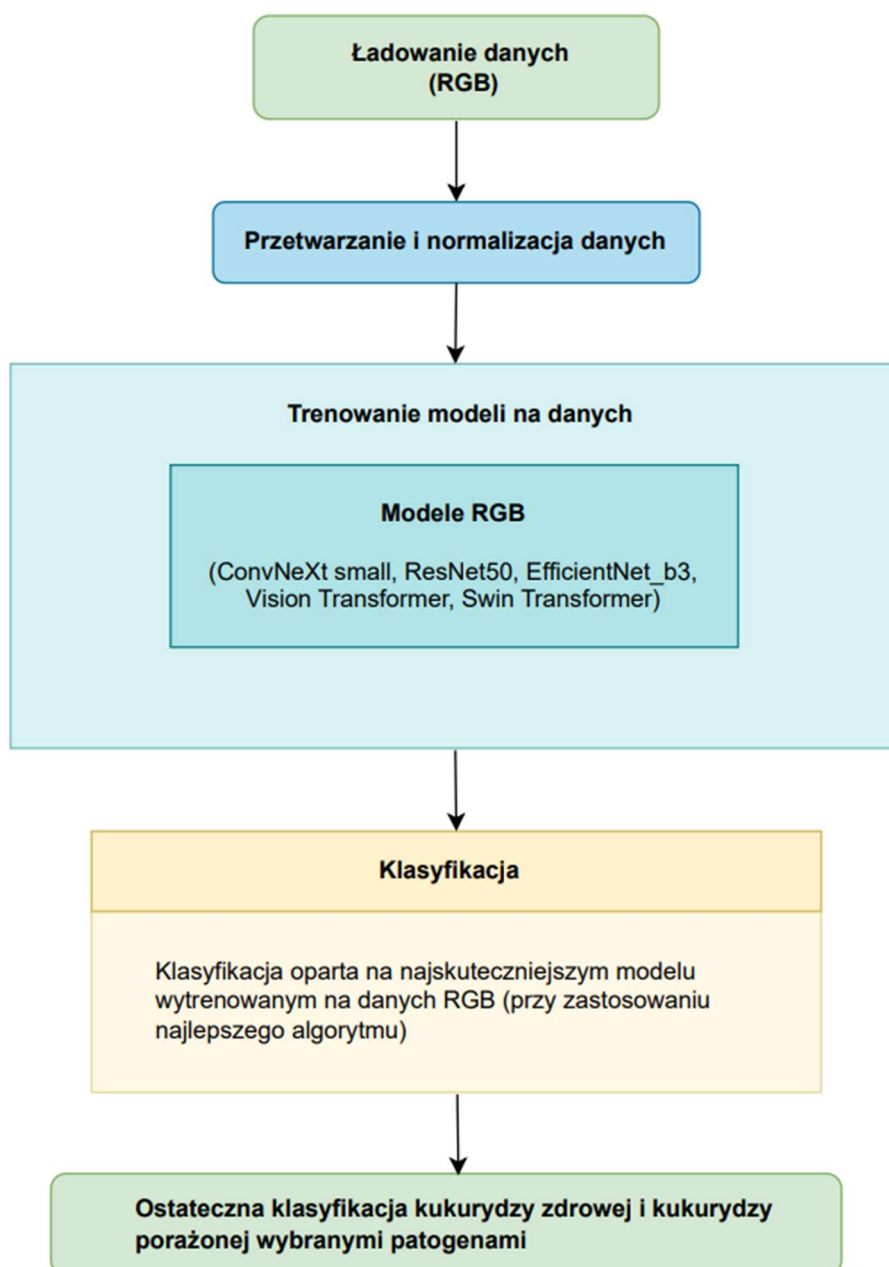
Zadanie związane z detekcją i klasyfikacją kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami miało na celu określenie, czy widoczna na wskazanym obrazie roślina jest zdrowa, czy porażona, a jeśli porażona to jakim patogenem. Jako analizowane przypadki porażenia rozważone zostały następujące przypadki: *kukurydza zdrowa*, *kukurydza porażona patogenem gąsiennicy spodoptera frugiperda*, *kukurydza porażona patogenem helmintosporiozy*, *kukurydza porażona patogenem rdzy pospolitej* i *kukurydza porażona patogenem szarej plamistości*.

Badania prowadzono w warunkach glebowych charakterystycznych dla obszaru centralnej Polski, a dokładnie dla miejscowości Borowiec (gmina Tarczyn, województwo mazowieckie). Celem nadrzędnym zadania było zaś zapewnienie dodatkowej funkcjonalności opracowanego systemu monitoringu kondycji kukurydzy polegającej na identyfikacji kukurydzy zdrowej oraz potencjalnie porażonej wybranymi patogenami i przypisaniu jej odpowiednich prawdopodobieństw detekcji na poszczególnych częściach pola, z dokładnością do częstotliwości pobierania jednego obrazu.

W kolejnych sekcjach tego rozdziału zaprezentowany zostanie schemat zaproponowanego rozwiązania, wyniki uzyskane dla danych RGB pozyskanych z zewnętrznego datasetu, przebieg krzywych uczenia dla obu typów danych przy zastosowaniu badanych algorytmów uczenia głębokich sztucznych sieci neuronowych oraz porównanie wyników i wybór najlepszych rozwiązań.

### 7.1. Schemat zaproponowanego rozwiązania

W podrozdziale tym zaprezentowany zostanie opis rozwiązań użytych podczas realizacji badania dotyczącego detekcji i klasyfikacji kukurydzy zdrowej oraz kukurydzy porażonej wybranymi patogenami. Rysunek 7.1. pokazuje ogólny schemat zaproponowanego systemu.



**Rysunek 7.1.** Schemat zaproponowanego rozwiązania w ramach realizacji zadania detekcji i klasyfikacji kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami.

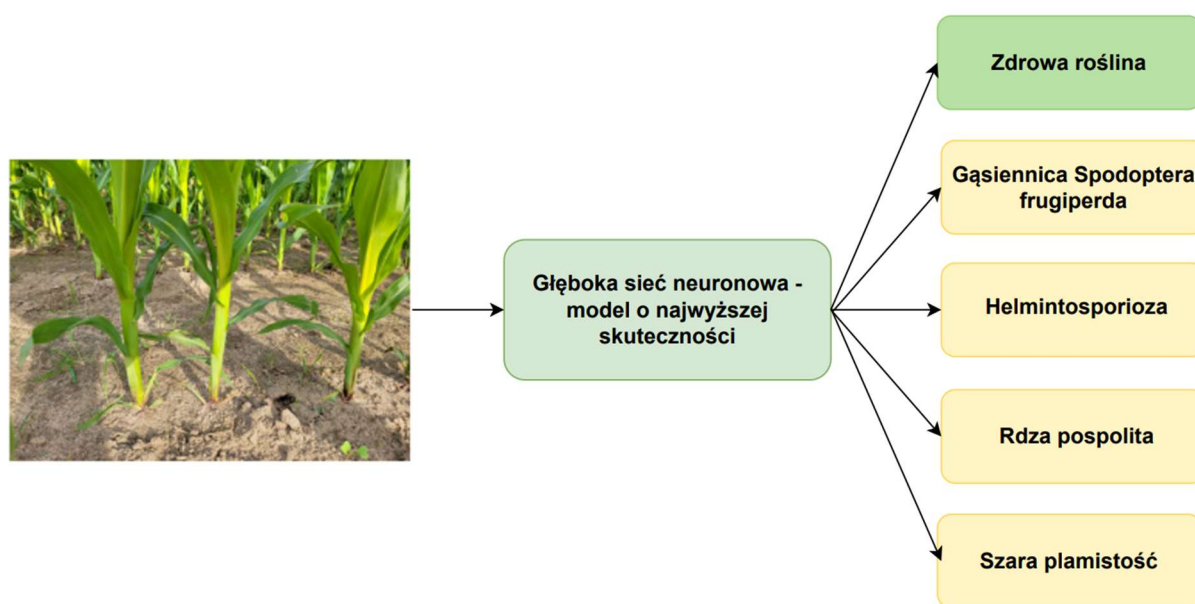
Schemat ten przedstawia główne podejście zastosowane w opracowanym systemie, a mianowicie podejście z użyciem obrazów RGB. Rozwiązanie z użyciem wyłącznie obrazowania RGB zaproponowano w oparciu o dostępność odpowiednich danych treningowych wyłącznie w formacie RGB. Do realizacji tego zadania wykorzystano natomiast różne architektury uczenia głębokiego.



Zgodnie ze schematem, najpierw następuje proces ładowania danych RGB, po czym dane te są poddawane odpowiedniemu przetwarzaniu i normalizacji. W następnym kroku następuje etap trenowania głębokich sztucznych sieci neuronowych na bazie wczytanych obrazów z użyciem zaimplementowanych architektur, którymi są: ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer, a każda z tych architektur traktowana jest oddzielnie i prowadzi do stworzenia osobnego modelu. Po etapie trenowania, analizowaniu poszczególnych modeli i wyborze najlepszego algorytmu, następnym kluczowym procesem jest klasyfikacja z użyciem najlepszego modelu. Analizowane jest tutaj podejście z wykorzystaniem wyłącznie obrazów RGB, gdyż patogeny inakulowane na polu testowym Sieci Badawczej Łukasiewicz – Instytutu Lotnictwa nie rozwinęły się ze względu na panującą suszę. Do badań wykorzystano zatem zewnętrzny dataset zawierający obrazy kukurydzy porażonej wybranymi patogenami.

Finalna klasyfikacja opiera się na wyborze najskuteczniejszej metody wykorzystującej zobrazowanie RGB oraz architekturę uczenia głębokiego, która osiągnęła najlepsze wyniki spośród wszystkich analizowanych architektur.

Rysunek 7.2. pokazuje z kolei schemat przepływu danych w rozwiązaniu opracowanym dla problemu określania porażenia kukurydzy wybranymi patogenami. Na wejściu podawany jest obraz, który następnie jest przetwarzany przez model oparty o głęboką sieć neuronową, a na wyjściu następuje przypisanie do poszczególnych klas z uwzględnieniem odpowiedniej wartości prawdopodobieństwa.



**Rysunek 7.2.** Schemat przepływu danych w rozwiązaniu opracowanym dla problemu określania porażenia kukurydzy wybranymi patogenami.

Wyniki uzyskane dla poszczególnych zobrazowań i wytrenowanych z ich użyciem modeli zostaną omówione w kolejnych podrozdziałach, aby prześledzić wybór najlepszego rozwiązania.

## 7.2. Wyniki i ich analiza

W przypadku analizy obrazów podczas prowadzenia badań zastosowano dane, które zostały omówione we wcześniejszej sekcji niniejszej rozprawy (podrozdział 3.2.3. *Dane RGB*). Użyte zaś algorytmy głębokich sztucznych sieci neuronowych to: ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer, a implementacja tychże architektur i szczegóły dotyczące procesu trenowania omówione zostały w podrozdziałach 3.3.4. *Implementacja modeli ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer, Swin Transformer* oraz 3.4.2. *Trenowanie modeli ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer, Swin Transformer*.

Poniżej znajduje się tabela z porównaniem wyników uzyskanych przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer. Do porównania zastosowano typowe metryki: Accuracy, Precision, Recall i F1-Score.

| Model              | Accuracy | Precision | Recall | F1-Score |
|--------------------|----------|-----------|--------|----------|
| ConvNeXt Small     | 0.9969   | 0.9930    | 0.9921 | 0.9926   |
| ResNet50           | 0.4416   | 0.4411    | 0.4171 | 0.3439   |
| EfficientNet B3    | 0.9951   | 0.9902    | 0.9905 | 0.9904   |
| Vision Transformer | 0.9988   | 0.9962    | 0.9980 | 0.9971   |
| Swin Transformer   | 0.9994   | 0.9981    | 0.9990 | 0.9985   |

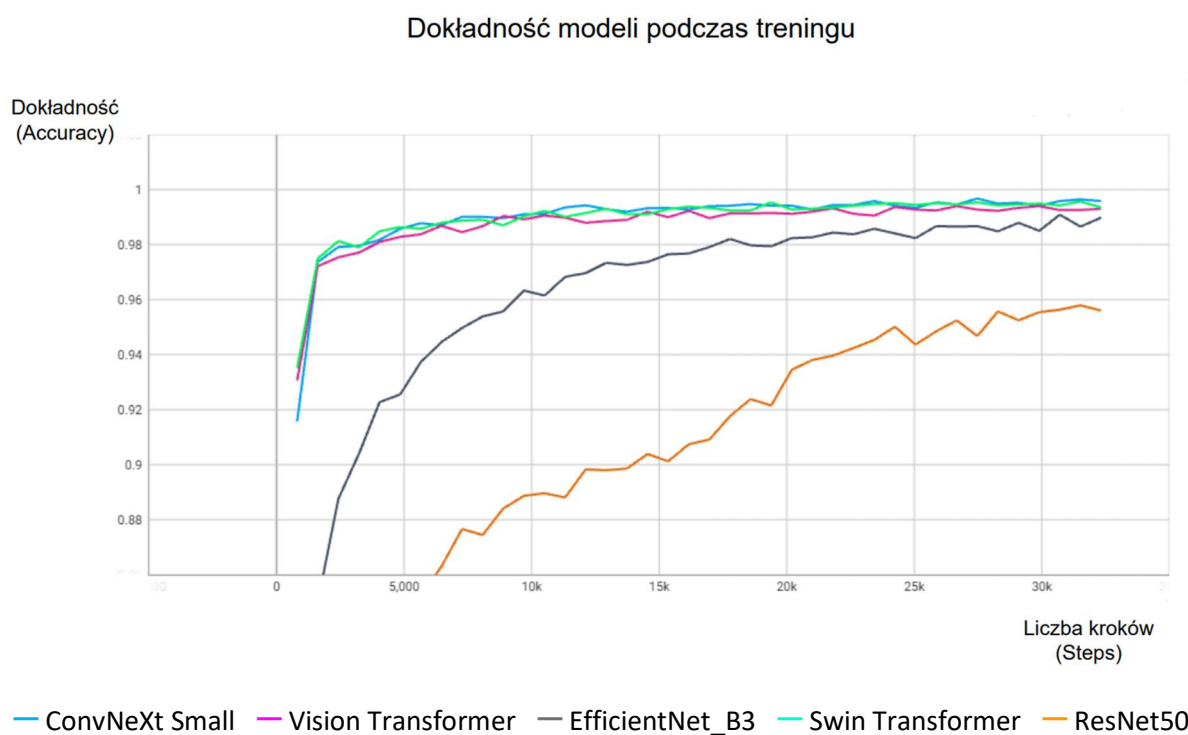
**Tabela 7.1.** Porównanie wyników uzyskanych dla danych RGB przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer i Swin Transformer.

Jak widać na bazie analizy zaprezentowanych wyników, najlepiej z detekcją i klasyfikacją porażenia kukurydzy poradził sobie algorytm Swin Transformer, który uzyskał wartość parametru Accuracy rzędu 0.9994. Tuż za nim jest algorytm Vision Transformer z

wartością Accuracy rzędu 0.9988. Dalej znajdują się kolejno algorytmy ConvNeXt Small, EfficientNet\_B3 i ResNet50 z wartościami metryki Accuracy kolejno: 0.9969, 0.9951 i 0.4416.

### 7.3. Krzywe uczenia

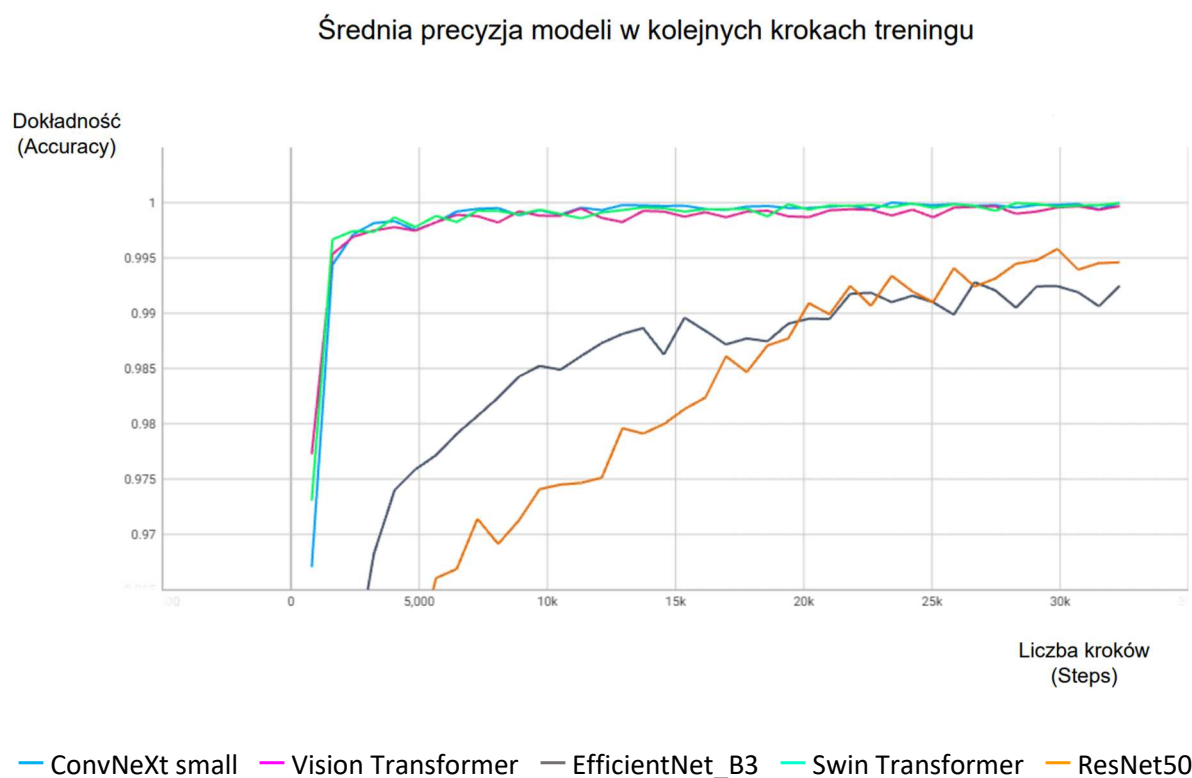
W tej części niniejszej rozprawy zaprezentowane zostaną krzywe uczenia zarejestrowane dla poszczególnych algorytmów służących do trenowania głębokich sztucznych sieci neuronowych w celu realizacji zadania polegającego na detekcji i klasyfikacji porażenia kukurydzy. Poniżej krzywych zamieszczone zostało podsumowanie z analizą porównawczą tychże krzywych.



**Rysunek 7.3.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas treningu,

Rysunek 7.3 przedstawia dokładność (Accuracy) modeli podczas treningu w zależności od liczby kroków (Steps). Algorytm ConvNeXt Small osiągnął najwyższą dokładność już na wczesnym etapie treningu, stabilizując się na poziomie bliskim 1.0 po około 5 tysiącach kroków i utrzymując tę wartość do końca treningu. Algorytmy Vision Transformer i Swin Transformer

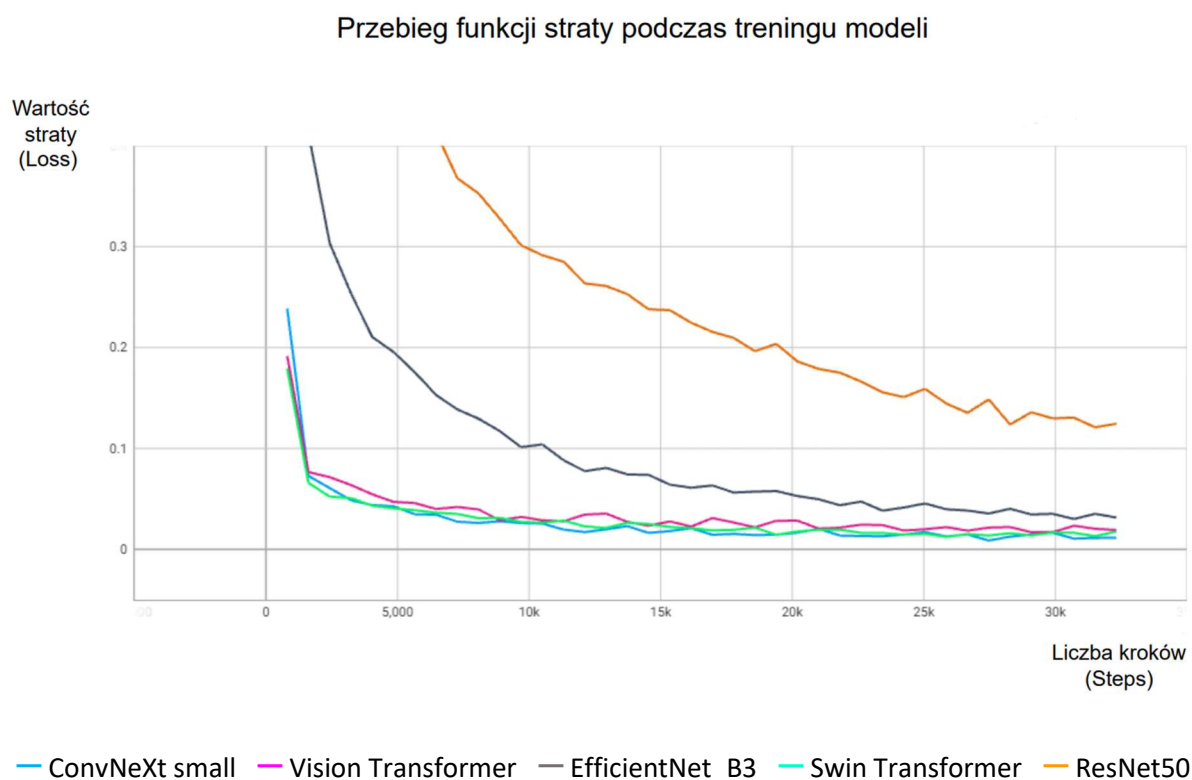
wykazywały bardzo podobny przebieg treningu, osiągając wysoką dokładność bliską 1.0, podobnie jak algorytm ConvNeXt Small. Oba modele wykazały się stabilnością i wysoką skutecznością przez większość trwania treningu. Algorytm EfficientNet\_B3 początkowo osiągał niższą dokładność, ale stopniowo poprawiał swoje wyniki, osiągając ostatecznie poziom około 0.98. Jego postęp był bardziej stopniowy w porównaniu do Vision Transformer, Swin Transformer i ConvNeXt Small. Algorytm ResNet50 wykazywał najniższą dokładność spośród wszystkich analizowanych modeli. Jego wyniki poprawiały się stopniowo, ale ostatecznie osiągnął dokładność jedynie około 0.93, co sugeruje, że miał trudności z osiągnięciem wysokiej skuteczności w porównaniu do pozostałych modeli. Podsumowując, ConvNeXt Small, Vision Transformer i Swin Transformer osiągnęły najwyższą dokładność i stabilność w trakcie treningu, natomiast ResNet50 miał największe trudności z osiągnięciem porównywalnych wyników.



**Rysunek 7.4.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich średniej precyzji podczas treningu.

Rysunek 7.4 przedstawia średnią precyzję modeli w kolejnych krokach treningu, mierzoną jako dokładność (Accuracy) w zależności od liczby kroków (Steps). Algorytmy ConvNeXt Small, Vision Transformer, i Swin Transformer wykazały się najwyższą precyzją

już na początku treningu, osiągając bardzo szybko wartości bliskie 1.0. Wszystkie trzy algorytmy utrzymały tę precyzję przez cały czas trwania treningu, wykazując jedynie minimalne wahania. Algorytm EfficientNet\_B3 początkowo osiągał niższą precyzję, jednak z biegiem treningu jego wartość stopniowo rosła, osiągając około 0.985 na końcu procesu. Mimo to, nie zdołał on osiągnąć poziomu precyzji zbliżonego do najlepszych modeli. Algorytm ResNet50 miał najniższą precyzję spośród wszystkich modeli na wykresie. Chociaż jego precyzja stopniowo rosła, osiągnęła wartość nieco poniżej 0.98 na końcu treningu, co nadal było znacznie niższe niż precyzja pozostałych algorytmów. Podsumowując, algorytmy ConvNeXt Small, Vision Transformer i Swin Transformer osiągnęły najwyższą precyzję i wykazały się dużą stabilnością w trakcie treningu, natomiast ResNet50 miał największe trudności z osiągnięciem wysokiej precyzji.

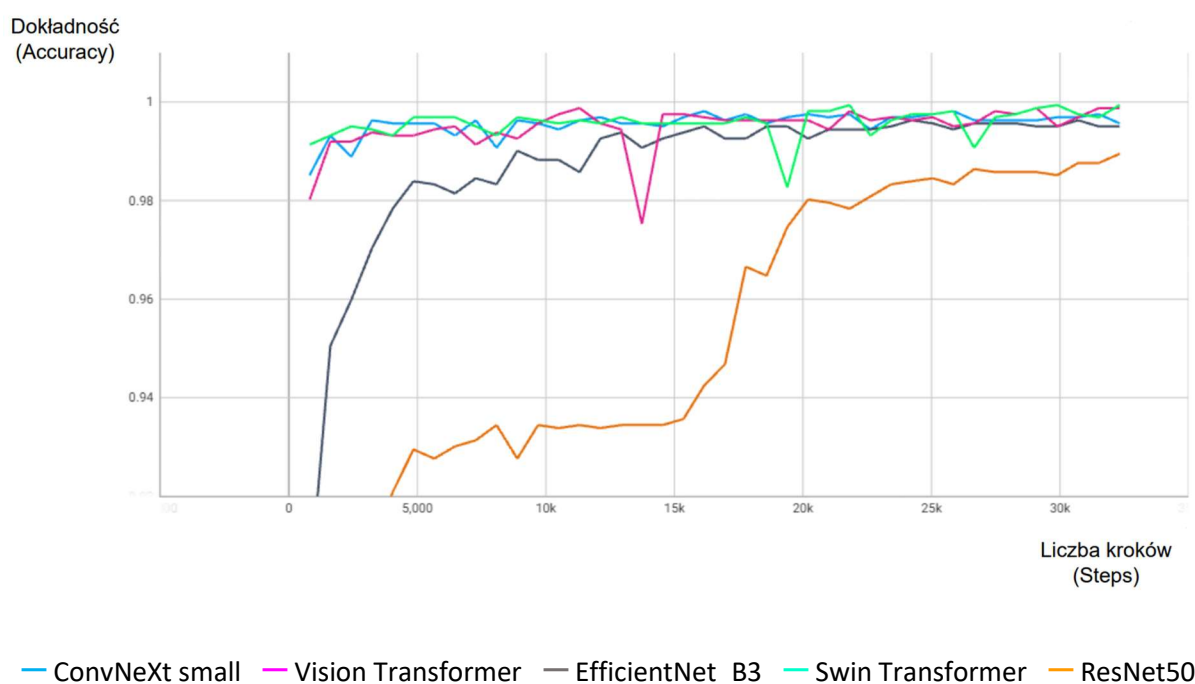


**Rysunek 7.5.** Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu.

Na zaprezentowanym wykresie przedstawiony jest przebieg funkcji straty (Loss) podczas treningu modeli w zależności od liczby kroków (Steps). Algorytmy ConvNeXt Small,

Vision Transformer, i Swin Transformer wykazały się najniższymi wartościami straty od samego początku treningu. Wszystkie trzy modele bardzo szybko osiągnęły niskie wartości straty i utrzymywały je na stabilnym poziomie, z minimalnymi wahaniami, oscylując w okolicach 0.05 lub nawet niżej. Algorytm EfficientNet\_B3 początkowo miał wyższą wartość straty, ale z czasem jego strata sukcesywnie malała, osiągając stabilny poziom około 0.1. Choć jego wynik jest nieco wyższy od wyników ConvNeXt Small, Vision Transformer i Swin Transformer, to jego strata była znacznie niższa od modelu ResNet50. Algorytm ResNet50 wykazywał najwyższą wartość straty spośród wszystkich modeli. Choć jego strata stopniowo malała, to pozostała na znacznie wyższym poziomie niż inne modele, kończąc trening z wartością straty powyżej 0.3. Podsumowując, algorytmy ConvNeXt Small, Vision Transformer i Swin Transformer były najbardziej efektywne w minimalizowaniu funkcji straty podczas treningu, natomiast algorytm ResNet50 miał największe trudności z redukcją straty, co może wskazywać na niższą skuteczność w procesie uczenia się.

Dokładność modeli podczas walidacji



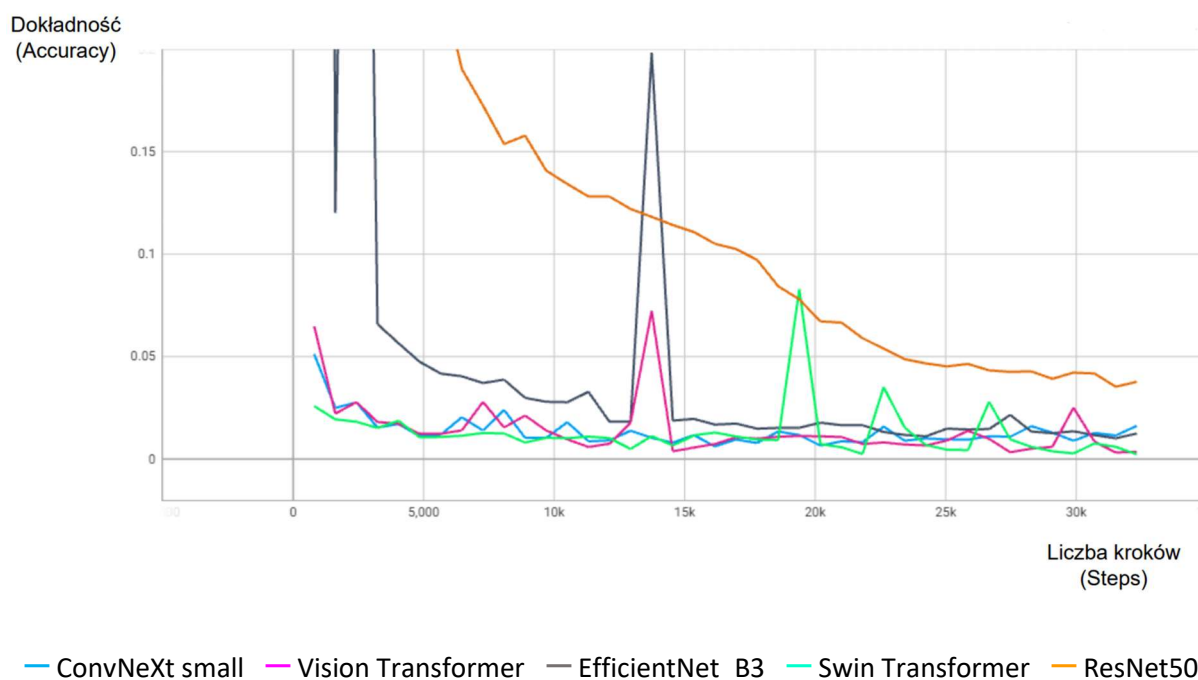
**Rysunek 7.6.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas procesu walidacji.

Wykres przedstawia dokładność (Accuracy) modeli podczas walidacji w zależności od liczby kroków (Steps). Algorytmy ConvNeXt Small, Vision Transformer, i Swin Transformer



wykazały się najwyższą dokładnością podczas walidacji, z wynikami bliskimi 1.0 przez większość czasu. Chociaż występowały drobne wahania, wszystkie trzy modele utrzymywały bardzo wysoką dokładność na przestrzeni całego treningu. Algorytm EfficientNet\_B3 początkowo osiągał niższą dokładność, jednak w pierwszej części treningu szybko zwiększył swoją dokładność, stabilizując się na poziomie około 0.98. Jego wyniki były stabilne, ale nie osiągnął tak wysokiej dokładności jak ConvNeXt Small, Vision Transformer i Swin Transformer. Algorytm ResNet50 miał najniższą dokładność spośród wszystkich analizowanych modeli. Jego dokładność rosła stopniowo, ale ostatecznie osiągnęła wartość nieco poniżej 0.96, co jest znacząco niższym wynikiem w porównaniu do pozostałych modeli. Podsumowując, algorytmy ConvNeXt Small, Vision Transformer i Swin Transformer były najsukuteczniejsze podczas walidacji, osiągając najwyższą dokładność. Algorytm EfficientNet\_B3 radził sobie dobrze, ale nie osiągnął poziomu najlepszych modeli, natomiast ResNet50 miał największe trudności z osiągnięciem wysokiej dokładności.

Przebieg funkcji straty podczas walidacji modeli



**Rysunek 7.7.** Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas walidacji.

Rysunek 7.7. przedstawia wykres przebiegu funkcji straty (Loss) podczas walidacji modeli w zależności od liczby kroków (Steps). Algorytmy ConvNeXt Small, Swin

Transformer, i Vision Transformer wykazały się najniższymi wartościami straty podczas walidacji. Wartości straty dla tych modeli były bardzo niskie przez większość czasu trwania walidacji, oscylując blisko zera, z niewielkimi i sporadycznymi wahaniami. Algorytm EfficientNet\_B3 wykazywał większe wahania wartości straty, szczególnie widoczne są skoki około 10,000 i 15,000 kroków, gdzie wartość straty znacząco wzrosła. Chociaż później strata spadała, te wahania sugerują niestabilność modelu podczas walidacji. Algorytm ResNet50 miał najwyższą wartość straty spośród wszystkich analizowanych modeli. Strata stopniowo malała, ale pozostała na wyraźnie wyższym poziomie w porównaniu do pozostałych modeli, co wskazuje na jego mniejszą skuteczność podczas walidacji. Podsumowując, algorytmy ConvNeXt Small, Swin Transformer i Vision Transformer były najbardziej efektywne w minimalizowaniu funkcji straty podczas walidacji, natomiast EfficientNet\_B3 wykazywał niestabilność, a ResNet50 miał największe trudności z redukcją straty.

#### 7.4. Porównanie wyników i wybór najlepszego algorytmu

W oparciu o analizę krzywych uczenia oraz wartości poszczególnych metryk oceny algorytmów można wywnioskować, iż najlepsze wyniki osiągnął algorytm Swin Transformer, tuż przed algorytmami Vision Transformer, ConvNeXt small, EfficientNet\_B3. Z kolei algorytm ResNet50 poradził sobie wyraźnie gorzej niż pozostałe algorytmy.

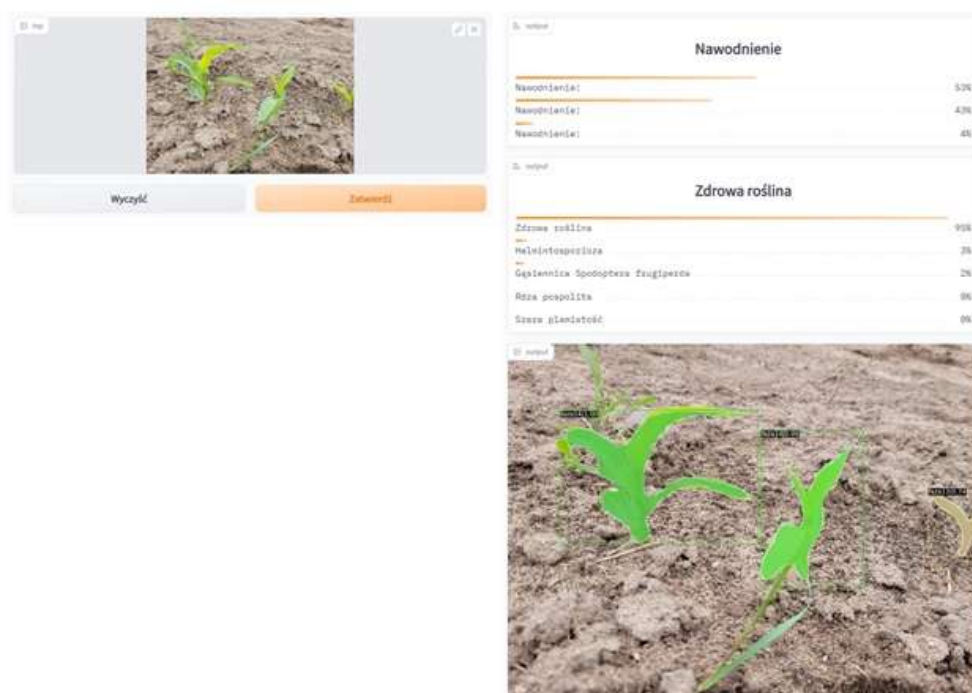
## 8. Prezentacja wyników

Wszystkie zadania projektowe będące przedmiotem niniejszej rozprawy, czyli detekcja kukurydzy i klasyfikacja jej faz rozwojowych, detekcja i klasyfikacja poziomów nawodnienia kukurydzy oraz detekcja i klasyfikacja kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami zostały zrealizowane, zatem niezwykle ważna stała się odpowiednia prezentacja otrzymanych wyników. Została ona zrealizowana dwutorowo: poprzez wizualizację w aplikacji Gradio oraz poprzez generowanie wyjściowego pliku w formacie .csv zawierającego wszystkie odpowiedzi zestawu wytrenowanych głębokich sztucznych sieci neuronowych wraz z niezbędnymi parametrami przetwarzania. W kolejnych podrozdziałach zaprezentowano oba podejścia oraz uzyskane w ten sposób wyniki.

### 8.1. Aplikacja Gradio

Wizualną prezentację wyników działania wytrenowanych modeli głębokich sieci neuronowych opracowanych w ramach niniejszej rozprawy zrealizowano przy użyciu biblioteki Gradio. Jest to biblioteka języka Python typu open source, która służy do tworzenia wizualnych demonstracji uczenia maszynowego. Pozwala ona na tworzenie interfejsu użytkownika dla modeli uczenia maszynowego i pozwala wypróbować działanie wytrenowanych modeli sieci neuronowych poprzez przeglądarkę. Taki był też cel zastosowania tejże aplikacji, aby w charakterze kontrolnym można było zbadać działanie wytrenowanych modeli na wybranych obrazach.

Dzięki bibliotece Gradio udało się stworzyć aplikację, która pozwala na interakcję użytkownika z wytrenowanymi modelami głębokich sieci neuronowych. Obecne w niej moduły pozwalają na wczytanie obrazu oraz przeanalizowanie go przez zestaw wytrenowanych modeli głębokich sieci neuronowych. Na wyjściu otrzymuje się odpowiedzi sieci w postaci wartości prawdopodobieństw wykrycia poszczególnych klas obiektów wyrażone w procentach. Dodatkowo, dla zadania związanego z wykrywaniem i klasyfikacją faz rozwojowych kukurydzy można obejrzeć wizualizację graficzną w postaci tzw. bounding boxów zaznaczonych na analizowanym obrazie. Poniższe rysunki prezentują zrzuty ekranu z przetworzeń, które można otrzymać w aplikacji. Na rysunkach poniżej przedstawiono wyniki analizy poziomów nawodnienia kukurydzy, wyniki analizy kukurydzy zdrowej oraz porażonej wybranymi patogenami oraz wyniki detekcji faz rozwojowych kukurydzy dla przykładowego obrazu RGB.



**Rysunek 8.1.** Wygląd całego interfejsu stworzonej aplikacji - przypadek analizy obrazu RGB.

## 8.2. Plik wyjściowy w formacie .csv

Oprócz strony wizualnej prezentującej wyniki działania zestawu wytrenowanych głębokich sztucznych sieci neuronowych stworzono również kod, który z odpowiedzi poszczególnych wytrenowanych modeli generuje wyjściowy plik w formacie .csv, który jest kluczowy dla monitorowania prowadzonych analiz i generowania danych, które następnie mogą trafić do bazy danych i być dalej przetwarzane. Pozwala on na agregację wszystkich odpowiedzi sieci w jednym pliku, dla wielu przetworzeń. Dodatkowo, znajdują się w nim parametry działania całego systemu w postaci m.in. nazw obrazów wraz z ich ścieżkami, wartości czasu przetwarzania oraz wartości parametru timestamp, który pozwala na opracowanie geolokalizacji obrazów w późniejszym etapie. Plik ten jest uzupełniany przez kolejne wiersze w ramach postępowania procesu przetwarzania obrazów, które trafiają na wejście zestawu wytrenowanych głębokich sieci neuronowych. W pliku .csv znajdują się następujące kolumny:

- Filename – mówiąca o nazwie obrazu
- Nawodnienie niskie – oznaczająca wartość prawdopodobieństwa wykrycia nawodnienia przypisanego do klasy „niskie”

- Nawodnienie średnie – oznaczająca wartość prawdopodobieństwa wykrycia nawodnienia przypisanego do klasy „średnie”
- Nawodnienie wysokie – oznaczająca wartość prawdopodobieństwa wykrycia nawodnienia przypisanego do klasy „wysokie”
- Porażenie Gąsiennica Spodoptera frugiperda – oznaczająca wartość prawdopodobieństwa wykrycia porażenia gąsienicą Spodoptera frugiperda
- Porażenie Helminthosporioza – oznaczająca wartość prawdopodobieństwa wykrycia porażenia grzybami z rodzaju Helminthosporioza
- Porażenie Rdza pospolita – oznaczająca wartość prawdopodobieństwa wykrycia porażenia rdzą pospolitą
- Porażenie Szara plamistość – oznaczająca wartość prawdopodobieństwa wykrycia porażenia szarą plamistością
- Porażenie Zdrowa roślina – oznaczająca wartość prawdopodobieństwa wykrycia zdrowej rośliny
- faza10 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 10 w skali BBCH
- faza11 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 11 w skali BBCH
- faza12 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 12 w skali BBCH
- faza13 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 13 w skali BBCH
- faza14 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 14 w skali BBCH
- faza15 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 15 w skali BBCH
- faza16 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 16 w skali BBCH
- faza17 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 17 w skali BBCH
- faza18 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 18 w skali BBCH

- faza19 – oznaczająca prawdopodobieństwo wykrycia kukurydzy będącej w fazie 19 w skali BBCH
- processing time – oznaczająca długość czasu przetwarzania
- timestamp – oznaczająca wartość parametru timestamp.

Poniżej znajduje się schemat przepływu danych w systemie z uwzględnieniem zapisu danych do pliku w formacie .csv, gdzie:

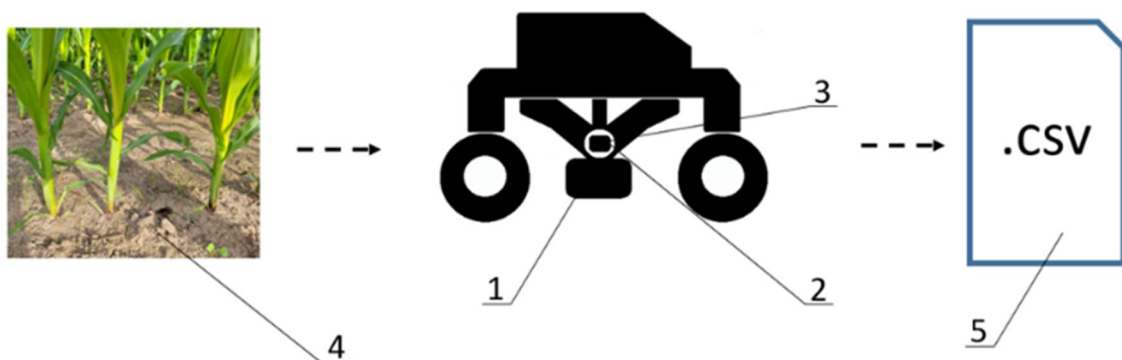
1 – kamera,

2 – komputer robota polowego,

3 – zestaw wytrenowanych modeli służących odpowiednio do detekcji i klasyfikacji faz rozwoju kukurydzy w skali BBCH, detekcji i klasyfikacji poziomów nawodnienia kukurydzy oraz detekcji i klasyfikacji kukurydzy zdrowej oraz kukurydzy porażonej wybranymi patogenami,

4 – obraz wejściowy,

5 – wyjściowy plik w formacie .csv.



**Rysunek 8.2.** Schemat układu z elementami wejścia i wyjścia.



## 9. Wnioski

### 9.1. Podsumowanie

Celem niniejszej rozprawy było zaproponowanie rozwiązań dla systemu robota polowego spełniającego wymogi rolnictwa precyzyjnego, wykorzystujących uczenie głębokie, analizę obrazów RGB oraz analizę obrazów wielospektralnych, które pozwolą na wsparcie procesu automatycznego określania kondycji roślin uprawnych, a w szczególności kukurydzy. Przeprowadzona w ramach rozprawy analiza dotyczyła zadań szczegółowych w postaci detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, detekcji i klasyfikacji poziomów nawodnienia kukurydzy oraz detekcji i klasyfikacji porażenia kukurydzy wybranymi patogenami.

Zrealizowane cele badawcze zostały nie tylko zweryfikowane w sposób teoretyczny, ale także przetestowane i wdrożone w rzeczywistym środowisku w postaci robota polowego, co czyni pracę szczególnie wartościową w kontekście jej praktycznej aplikacji. Wyniki przeprowadzonych prac znalazły zastosowanie w dwóch rzeczywistych projektach prowadzonych przez Sieć Badawczą Łukasiewicz – Instytut Lotnictwa, we współpracy z którym realizowany był doktorat wdrożeniowy, którego wynikiem jest niniejsza rozprawa. Mowa tutaj o projekcie Robot Polski NCBiR oraz o projekcie Fitoexport, w ramach których zaplanowano i przeprowadzono prace, udostępniono niezbędny sprzęt do pozyskiwania zobrażeń i prowadzenia obliczeń oraz podsumowano uzyskane wyniki w ramach dokumentacji projektowej, publikacji naukowych oraz zrealizowanego wdrożenia.

System opracowany w ramach niniejszej pracy, wspiera automatyzację oceny kondycji upraw działając na rzeczywistym prototypie robota polowego, obok innych modułów robota, takich jak: wykrywanie chwastów, pielenie, automatyczna nawigacja wzdłuż rzędów kukurydzy, nawożenie oraz stosowanie środków ochrony roślin (prace te zostały zrealizowane przez innych członków zespołu projektowego i nie są częścią niniejszej rozprawy). Efektywność i wszechstronność zastosowanych w ramach prac nad doktoratem modeli uczenia głębokiego potwierdzają, że sztuczna inteligencja odgrywa kluczową rolę w rozwoju zautomatyzowanych rozwiązań w dziedzinie rolnictwa precyzyjnego.

### 9.1.1. Główne osiągnięcia pracy i jej oryginalny wkład

**Oryginalnym dorobkiem naukowym autora, powstałym w ramach prac badawczych prowadzonych nad przygotowywaniem niniejszej rozprawy oraz wnoszącym wkład w rozwój dyscypliny naukowej, jest:**

- a) Stworzenie autorskich oznaczonych zbiorów danych RGB oraz wielospektralnych z kamery MicaSense RedEdge i kamery MapIR w celu wytrenowania na ich podstawie modeli służących do detekcji i klasyfikacji faz rozwoju kukurydzy w skali BBCH oraz poziomów nawodnienia kukurydzy - (rozdział 3.2.) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]
- b) Przeprowadzenie nowatorskich badań i wykorzystanie zobrazowań wielospektralnych z kamery MicaSense RedEdge i kamery MapIR do zupełnie nowego zastosowania, jakim jest detekcja i klasyfikacja faz rozwojowych kukurydzy oraz detekcja i klasyfikacja poziomów nawodnienia kukurydzy – (rozdziały 4, 5, 6) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]
- c) Opracowanie metody klasyfikatora głosującego, w którym modele wytrenowane na pojedynczych kanałach spektralnych zarejestrowanych za pomocą kamery MicaSense RedEdge głosują nad sumarycznym wynikiem klasyfikacji faz rozwojowych kukurydzy – (rozdziały 3.5.2., 4.4.) [Stypułkowska, Rokita, 2024]
- d) Przeprowadzenie badań nad detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH z bliskiej odległości, z poziomu robota polowego, co rozszerza dominujący sposób badania z poziomu BSP – (rozdziały 4, 5) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]
- e) Wykorzystanie zaawansowanych architektur głębokich sieci neuronowych do zupełnie nowego zastosowania, niż przykłady obecne w literaturze naukowej – (rozdziały 3.3., 3.4., 3.5., 4, 5, 6, 7) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]
- f) Automatyzacja procesów jednoczesnego określania takich parametrów kondycji kukurydzy jak: fazy rozwojowe, poziom nawodnienia i porażenie patogenami (rozdziały 4, 5, 6, 7, 8) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]
- g) Wdrożenie opracowanego rozwiązania w rzeczywistym projekcie robota polowego – (rozdziały 4, 5, 6, 7, 8) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]

- h) Wpływ opracowanego rozwiązania na automatyzację procesów charakterystycznych dla innej dziedziny naukowej, jaką jest rolnictwo (rozdziały 4, 5, 6, 7) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]

**Ad. a)** Stworzenie autorskich datasetów w postaci:

- oznaczonego zbioru obrazów pochodzących z kamery MicaSense RedEdge, odpowiadających poszczególnym 10 kanałom spektralnym, z obiektami w postaci kukurydzy oznaczonymi metodą polygon i mającymi przypisaną klasę odpowiadającą poszczególnym fazom rozwoju w skali BBCH;
- oznaczonego zbioru obrazów pochodzących z kamery RGB, z obiektami w postaci kukurydzy oznaczonymi metodą polygon i mającymi przypisaną klasę odpowiadającą poszczególnym fazom rozwoju w skali BBCH (obiekty w tym zbiorze są identyczne w obiektami powyższego zbioru pochodzącego z kamery RedEdge);
- oznaczonego zbioru obrazów pochodzących z kamery wielospektralnej MapIR, z obiektami w postaci kukurydzy oznaczonymi metodą polygon i mającymi przypisaną klasę odpowiadającą poszczególnym fazom rozwoju w skali BBCH;
- oznaczonego zbioru obrazów pochodzących z kamery RGB, z obiektami w postaci kukurydzy oznaczonymi metodą polygon i mającymi przypisaną klasę odpowiadającą poszczególnym fazom rozwoju w skali BBCH (obiekty w tym zbiorze są identyczne w obiektami powyższego zbioru pochodzącego z kamery MapIR);
- oznaczonego zbioru obrazów pochodzących z kamery wielospektralnej MapIR, oznaczonymi metodą ground truth i odpowiadających przyjętym zakresom nawodnienia kukurydzy;
- oznaczonego zbioru obrazów pochodzących z kamery RGB, oznaczonymi metodą ground truth i odpowiadających przyjętym zakresom nawodnienia kukurydzy (obiekty w tym zbiorze są identyczne w obiektami powyższego zbioru pochodzącego z kamery MapIR).

Jak zauważono na podstawie analizowanej literatury, żaden analogiczny zbiór danych nie został jeszcze dotychczas opublikowany. Są to zatem zupełnie nowe datasety, które otwierają pole do prowadzenia analogicznych badań i niewątpliwie stanowią oryginalny

dorobek naukowy. Stworzone zbiory powstały na poletku testowym uprawionym na potrzeby badań przez autora rozprawy, są zatem obiektem unikatowym.

**Ad. b)** Wykorzystanie zobrazowania wielospektralnego z kamery RedEdge oraz zobrazowania wielospektralnego z kamery MapIR do zupełnie nowego zastosowania, jakim jest detekcja i klasyfikacja faz rozwojowych kukurydzy oraz detekcja i klasyfikacja poziomów nawodnienia kukurydzy. Analiza literaturowa pozwoliła wysnuć wniosek, iż do tej pory jeszcze nikt nie prowadził identycznych badań, co podkreśla innowacyjność przeprowadzonych prac.

**Ad. c)** Opracowanie metody klasyfikatora głosującego, w którym modele wytrenowane na pojedynczych kanałach spektralnych głosują nad sumarycznym wynikiem klasyfikacji faz rozwojowych kukurydzy. W literaturze naukowej dostępne są liczne przykłady, w których modele wytrenowane przy pomocy różnych algorytmów głosują nad sumarycznym wynikiem, zaś w tym przypadku nowością jest głosowanie modeli opartych o ten sam algorytm użyty do danych odnoszących się do tych samych obiektów, lecz zobrazowanych przy użyciu różnych kanałów spektralnych.

**Ad. d)** Prowadzenie badań nad detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH z bliskiej odległości, z poziomu robota polowego. W literaturze dostępne są przykłady określania faz rozwojowych kukurydzy lub zliczania jej liści, jednak są to głównie badania prowadzone z poziomu BSP. Pozyskiwanie danych na temat rozwoju roślin z bliskiej odległości pozwala na uzyskanie nowej perspektywy i nowego sposobu pozyskiwania danych, co niewątpliwie poszerza dorobek naukowy dziedziny w tej kwestii.

**Ad. e)** Wykorzystanie zaawansowanych architektur głębokich sieci neuronowych do zupełnie nowego zastosowania, niż przykłady obecne w literaturze naukowej. Analiza literaturowa pozwoliła na wyciągnięcie wniosku, iż zadania realizowane w ramach niniejszej rozprawy nie były dotychczas przeprowadzone z wykorzystaniem architektur, które zastosowano podczas prac przedstawionych w niniejszej rozprawie. Pozwala to na pokazanie i porównanie ze sobą możliwości oferowanych przez poszczególne architektury, czyli w ramach zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy są to architektury: HTC\_r50, HTC\_r101, HTC\_x101, Mask2Former, zaś w przypadku zadania związanego z detekcją i klasyfikacją poziomów nawodnienia oraz detekcją i klasyfikacją porażenia kukurydzy wybranymi patogenami są to architektury: ResNet50, ConvNeXt small, EfficientNet\_B3, Vision Transformer oraz Swin Transformer.

**Ad. f)** Automatyzacja procesów jednoczesnego określania takich parametrów kondycji kukurydzy jak: fazy rozwojowe, poziom nawodnienia i porażenie patogenami - w przypadku obrazów RGB oraz w przypadku takich parametrów jak: fazy rozwojowe i poziomy nawodnienia - w przypadku obrazów wielospektralnych z kamery MapIR, a także jednoczesny export wyników w postaci dogodnej do dalszego przetwarzania w formacie .csv.

**Ad. g)** Wdrożenie rozwiązania w rzeczywistym projekcie robota polowego, co pokazało jego praktyczne zastosowanie w rzeczywistych warunkach uprawy kukurydzy.

**Ad. h)** Wpływ opracowanego rozwiązania na automatyzację procesów charakterystycznych dla innej dziedziny naukowej, jaką jest rolnictwo. Opracowane rozwiązanie przyczynia się do oszczędności czasu, zasobów oraz optymalizację monitorowania upraw, w porównaniu do ręcznego określania badanych parametrów z analogiczną dokładnością i częstością pomiarów. System pomaga również na szybkie reagowanie na zmieniające się warunki uprawy.

### 9.1.2. Wnioski z przeprowadzonych badań

**Poniżej przedstawiono podsumowanie wyników i wniosków uzyskanych w ramach prac eksperymentalnych opisanych w niniejszej rozprawie:**

- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB pozyskiwanych w sezonie wegetacyjnym 2022, w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, najlepsze wyniki według metryki F1-score uzyskał algorytm HTC\_x101, tuż za nim był algorytm HTC\_r101, następnie HTC\_r50, a najgorsze wyniki spośród badanych algorytmów osiągnął Mask2Former.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów wielospektralnych z kamery RedEdge pozyskiwanych w sezonie wegetacyjnym 2022, w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, najlepsze wyniki według metryki F1-score uzyskały następujące architektury: dla kanału spektralnego nr 01 – HTC\_r50, dla kanału spektralnego nr 02 – HTC\_x101, dla kanału spektralnego nr 03 – HTC\_r50, dla kanału spektralnego nr 04 – HTC\_r101, dla kanału spektralnego nr 05 – HTC\_r50, dla kanału spektralnego nr 06 – HTC\_x101, dla kanału spektralnego nr 07 – HTC\_r101, dla kanału spektralnego nr 08 – HTC\_x101, dla kanału spektralnego nr 09 – HTC\_r50, dla kanału spektralnego nr 10 – HTC\_r50.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów wielospektralnych z kamery RedEdge pozyskiwanych w sezonie wegetacyjnym 2022,

w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH oraz przy wykorzystaniu klasyfikatora głosującego najlepsze wyniki uzyskał algorytm HTC\_r50, tuż za nim był algorytm HTC\_x101, następnie HTC\_r101, a najgorsze wyniki uzyskał algorytm Mask2Former.

- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB oraz wyniki tychże modeli trenowanych na podstawie obrazów z kamery RedEdge pozyskiwanych w sezonie wegetacyjnym 2022 i zastosowaniu klasyfikatora głosującego, w celu realizacji zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH algorytm HTC\_r101 uzyskał lepszą skuteczność dla obrazów RGB, algorytm HTC\_r50 uzyskał minimalnie lepszą skuteczność dla danych wielospektralnych przy zastosowaniu klasyfikatora głosującego, algorytm HTC\_x101 uzyskał lepszą skuteczność dla obrazów RGB, zaś algorytm Mas2Former uzyskał zdecydowanie lepszą skuteczność dla danych wielospektralnych i przy zastosowaniu klasyfikatora głosującego.
- Porównując globalne wyniki dla zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH, na podstawie obrazów RGB i wielospektralnych z kamery RedEdge pozyskiwanych w sezonie wegetacyjnym 2022, najlepsze wyniki osiągnięto przy zastosowaniu obrazowania RGB oraz architektury HTC\_x101. To właśnie rozwiązanie zostało wdrożone w rzeczywistym projekcie.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB pozyskiwanych w sezonie wegetacyjnym 2024, w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, najlepsze wyniki uzyskał algorytm HTC\_r50, tuż za nim był algorytm HTC\_x101, następnie HTC\_r101, a najgorsze wyniki spośród badanych algorytmów osiągnął Mask2Former.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów wielospektralnych z kamery MapIR, pozyskiwanych w sezonie wegetacyjnym 2024, w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, najlepsze wyniki uzyskał algorytm HTC\_r50, tuż za nim znalazł się algorytm HTC\_r101, następnie HTC\_x101, zaś najgorsze wyniki uzyskał algorytm Mask2Former.
- Porównując wyniki uzyskane z wykorzystaniem modeli trenowanych na podstawie obrazów RGB i wielospektralnych z kamery MapIR pozyskiwanych w sezonie wegetacyjnym 2024, w celu realizacji zadania związanego z detekcją i klasyfikacją faz



rozwijających kukurydzy w skali BBCH, najlepsze wyniki uzyskano przy zastosowaniu obrazowania RGB oraz architektury HTC\_r50.

- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB pozyskiwanych w sezonie wegetacyjnym 2024, w celu detekcji i klasyfikacji poziomów nawodnienia kukurydzy, najlepsze wyniki uzyskał algorytm ConvNeXt Small, tuż za nim był algorytm Vision Transformer, następnie Swin Transformer, EfficientNet\_B3, a najgorsze wyniki spośród badanych algorytmów osiągnął ResNet50.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów wielospktralnych z kamery MapIR pozyskiwanych w sezonie wegetacyjnym 2024, w celu detekcji i klasyfikacji poziomów nawodnienia kukurydzy, najlepsze wyniki uzyskał algorytm Swin Transformer, tuż za nim był algorytm ConvNeXt Small, następnie Vision Transformer, EfficientNet\_B3, a najgorsze wyniki spośród badanych algorytmów osiągnął ResNet50.
- Dla zadania mającego na celu detekcję i klasyfikację poziomów nawodnienia kukurydzy pozyskiwanych w sezonie wegetacyjnym 2024, najlepsze wyniki uzyskano dzięki zastosowaniu obrazowania RGB oraz architektury ConvNeXt Small. To właśnie rozwiązanie zostało wdrożone w rzeczywistym projekcie.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB, w celu detekcji i klasyfikacji porażenia kukurydzy, najlepsze wyniki uzyskał algorytm Swin Transformer, tuż za nim był algorytm Vision Transformer, następnie ConvNeXt Small, EfficientNet\_B3, a najgorsze wyniki spośród badanych algorytmów osiągnął ResNet50, który zdecydowanie odstawał od reszty algorytmów.
- Architektury sieci neuronowych typu: ConvNeXt small, Vision Transformer, Swin Transformer oraz HTC sprawdziły się w klasyfikacji obrazów roślin, zapewniając wysoką dokładność uzyskiwanych wyników.
- Dzięki integracji opracowanego systemu z innymi modułami robota, możliwe było stworzenie kompleksowego rozwiązania, które automatyzuje procesy monitorowania parametrów kondycji roślin z procesami pielęgnacji, nawożenia i ochrony roślin. To pokazuje, że systemy AI mogą skutecznie współpracować z innymi zaawansowanymi narzędziami technologicznymi, usprawniając systemy zarządzania w produkcji rolnej zgodnie z ideą rolnictwa precyzyjnego.

## 9.2. Rekomendacje dla przyszłych badań

Opracowane rozwiązanie wykorzystuje zbiory oznaczonych danych RGB pochodzących z sezonu wegetacyjnego 2022 i 2024, zbiór danych wielospektralnych z kamery MicaSense RedEdge MX Dual pochodzących z sezonu wegetacyjnego 2022 oraz zbiór danych wielospektralnych z kamery MapIR Survey3 Red + Green + NIR pochodzące z sezonu wegetacyjnego 2024. Niewątpliwym kierunkiem rozwoju w tym przypadku jest rozszerzenie tychże zbiorów o dodatkowe obrazy i oznaczone według przyjętych zasad omówionych w niniejszej rozprawie. Zwiększenie liczności zbiorów treningowych pozwoli na uzyskanie jeszcze wyższej skuteczności omawianych modeli uczenia głębokiego, gdyż wpłynie na zwiększenie różnorodności tychże zbiorów oraz polepszy zdolność modeli do uogólniania wyników na nowe przypadki. Mowa tutaj zarówno o zadaniu związanym z detekcją i klasyfikacją faz rozwojowych kukurydzy, jak i detekcją i klasyfikacją jej poziomów nawodnienia oraz porażenia wybranymi patogenami.

Jeśli mowa już o zbiorach danych, to równie cennym usprawnieniem będzie efektywne porażenie roślin na poletku testowym wybranymi patogenami, aby stworzyć dzięki temu własny zbiór danych i móc wykorzystać w nim nie tylko obrazowanie RGB, jak to było do tej pory, ale również obrazowanie wielospektralne, które może pokazać więcej różnic między okazami roślin zdrowych i porażonych, niż to jest obecne w przypadku obrazowania RGB.

Obecne rozwiązanie opiera się na wykorzystaniu w ramach zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy w skali BBCH, następujących algorytmów: HRC\_r50, HTC\_r10, HTC\_x101, Mask2Former, a w przypadku zadania związanego z detekcją i klasyfikacją poziomów nawodnienia kukurydzy oraz porażenia kukurydzy wybranymi patogenami, następujących algorytmów: ResNet50, ConvNeXt small, EfficientNet\_B3, Vision Transformer i Swin Transformer. Niewątpliwym kierunkiem rozwoju może być zarówno dalsza optymalizacja obecnych algorytmów, jak i użycie zupełnie nowych architektur i porównanie ich wyników z wynikami dotychczasowych algorytmów. Może być to ścieżką do uzyskania wyższej wydajności i skuteczności wyników opracowanego rozwiązania.

Kolejnym kierunkiem rozwoju może być rozszerzenie obecnego rozwiązania na inne uprawy. Obecne rozwiązanie skupia się wyłącznie na analizie obrazów kukurydzy, lecz nie stoi na przeszkodzie, aby założyć nowe poletka badawcze i hodować na nich inne gatunki roślin uprawnych, powtarzając jednocześnie dotychczas opracowany cykl badań wraz ze zbieraniem zobrażeń, ich oznaczaniem, trenowanie głębokich sieci neuronowych i analizą wyników

uzyskanych dzięki wytrenowanym modelom. W literaturze naukowej można odnaleźć uszczegółowienie skali BBCH również dla innych upraw zbóż, co wypełnia podstawowy warunek profesjonalnego określania faz rozwojowych tychże roślin. W ten sposób analiza stanu zdrowia i stopnia rozwoju roślin uprawnych może być rozszerzona o nowe gatunki kluczowych roślin uprawnych, jakimi są np. żyto, pszenica, owies, proso itp.

Opracowane rozwiązanie było przygotowywane na bazie obrazów kukurydzy pozyskiwanych na terenie Polski. Jako dalszy kierunek rozwoju warto zbadać, jak system radzi sobie w różnych warunkach klimatycznych i glebowych, aby zwiększyć jego adaptacyjność i wszechstronność w innych regionach świata.

Opracowane rozwiązanie mogłoby dodatkowo łączyć się z technologiami IoT, zapewniając jeszcze lepszy monitoring w czasie rzeczywistym i dając możliwość zbierania i analizy danych z wielu pól jednocześnie. Systemy te mogłyby także komunikować się bezpośrednio z urządzeniami na polu, jeszcze bardziej automatyzując proces zarządzania uprawami.

Nowe technologie w rolnictwie, takie jak sztuczna inteligencja, mają potencjał rewolucjonizować uprawy i wspierać rolników w ich codziennych działaniach, co czyni dalsze badania w tej dziedzinie niezwykle wartościowymi.

## Spis rysunków

**Rysunek 2.1.** Całkowite pobudzenie zastosowane w uczenia maszynowym [Rashka, 2018].

**Rysunek 2.2.** Rozdzielczość liniowa klas [Rashka, 2018].

**Rysunek 2.3.** Ogólny model perceptronu [Rashka, 2018].

**Rysunek 2.4.** Schemat modelu Adaline [Rashka, 2018].

**Rysunek 2.5.** Poglądowy schemat działania algorytmu gradientu prostego [Rashka, 2018].

**Rysunek 2.6.** Model wielowarstwowej sieci neuronowej [Rashka, 2018].

**Rysunek 2.7.** Dla danych graficznych można zdefiniować każdą uczoną cechę jako funkcję od małego obszaru z danych [Weidman, 2019].

**Rysunek 2.8.** Skala BBCH w ujęciu graficznym (źródło: [https://pdf.helion.pl/e\\_1wwu/e\\_1wwu.pdf](https://pdf.helion.pl/e_1wwu/e_1wwu.pdf)) [Helion, 2024].

**Rysunek 2.9.** Przykład obrazu kukurydzy będącej w 16 fazie rozwoju w skali BBCH.

**Rysunek 3.1.** Widok szklarni z zewnątrz i wewnątrz, w której prowadzona była uprawa kukurydzy w sezonie wegetacyjnym 2021, miejscowość Borowiec k. Warszawy.

**Rysunek 3.2.** Widok poletka testowego, na którym prowadzona była uprawa kukurydzy w sezonie wegetacyjnym 2022, miejscowość Borowiec k. Warszawy.

**Rysunek 3.3.** Widok poletka testowego, na którym prowadzona była uprawa kukurydzy w sezonie wegetacyjnym 2022 i 2024, miejscowość Borowiec k. Warszawy.

**Rysunek 3.4.** Obrazy kukurydzy wykonane w szklarni testowej.

**Rysunek 3.5.** Obrazy kukurydzy będącej w jej początkowych fazach rozwojowych.

**Rysunek 3.6.** Obrazy kukurydzy niezachwaszczonej i zachwaszczonej będącej w jej wyższych fazach rozwojowych.

**Rysunek 3.7.** Obrazy kukurydzy wykonane wieczorem.

**Rysunek 3.8.** Obrazy kukurydzy charakteryzującej się niskim (po lewej) oraz wysokim poziomem nawodnienia (po prawej).

**Rysunek 3.9.** Przykładowe obrazy pochodzące z analizowanego datasetu - gąsiennica *Spodoptera frugiperda*

**Rysunek 3.10.** Przykładowe obrazy pochodzące z analizowanego datasetu - Helmintosporioza - grzyby *Setosphaeria turcica*

**Rysunek 3.11.** Przykładowe obrazy pochodzące z analizowanego datasetu - rdza pospolita - grzyby z rzędu rdzowców (Pucciniales)

**Rysunek 3.12.** Przykładowe obrazy pochodzące z analizowanego datasetu - szara plamistość liści - grzyby *Aureobasidium zeae*

**Rysunek 3.13.** Przykładowe obrazy pochodzące z analizowanego datasetu - zdrowa kukurydza

**Rysunek 3.14.** Parametry kamery MicaSense RedEdge MX DUAL.

**Rysunek 3.15.** Wygląd kamery MicaSense RedEdge-MX DUAL.

**Rysunek 3.16.** Przykładowe obrazy kukurydzy zarejestrowane przez poszczególne obiektywy kamery wielospektralnej MicaSense RedEdge-MX DU-AL oraz przykładowy obraz tych samych roślin zarejestrowany przez kamerę RGB.

**Rysunek 3.17.** Wygląd kamery MapIR.

**Rysunek 3.18.** Obrazy pozyskane za pomocą kamery MapIR.

**Rysunek 3.19.** Widok środowiska Label Studio podczas procesu oznaczania. Na ekranie widać rośliny będące w fazie 14 oznaczone metodą wielokąta (ang. „polygon”).

**Rysunek 3.20.** Widok środowiska Label Studio podczas procesu oznaczania. Na ekranie widać rośliny będące w fazie 17 i 18 oznaczone metodą wielokąta (ang. „polygon”).

**Rysunek 3.21.** Widok środowiska Label Studio podczas procesu oznaczania. Na ekranie widać rośliny będące w fazie 14 oznaczone metodą wielokąta (ang. „polygon”).

**Rysunek 3.22.** Widok środowiska Label Studio podczas procesu oznaczania. Na ekranie widać rośliny będące w fazie 19 oznaczone metodą wielokąta (ang. „polygon”).

**Rysunek 3.23.** Widok panelu z wyborem formatu wyjściowego w środowisku Label Studio.

**Rysunek 3.24.** Wykrywanie obiektów – state of the art.

**Rysunek 4.1.** Schemat zaproponowanego systemu klasyfikacji faz rozwojowych kukurydzy w skali BBCH.

**Rysunek 4.2.** Wynik predykcji modelu opartego o uczenie na bazie obrazów wielospktralnych.

**Rysunek 4.3.** Wynik predykcji modelu – pierwsze eksperymenty.

**Rysunek 4.4.** Wynik predykcji modelu.

**Rysunek 4.5.** Wynik predykcji modelu wykrywania kukurydzy i klasyfikacji jej faz rozwojowych.

**Rysunek 4.6.** Wynik predykcji modelu wykrywania kukurydzy i klasyfikacji jej faz rozwojowych.

**Rysunek 4.7.** Wynik predykcji modelu wykrywania kukurydzy i klasyfikacji jej faz rozwojowych.

**Rysunek 4.8.** Wizualizacja porównująca wyniki klasyfikacji poszczególnych algorytmów na obrazach RGB z wynikami uzyskanymi przez te algorytmy z użyciem obrazów wielospktralnych i zastosowaniem klasyfikatora głosującego.

**Rysunek 4.9.** Porównanie wyników modelu HTC\_r101 trenowanego na poszczególnych kanałach spektralnych.

**Rysunek 4.10.** Porównanie wyników modelu HTC\_x101 trenowanego na poszczególnych kanałach spektralnych.

**Rysunek 4.11.** Porównanie wyników modelu HTC\_r50 trenowanego na poszczególnych kanałach spektralnych.

**Rysunek 4.12.** Porównanie wyników modelu Mask2Former trenowanego na poszczególnych kanałach spektralnych.

**Rysunek 4.13.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych RGB.

**Rysunek 4.14.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z pierwszego kanału spektralnego.

**Rysunek 4.15.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z drugiego kanału spektralnego.

**Rysunek 4.16.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z trzeciego kanału spektralnego.

**Rysunek 4.17.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z czwartego kanału spektralnego.

**Rysunek 4.18.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z piątego kanału spektralnego.

**Rysunek 4.19.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z szóstego kanału spektralnego.

**Rysunek 4.20.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z siódmego kanału spektralnego.

**Rysunek 4.21.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z ósmego kanału spektralnego.

**Rysunek 4.22.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z dziewiątego kanału spektralnego.

**Rysunek 4.23.** Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z dziesiątego kanału spektralnego.

**Rysunek 5.1.** Schemat zaproponowanego systemu klasyfikacji faz rozwojowych kukurydzy w skali BBCH z użyciem zobrazowania RGB oraz zobrazowania z kamery MapIR.

**Rysunek 5.2.** Wykres przedstawiający wartości parametru F1-score dla poszczególnych zobrazowań oraz analizowanych typów zobrazowania: RGB oraz zobrazowania wielospektralnego z kamery MapIR.

**Rysunek 5.3.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach RGB.

**Rysunek 5.4.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach RGB.

**Rysunek 5.5.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach RGB.

**Rysunek 5.6.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR.



**Rysunek 5.7.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR.

**Rysunek 5.8.** Wynik działania modelu HTC\_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR.

**Rysunek 5.9.** Przebieg uczenia modelu opartego o algorytm HTC\_r50 dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.

**Rysunek 5.10.** Przebieg uczenia modelu opartego o algorytm HTC\_r101 dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.

**Rysunek 5.11.** Przebieg uczenia modelu opartego o algorytm HTC\_x101 dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.

**Rysunek 5.12.** Przebieg uczenia modelu opartego o algorytm Mask2Former dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR.

**Rysunek 5.13.** Przebieg uczenia modeli opartych o algorytmy HTC\_r50, HTC\_r101, HTC\_x101 oraz Mask2Former dla zobrazowania RGB.

**Rysunek 5.14.** Przebieg uczenia modeli opartych o algorytmy HTC\_r50, HTC\_r101, HTC\_x101 oraz Mask2Former dla zobrazowania wielospektralnego z kamery MapIR.

**Rysunek 6.1.** Schemat zaproponowanego rozwiązania w ramach realizacji zadania detekcji i klasyfikacji poziomów nawodnienia kukurydzy.

**Rysunek 6.2.** Schemat przepływu danych w rozwiązaniu opracowanym dla problemu określania poziomów nawodnienia kukurydzy.

**Rysunek 6.3.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas treningu na danych z kamery MapIR.

**Rysunek 6.4.** Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu na danych z kamery MapIR.

**Rysunek 6.5.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas procesu walidacji na danych z kamery MapIR.

**Rysunek 6.6.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich średniej precyzji podczas treningu na danych z kamery MapIR.

**Rysunek 6.7.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas treningu na danych z kamery RGB.

**Rysunek 6.8.** Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu na danych z kamery RGB.

**Rysunek 6.9.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas procesu walidacji na danych z kamery RGB.

**Rysunek 6.10.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich średniej precyzji podczas treningu na danych z kamery RGB.

**Rysunek 7.1.** Schemat zaproponowanego rozwiązania w ramach realizacji zadania detekcji i klasyfikacji kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami.

**Rysunek 7.2.** Schemat przepływu danych w rozwiązaniu opracowanym dla problemu określania porażenia kukurydzy wybranymi patogenami.

**Rysunek 7.3.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas treningu,

**Rysunek 7.4.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich średniej precyzji podczas treningu.

**Rysunek 7.5.** Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu.

**Rysunek 7.6.** Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas procesu walidacji.

**Rysunek 7.7.** Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas walidacji.

**Rysunek 8.1.** Wygląd całego interfejsu stworzonej aplikacji - przypadek analizy obrazu RGB.

**Rysunek 8.2.** Schemat układu z elementami wejścia i wyjścia.

## Spis tabel

**Tabela 4.1.** Wyniki klasyfikacji dla każdego z analizowanych kanałów spektralnych z użyciem poszczególnych algorytmów uczenia głębokiego.

**Tabela 4.2.** Wyniki klasyfikacji dla wybranych kanałów spektralnych z wykluczeniem kanału nr 10.

**Tabela 4.3.** Wyniki klasyfikacji dla każdego badanego algorytmu.

**Tabela 4.4.** Porównanie wyników klasyfikacji poszczególnych algorytmów na obrazach RGB z wynikami uzyskanymi przez te algorytmy przy użyciu obrazów wielospektralnych oraz zastosowaniu opracowanej metody z użyciem klasyfikatora głosującego. Do analizy wykorzystano metryki: Accuracy, Precision, Recall oraz F1-score.

**Tabela 5.1.** Wyniki klasyfikacji dla poszczególnych algorytmów oraz analizowanych typów zobrazenia RGB oraz wielospektralnego z kamery MapIR.

**Tabela 6.1.** Porównanie wyników uzyskanych dla danych wielospektralnych z kamery MapIR przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer.

**Tabela 6.2.** Porównanie wyników uzyskanych dla danych RGB przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_b3, Vision Transformer i Swin Transformer.

**Tabela 7.1.** Porównanie wyników uzyskanych dla danych RGB przy zastosowaniu algorytmów ConvNeXt small, ResNet50, EfficientNet\_B3, Vision Transformer i Swin Transformer.

## Bibliografia

[Abadi i in., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16) (pp. 265-283).

[Anaconda, 2020] Anaconda. (2020). Anaconda Software Distribution. <https://anaconda.com>

[Albon, 2019] Albon C., *Uczenie maszynowe w Pythonie. Receptury*, Helion, Gliwice, 2019.

[Berger i in., 2010] Berger, B., Parent, B., & Tester, M. (2010). High-throughput shoot imaging to study drought responses. *Journal of Experimental Botany*, 61(12), 3519-3528.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Bock et al., 2010] Bock, C. H., Poole, G. H., Parker, P. E., & Gottwald, T. R. (2010). Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging. *Critical Reviews in Plant Sciences*, 29(2), 59-107.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123-140.

[Brown, 2015] Brown, C. (2015). Challenges in Color Image Processing: Lighting Conditions and Noise. *IEEE Transactions on Image Processing*, 24(4), 1342-1353.

[Bydin, Pearmutter, 2014] A.G. Baydin, B.A. Pearmutter, Automatic Differentiation of Algorithms for Machine Learning, arXiv preprint arXiv:1404.7456, 2014 <https://arxiv.org/pdf/1404.7456.pdf>

[Chen i in., 2018] Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801-818).

[Chen i in., 2019] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., ... & Sun, J. (2019). Hybrid Task Cascade for Instance Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4974-4983).

[Chen i in., 2021] Chen, L. C., Lu, W. C., Yu, Y., Wang, X., & Sun, J. (2021). Mask2Former: Multi-scale masked transformer for image recognition. arXiv preprint arXiv:2105.06214.

**[Chollet, 2019]** Chollet F., *Deep Learning. Praca z językiem Python i biblioteką Keras*, Helion, Gliwice, 2019.

**[Clark, 2013]** Clark, D. (2013). Precision Agriculture and the Use of Multispectral Data for Plant Health Monitoring. *Agricultural Sciences*, 18(4), 445-462.

**[Dalal, Triggs, 2005]** Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

**[Deng, 2018]** Deng, L. (2018). Cloud Computing for Deep Learning: Opportunities and Concerns. *IEEE Signal Processing Magazine*.

**[Dosovitskiy i in., 2020]** Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

**[Geiger, 2012]** Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3354-3361.

**[Ge i in., 2021]** Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOX: Exceeding YOLO Series in 2021. *arXiv preprint arXiv:2107.08430*.

**[Gonzalez, Woods, 2008]** Gonzalez, R., & Woods, R. (2008). *Digital Image Processing*. Pearson.

**[Goodfellow i in., 2016]** Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

**[Hastie, Tibshirani, Friedman, 2009]** Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

**[He i in., 2016]** He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

**[He i in., 2017]** He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2961-2969.

**[Helion, 2024]** [https://pdf.helion.pl/e\\_1www/e\\_1www.pdf](https://pdf.helion.pl/e_1www/e_1www.pdf), dostęp wrzesień 2024

**[Huang i in., 2019]** Huang, Y., Liao, B., Cheng, S., & Wu, H. (2019). GPipe: Efficient training of giant neur.

**[Jain, 2001]** Jain, A. (2001). Fundamentals of Digital Image Processing. Prentice Hall.

**[Jones, 2004]** Jones, H. G. (2004). Irrigation scheduling: advantages and pitfalls of plant-based methods. *Journal of Experimental Botany*, 55(407), 2427-2436.

**[Jones, 2016]** Jones, B. (2016). Multispectral Imaging in Environmental Monitoring. *Journal of Remote Sensing*, 32(7), 1225-1240.

**[Jones, 2018]** Jones, B. (2018). Digital Image Representation: Fundamentals of RGB and Beyond. Cambridge: Cambridge University Press.

**[Jouppi i in., 2017]** Jouppi, N. P., Young, C., Patil, N., et al. (2017). In-Datcenter Performance Analysis of a Tensor Processing Unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*.

**[Kamilaris, Prenafeta-Boldú, 2018]** Kamilaris, A., Prenafeta-Boldú, F. X. (2018). Deep Learning in Agriculture: A Survey. *Computers and Electronics in Agriculture*.

**[Khan, 2021]** Khan, A., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2021). Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*.

**[Kingma & Ba, 2015]** Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

**[Krizhevsky i in., 2012]** Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097-1105.

**[Krzanowski, 2000]** W.J. Krzanowski, *Principles of Multivariate Analysis: A User's Perspective*, Oxford University Press, 2000

**[Lancashire i in., 1991]** Lancashire, P.D., Bleiholder, H., Van Den Boom, T., Langelüddeke, P., Stauss, R., Weber, E., & Witzemberger, A. (1991). A uniform decimal code for growth stages of crops and weeds. *Annals of Applied Biology*, 119(3), 561-601.

**[LeCun i in., 1998]** LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

**[LeCun i in., 2015]** LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

**[Lin i in., 2017]** Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125).

**[Liu i in., 2019]** Liu, M., et al. (2019). Maize Seedling Detection Using UAV RGB Imagery and YOLO Algorithm. *Remote Sensing*.

**[Liu, M., i in., 2019]** Liu, M., et al. (2019). Maize Seedling Detection Using UAV RGB Imagery and YOLO Model. *Precision Agriculture*, 20(6), 1099-1114.

**[Li i in., 2020]** Li, W., et al. (2020). Multispectral Imaging and Artificial Intelligence for Agricultural Applications. *Remote Sensing of Environment*.

**[Liu i in., 2021]** Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 10012-10022).

**[Liu i in., 2022]** Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11976-11986).

**[Litjens, 2017]** Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A., Ciompi, F., Ghafoorian, M., ... & van Ginneken, B. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42, 60-88.

**[Mahlein i in., 2016]** Mahlein, A. K. (2016). Hyperspectral Imaging for Detection of Plant Diseases. *Annual Review of Phytopathology*.

**[Mahlein, 2016]** Mahlein, A. K. (2016). Plant disease detection by imaging sensors – parallels and specific demands for precision agriculture and plant phenotyping. *Plant Disease*, 100(2), 241-251.

**[Manning i in., 2008]** Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.



[Meier, 2001] Meier, U. (2001). Growth Stages of Mono and Dicotyledonous Plants. BBCH Monograph. Federal Biological Research Centre for Agriculture and Forestry, Bonn.

[MMDetection, 2020] Contributors to MMDetection. (2020). MMDetection: OpenMMLab Detection Toolbox and Benchmark. <https://github.com/open-mmlab/mmdetection>

[Nickolls i in., 2008] Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with CUDA. ACM Queue, 6(2), 40-53.

[NVIDIA, 2020] NVIDIA. (2020). NVIDIA Ampere Architecture. <https://www.nvidia.com/en-us/technologies/ampere-architecture/>

[Oreilly, 2024] <https://oreil.ly/2KOfzfs>, dostęp wrzesień 2024

[Osinga, 2019] Osinga D., *Deep Learning. Receptury*, Helion, Gliwice, 2019.

[paperswithcode.com, 2024] [COCO test-dev Benchmark \(Instance Segmentation\) | Papers With Code](#)

[Parker, 2010] Parker, J. (2010). Algorithms for Image Processing and Computer Vision. Wiley.

[Paszke i in., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in neural information processing systems (pp. 8024-8035).

[Powers, 2011] Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies, 2(1), 37–63.

[Rashka, 2018] S. Rashka, *Python. Uczenie maszynowe*, Helion, Gliwice, 2018.

[Redmon, Farhadi, 2018] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

[Rosenblatt, 1957] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automation*, „Cornel Aeronautical Laboratory”, 1957

**[Rosenblatt, 1958]** Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

**[Rumelhart i in., 1986]** Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

**[Sasaki, 2007]** Sasaki, Y. (2007). The Truth of the F-measure [Technical Report]. Tokyo: School of Computer Science, University of Manchester

**[Sharma i in., 2020]** Sharma, A., Kamilaris, A., Prenafeta-Boldú, F. X. (2020). Artificial Intelligence in Agriculture: A Review. *Agronomy*.

**[Shorten & Khoshgoftaar, 2019]** Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 1-48.

**[Simonyan, 2014]** Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

**[Smith, 2012]** Smith, A. (2012). *Image Processing and Analysis with RGB Data*. New York: Springer.

**[Sokolova & Lapalme, 2009]** Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4), 427-437.

**[Srivastava i in., 2014]** Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929-1958.

**[Stypułkowska, 2023a]** Stypułkowska Justyna, "The Use of AI to Determine the Condition of Corn in a Field Robot that Meets the Requirements of Precision Farming", W: *Communication Papers of the 18th Conference on Computer Science and Intelligence Systems / Ganzha Maria [i in.] (red.), Annals of Computer Science and Information Systems*, 2023, vol. 37, Warszawa, Polskie Towarzystwo Informatyczne, s.313-321, Numer artykułu:3649, ISBN 978-83-969601-4-6. DOI:10.15439/2023f3649

**[Stypułkowska, 2023b]** Zgłoszenie patentowe nr P.446025 „Sposób oraz urządzenie do kontroli parametrów jakościowych upraw, w szczególności upraw kukurydzy”, Urząd

Patentowy Rzeczypospolitej Polskiej, zgłaszający: Sieć Badawcza Łukasiewicz – Instytut Lotnictwa, twórca: Justyna Stypułkowska

**[Stypułkowska, Rokita, 2024]** Stypułkowska Justyna, Rokita Przemysław, „Detection and Classification of Maize Growth Stages in BBCH Scale Using Multispectral and RGB Images with Deep Neural Networks and Voting Classifier Approach”, Machine Graphics and Vision – artykuł zaakceptowany do publikacji

**[Taigman i in., 2014]** Y. Taigman, M. Yang, M. Ranzato, L. Wolf, *DeepFace: Closing the gap to human-level performance in face verification*, „Computer Vision and Pattern Recognition CVPR”, 2014 IEEE Conference, s. 1701 – 1708)

**[Tan, Le, 2019]** Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

**[Thompson, 2015]** Thompson, E. (2015). Applications of Ultraviolet and Infrared Imaging in Material Analysis. Journal of Optical Science, 27(3), 333-350.

**[Vaswani, 2017]** Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998-6008.

**[Vaswani i in., 2017]** Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*.

**[Wang i in., 2020]** Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops (pp. 390-391).

**[Wang i in., 2020]** Wang, C. Y., et al. (2020). CSPNet: A New Backbone that can Enhance Learning Capability of CNN. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 1571-1580.

**[Weidman, 2019]** S. Weidman, Deep Learning from Scratch: Building with Python from First Principles, 2019, tytuł Polski “Uczenie głębokie od zera. Podstawy Implementacji w Pythonie”, tłumaczenie: Tomasz Walczak, wyd. Helion, Gliwice, 2020

**[White, 2019]** White, D. (2019). Advances in Image Noise Reduction Techniques. *Journal of Visual Computing*, 9(2), 85-101.

**[Widrow i in., 1960]** B. Widrow i in., *An adaptive „Adaline” neuron using chemical „memistors”*, „Number Technical Report” 1553-2, Stanford Electron. Labs, Stanford, California, October 1960

**[Wightman, 2019]** Wightman, R. (2019). PyTorch Image Models. <https://github.com/rwightman/pytorch-image-models>

**[Wolfert i in., 2017]** Wolfert, S., Ge, L., Verdouw, C., Bogaardt, M.-J. (2017). Big Data in Smart Farming – A Review. *Agricultural Systems*.

**[Xie i in., 2017]** Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1492-1500).

**[Xie, 2017]** Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5987-5995.

**[Xu, X., i in., 2020]** Xu, X., et al. (2020). Corn Leaf Detection Using Deep Learning for Precision Agriculture. *Journal of Field Robotics*, 37(4), 585-597.

**[Xu i in., 2020]** Xu, X., et al. (2020). Maize Leaf Detection Based on RGB UAV and Deep Learning Algorithms. *Computers and Electronics in Agriculture*.

**[Yosinski i in., 2014]** Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. *Advances in Neural Information Processing Systems*, 27, 3320-3328.

**[Yu, D., i in., 2021]** Yu, D., et al. (2021). Estimating Above-Ground Biomass of Maize Using UAV-Based Multispectral and Thermal Imagery. *Remote Sensing*, 13(18), 3647.

**[Zan, X., i in., 2018]** Zan, X., et al. (2018). Tassel Detection in Maize Using UAV RGB Imagery and Machine Learning Techniques. *Computers and Electronics in Agriculture*, 148, 93-101.

**[Zhang & Ma, 2012]** Zhang, C., & Ma, Y. (2012). Ensemble machine learning: Methods and applications. Springer Science & Business Media.

**[Zhang i in., 2019]** Zhang, J., Pu, R., Huang, W., Yuan, L., & Luo, J. (2019). Detecting powdery mildew of winter wheat using leaf level hyperspectral measurements. *Computers and Electronics in Agriculture*, 157, 114-121.

**[Zhang, Sabuncu, 2018]** Zhang, Z., & Sabuncu, M. R. (2018). Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. *Advances in Neural Information Processing Systems*, 31, 8778-8788.

**[Zhao i in., 2020]** Zhao, J., et al. (2020). Deep Learning for Maize Growth Stage Classification Based on RGB Images. *Plant Methods*, 16(1), 99-112.

**[Zhou i in., 2020]** Zhou, X., et al. (2020). Multispectral Imaging in Agriculture: Advances and Applications. *Plant Methods*.

**[Zocca i in., 2018]** Zocca V., Spacagna G., Slater D., Roelants P., *Deep Learning. Uczenie głębokie z językiem Python. Sztuczna inteligencja i sieci neuronowe*, Helion, Gliwice, 2018.

## Załączniki

### A. Publikacje, nagroda, wniosek patentowy

**A1** – Artykuł naukowy J.Stypułkowska, *The Use of AI to Determine the Condition of Corn in a Field Robot that Meets the Requirements of Precision Farming*, 2023

**A2** – Artykuł naukowy J.Stypułkowska, P.Rokita, *Detection and Classification of Maize Growth Stages in BBCH Scale Using Multispectral and RGB Images with Deep Neural Networks and Voting Classifier Approach*, 2024

**A3** – Dyplom wyróżnienia uzyskany na konferencji naukowej *18th Conference on Computer Science and Intelligence Systems FedCSIS 2023* za wystąpienie oraz napisanie artykułu naukowego pt. *The Use of AI to Determine the Condition of Corn in a Field Robot that Meets the Requirements of Precision Farming*

**A4** – Potwierdzenie złożenia wniosku patentowego w Urzędzie Patentowym Rzeczypospolitej Polskiej pt. *Sposób oraz urządzenie do kontroli parametrów jakościowych upraw, w szczególności upraw kukurydzy*

**A5** – Artykuł popularnonaukowy z portalu Farming Future Food, w którym opisano wyniki moich badań, tytuł artykułu: *Artificial intelligence can provide accurate overview of corn development and health*

### B. Dokumentacja techniczna wykorzystanych kamer wielospektralnych

**B1** – dokumentacja techniczna kamery MicaSense RedEdge MX Dual

**B2** – dokumentacja techniczna kamery MapIR Survey3

## Streszczenie

Analiza obrazów wielospektralnych i RGB z zastosowaniem głębokich sieci neuronowych dla potrzeb automatycznej oceny parametrów kondycji kukurydzy przez robota polowego.

*Celem niniejszej rozprawy jest przeprowadzenie badań z wykorzystaniem głębokich sieci neuronowych w celu analizy obrazów wielospektralnych i RGB na potrzeby opracowania systemu do automatycznej detekcji i klasyfikacji faz rozwoju kukurydzy w skali BBCH, poziomów nawodnienia oraz porażenia kukurydzy patogenami. Zadania te są częścią rzeczywistych prac projektowych prowadzonych w ramach doktoratu wdrożeniowego realizowanego w Politechnice Warszawskiej oraz Sieci Badawczej Łukasiewicz – Instytucie Lotnictwa. Analiza literatury pokazuje niezliczone przykłady zastosowań sztucznej inteligencji, która jest wyjątkowo szybko rozwijającym się działem informatyki mającym wpływ na rozwój wielu innych dyscyplin naukowych. Niniejsza rozprawa skupia się na wykorzystaniu głębokich sieci neuronowych do analizy obrazów wspierając tym samym rozwój rolnictwa precyzyjnego. Podczas realizacji badań skupiono się na wypełnieniu zauważonych luk badawczych, a przeprowadzone eksperymenty oparte zostały o nowoczesne architektury uczenia głębokiego i pozwoliły sprawdzić ich możliwości w konfrontacji z nowymi zastosowaniami.*

*Wszystkie przedstawione cele zostały zrealizowane, a finalne rozwiązanie zostało wypracowane na podstawie analizy porównawczej wyników uzyskanych przy zastosowaniu różnych architektur głębokich sieci neuronowych oraz wskazanych typów zobrazowań. Na podstawie uzyskanych wyników zaproponowano trzy modele odpowiadające poszczególnym podzadaniom dotyczącym detekcji i klasyfikacji faz rozwojowych, poziomów nawodnienia i porażenia kukurydzy wybranymi patogenami. Zaproponowane podejścia zostały skonfrontowane z podobnymi rozwiązaniami dostępnymi w literaturze, co podkreśliło innowacyjność opracowanych w ramach niniejszej rozprawy metod.*

*Rozwiązania zaproponowane w rozprawie zostały wdrożone w rzeczywistym projekcie Robot Polski NCBiR, którego celem jest stworzenie robota polowego spełniającego wymogi rolnictwa precyzyjnego. Ponadto, rozszerzone wyniki badań stały się również wkładem merytorycznym do innego projektu pn. Fitoexport NCBiR. Fakty te świadczą o rzeczywistej aplikowalności opracowanego rozwiązania oraz o istnieniu zapotrzebowaniu na tego typu badania.*

**Słowa kluczowe:** *głębokie sieci neuronowe, sztuczna inteligencja, analiza obrazów RGB, analiza obrazów wielospektralnych, rolnictwo precyzyjne*



prof. dr hab. inż. Rafał Scherer  
Katedra Sztucznej Inteligencji  
Wydział Informatyki i Sztucznej Inteligencji  
Politechnika Częstochowska  
al. Armii Krajowej 36  
42-200 Częstochowa

Częstochowa, dn. 15 maja 2025 r.

## **Recenzja**

rozprawy doktorskiej mgr inż. Justyny Stypułkowskiej, pt.: Analiza obrazów wielospektralnych i RGB z zastosowaniem głębokich sieci neuronowych dla potrzeb automatycznej oceny parametrów kondycji kukurydzy przez robota polowego.

Niniejszą recenzję opracowano zgodnie z uchwałą nr 811/2024 Rady naukowej dyscypliny informatyka techniczna i telekomunikacja PW. Promotorem jest prof. dr hab. inż. Przemysław Rokita.

### **1. Charakterystyka tematu, celu i tezy badawczej rozprawy**

Tematyka rozprawy wpisuje się w kluczowe wyzwania współczesnego rolnictwa precyzyjnego. W dobie rosnącej populacji, zmian klimatycznych i presji na zrównoważone gospodarowanie zasobami, opracowanie narzędzi umożliwiających precyzyjny monitoring upraw ma fundamentalne znaczenie dla bezpieczeństwa żywnościowego i ochrony środowiska. Głównym celem pracy jest stworzenie systemu opartego na sztucznej inteligencji, który integruje analizę obrazów wielospektralnych i RGB z autonomicznym działaniem robota polowego. Autorce przyświeca idea zastąpienia tradycyjnych, czasochłonnych metod oceny kondycji roślin (np. manualnej inspekcji pól) przez zautomatyzowane rozwiązania oparte na danych. Badania koncentrują się na trzech kluczowych parametrach agronomicznych: identyfikacji faz rozwojowych w skali BBCH, klasyfikacji poziomów nawodnienia oraz wykrywaniu porażenia wybranymi patogenami. Temat nabiera dodatkowej wagi w świetle wyzwań związanych z adaptacją rolnictwa do zmian klimatu. Opracowane rozwiązania nie tylko przyczyniają się do zwiększają efektywność produkcji, ale także do redukcji śladu węglowego poprzez precyzyjne zarządzanie agrochemikaliami.

### **2. Zawartość rozprawy**

Recenzowana praca mgr inż. Justyny Stypułkowskiej składa się z dziewięciu rozdziałów, wykazu oznaczeń i skrótów, zmiennych, spisu rysunków, tabel, bibliografii oraz załączników. Dokument liczy 214 stron.

Pierwszy rozdział jest krótkim wprowadzeniem do tematyki, omówieniem projektów z którymi związane były prace wykonane w ramach rozprawy doktorskiej; zawiera również cele pracy: analiza faz rozwoju kukurydzy (w skali BBCH) przez zastosowanie głębokich sieci neuronowych do automatycznej detekcji i klasyfikacji na podstawie obrazów RGB i wielospektralnych, wybór najlepszego algorytmu i typu obrazowania; ocena poziomu nawodnienia kukurydzy za pomocą sieci neuronowych w analizie obrazów RGB i wielospektralnych w celu detekcji i klasyfikacji stanu nawodnienia; wykrywanie chorób kukurydzy przez automatyczną detekcję i klasyfikację porażenia patogenami na podstawie obrazów RGB z użyciem najskuteczniejszego algorytmu; stworzenie autorskich zbiorów danych składających się z oznaczonych obrazów (RGB i wielospektralnych) podzielonych na zestawy treningowe, walidacyjne i testowe, pochodzących z pola testowego. Rozdział kończy omówienie zawartości rozprawy.

Rozdział drugi omawia sieci neuronowe w zastosowaniach wizji komputerowej. Omówiono neurony, sieci typu nieliniowy MLP, oraz zasadę uczenia sieci z danych oraz dokładnie działanie sieci splotowych (CNN). Omówiono również popularne architektury jak odmiany RsNet oraz Transformer (HTC, Mask2Former, ConvNeXt, EfficientNet, Vision Transformer, Swin Transformer). Dalej omówiono specyfikę przetwarzania obrazów dwuwymiarowych składających się z wielu kanałów, szczególnie wielospektralnych oraz podstawowe sposoby oceny jakości działania systemów uczących się. Jest to istotny przegląd istniejących rozwiązań w dziedzinie monitoringu upraw, w którym autorka nie tylko prezentuje aktualny stan wiedzy, ale także, co szczególnie cenne, identyfikuje luki badawcze, uzasadniając tym samym potrzebę i nowatorski charakter własnych badań. Wnikliwa analiza literaturowa obejmuje zarówno metody oparte na tradycyjnym przetwarzaniu obrazów, jak i najnowsze rozwiązania wykorzystujące głębokie uczenie.

Rozdział 3 omawia dane użyte w badaniach oraz proces ich przygotowania i oznaczania. Autorka przedstawia kompleksową metodykę przygotowania danych, implementacji modeli oraz procesów trenowania i walidacji, co stanowi solidną podstawę dla uzyskanych wyników. Przedstawiono również architektury sieci neuronowych zastosowane do detekcji i klasyfikacji obiektów, metody ich trenowania oraz opracowany klasyfikator głoszący, który poprawia wyniki analizy. Rozdział rozpoczyna się od opisu metodyki przygotowania danych, w tym pozyskiwania i oznaczania obrazów RGB oraz wielospektralnych z kamer MicaSense RedEdge MX DUAL i MapIR Survey3 Red+Green+NIR. Dane zostały zebrane z pól testowych, a następnie poddane procesowi oznaczania, który obejmował ręczną adnotację faz rozwojowych kukurydzy w skali BBCH, poziomów nawodnienia oraz oznak porażenia patogenami. Wykorzystano narzędzie Label Studio, co zapewniło spójność i dokładność oznaczeń. Dane zostały podzielone na zbiory treningowe, walidacyjne i testowe, co umożliwiło rzetelną ocenę modeli. Autorka ogólnie opisuje implementację zaawansowanych architektur głębokich sieci neuronowych, w tym HTC (HTC\_x101, HTC\_r101, HTC\_r50), Mask2Former, ConvNeXt Small, ResNet50, EfficientNet-B3, Vision Transformer i Swin Transformer. Każdy z modeli został dostosowany do specyfiki zadania, takiego jak detekcja faz rozwojowych, klasyfikacja

poziomów nawodnienia czy identyfikacja patogenów. Wykorzystano frameworki m.in. PyTorch i MMDetection, co zapewniło efektywność obliczeniową i skalowalność rozwiązań. W rozdziale szczegółowo omówiono proces trenowania modeli, uwzględniając techniczne aspekty, takie jak optymalizacja hiperparametrów (np. współczynnik uczenia, liczba epok) oraz zastosowanie technik augmentacji danych. Autorka zastosowała metodę walidacji *hold-out*, która pozwoliła na ocenę skuteczności modeli na niezależnym zbiorze testowym. Monitorowano metryki takie jak *bbox mAP* i *segm mAP*, co umożliwiło precyzyjną ocenę jakości detekcji i segmentacji. W końcowej części rozdziału opisano proces detekcji i klasyfikacji obiektów, w tym zastosowanie klasyfikatora głosującego dla danych wielospektralnych. Model ten agregował wyniki z różnych kanałów spektralnych, zwiększając dokładność predykcji. Autorka podkreśliła również znaczenie interpretacji wyników i analizy błędów, co pozwoliło na identyfikację obszarów wymagających poprawy.

W rozdziale 4 autorka prezentuje wyniki badań dotyczących detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, wykorzystując obrazy RGB oraz wielospektralne z kamery MicaSense RedEdge MX Dual. Kluczowym elementem jest porównanie skuteczności obu typów obrazowania oraz ocena różnych architektur głębokich sieci neuronowych. Autorka przedstawia kompleksowy schemat opracowanego systemu, obejmujący: przetwarzanie wstępne (kalibracja i korekcja obrazów wielospektralnych, normalizacja danych RGB), implementację modeli (zastosowanie architektur HTC, Mask2Former i innych do detekcji obiektów oraz segmentacji instancji) oraz klasyfikację faz BBCH (przypisanie roślin do odpowiednich klas rozwojowych na podstawie analizy liści). Obrazy wielospektralne, dzięki dodatkowym kanałom (np. NIR, bliska podczerwień), zapewniły wyższą dokładność w zadaniu detekcji faz rozwojowych niż obrazy RGB. Najlepsze wyniki osiągnięto dla modeli HTC\_x101 i Mask2Former, które skutecznie radziły sobie z różnymi warunkami oświetleniowymi i nakładaniem się roślin. Wykorzystanie danych wielospektralnych pozwoliło na lepszą segmentację liści, co przełożyło się na dokładniejsze określenie fazy BBCH. Modele oparte na RGB również osiągnęły dobre rezultaty, ale były bardziej podatne na zakłócenia związane z cieniami i zmiennymi warunkami świetlnymi. Wykazano, że obrazy RGB są wystarczające do podstawowej klasyfikacji, ale w przypadku wymagających warunków ich skuteczność spada. Z powodu problemów z identycznością obrazów poszczególnych kanałów multispektralnych opisanych w rozdziale 3, autorka zastosowała klasyfikator głosujący, który agregował wyniki z różnych kanałów spektralnych. Dzięki temu osiągnięto wyższą precyzję w porównaniu do pojedynczych modeli oraz lepsze uogólnienie na różne warunki środowiskowe. Analiza krzywych uczenia wykazała, że modele oparte na danych wielospektralnych szybciej osiągały zbieżność i miały niższy błąd walidacyjny. Obrazy wielospektralne zapewniły znacznie lepszą dokładność niż RGB w detekcji faz rozwojowych kukurydzy.

Rozdział piąty pracy doktorskiej kontynuuje temat detekcji faz rozwojowych kukurydzy w skali BBCH, skupiając się tym razem na analizie obrazów pozyskanych kamerą wielospektralną MapIR Survey3 (Red+Green+NIR) oraz porównaniu ich skuteczności z tradycyjnym obrazowaniem RGB. Autorka prezentuje spójną metodologię badawczą, w której szczególną

uwagę poświęca różnicom w efektywności obu typów obrazowania oraz optymalizacji procesu klasyfikacji. Kluczowym elementem rozdziału jest szczegółowa analiza wyników uzyskanych dla różnych architektur sieci neuronowych. Podobnie jak w poprzednim rozdziale, najlepsze rezultaty osiągnięto dla modeli HTC i Mask2Former, jednak w tym przypadku autorka zwraca szczególną uwagę na specyficzne zalety kamery MapIR Survey3. Jej zdolność do rejestracji kanału bliskiej podczerwieni (NIR) okazała się szczególnie cenna w identyfikacji subtelnych różnic w stanie fizjologicznym roślin, co przełożyło się na wyższą dokładność klasyfikacji w porównaniu do obrazów RGB. Co istotne, rozdział zawiera pogłębioną dyskusję na temat wpływu poszczególnych kanałów spektralnych na skuteczność detekcji, co stanowi wartościowy wkład w rozwój metod analizy obrazów w rolnictwie precyzyjnym. Autorka przedstawia również interesujące porównanie krzywych uczenia dla różnych konfiguracji modeli, wskazując na szybszą zbieżność i niższe wartości błędów w przypadku wykorzystania danych wielospektralnych. Szczegółowa analiza metryk jakościowych (mAP, precision, recall) potwierdza przewagę obrazowania wielospektralnego, zwłaszcza w trudniejszych warunkach, takich jak gęste ulistnienie czy nakładanie się roślin. Rozdział zawiera także cenne uwagi praktyczne dotyczące implementacji systemu w rzeczywistych warunkach polowych, co świadczy o dbałości autorki o aspekty aplikacyjne prowadzonych badań.

Rozdział szósty pracy doktorskiej wprowadza nowy, istotny wątek badawczy dotyczący automatycznej oceny stanu nawodnienia kukurydzy z wykorzystaniem obrazowania RGB i wielospektralnego. Autorka w sposób systematyczny prezentuje kompleksowe podejście do problemu, rozpoczynając od szczegółowego opisu przyjętej metodyki, przez analizę wyników, aż po praktyczne wnioski implementacyjne. Kluczowym osiągnięciem tego rozdziału jest opracowanie i walidacja trójklasowego systemu klasyfikacji (małe, średnie i duże nawodnienie), który wykorzystuje zarówno wizualne cechy morfologiczne roślin (widoczne w obrazach RGB), jak i charakterystyczne sygnatury spektralne (rejestrowane przez kamerę MapIR Survey3). Szczególnie wartościowe są przedstawione przez autorkę wyniki porównawcze, które jednoznacznie wskazują na wyższą skuteczność obrazowania wielospektralnego.

Rozdział siódmy pracy doktorskiej koncentruje się na kluczowym zagadnieniu diagnostyki fitosanitarnej, tj. automatycznej identyfikacji objawów porażenia kukurydzy przez wybrane patogeny. Autorka w sposób metodyczny prezentuje kompleksowe rozwiązanie oparte wyłącznie na analizie obrazów RGB, co stanowi istotne novum w kontekście dotychczasowych badań wykorzystujących głównie dane spektralne. Centralnym elementem rozdziału jest opracowanie i walidacja systemu detekcji głównych patogenów oraz śladów żerowania szkodnika. Autorka szczegółowo opisuje proces tworzenia specjalistycznego zbioru danych, uwzględniającego zróżnicowane stadia rozwoju objawów chorobowych, co miało kluczowe znaczenie dla skuteczności późniejszej klasyfikacji. W rozdziale szczególnie wartościowa jest pogłębiona analiza charakterystycznych cech wizualnych wykorzystywanych przez algorytmy do identyfikacji patogenów. Autorka przedstawia innowacyjne podejście polegające na integracji systemu diagnostycznego z platformą robota polowego, ze szczególnym

uwzględnieniem optymalizacji parametrów akwizycji obrazów w warunkach polowych. Praktyczne aspekty implementacji, w tym analiza wpływu warunków oświetleniowych na skuteczność detekcji oraz propozycje algorytmów kompensacji zmiennego natężenia światła, stanowią szczególnie wartościowy element pracy.

Rozdział ósmy stanowi praktyczne zwieńczenie badań przedstawionych w pracy, koncentrując się na implementacyjnych aspektach opracowanego systemu do automatycznej oceny kondycji kukurydzy. Autorka prezentuje dwa kluczowe elementy systemu: interaktywną aplikację opartą na bibliotece Gradio oraz mechanizm eksportu wyników do plików CSV, które razem tworzą kompletną platformę analityczną dla użytkowników końcowych. Aplikacja wykorzystująca bibliotekę Gradio została zaprojektowana jako intuicyjne narzędzie umożliwiające przetwarzanie obrazów w czasie rzeczywistym. Autorka szczegółowo opisuje jej modułową architekturę, która integruje wcześniej wytrenowane modele do detekcji faz rozwojowych, poziomów nawodnienia i patogenów. Interfejs użytkownika został zoptymalizowany pod kątem praktycznego zastosowania w rolnictwie precyzyjnym, oferując możliwość przesyłania pojedynczych obrazów lub całych sekwencji z pól uprawnych. W aplikacji zaimplementowano także funkcje wizualizacji wyników, w tym nakładanie masek segmentacyjnych i podświetlanie obszarów zainteresowania, co znacznie ułatwia interpretację wyników przez agronomów. Drugim istotnym komponentem systemu jest generowanie plików CSV zawierających szczegółowe parametry przetwarzania oraz wyniki analizy. Mechanizm ten umożliwia łatwą integrację z istniejącymi systemami zarządzania uprawami oraz dalszą analizę statystyczną zgromadzonych danych. W rozdziale podkreślono także znaczenie standaryzacji formatu wyjściowego, co jest szczególnie istotne w kontekście przyszłej skalowalności rozwiązania.

Rozdział 9 podsumowuje rozprawę oraz proponuje dalsze możliwości rozbudowy zaproponowanych metod. Dalej następuje spis ilustracji i tabel, bibliografia oraz załączniki.

### 3. Ocena rozprawy

W ramach rozprawy doktorskiej Pani mgr inż. Justyna Stypułkowska zaproponowała oryginalne rozwiązanie, które łączy nowatorskie rozwiązania uczenia maszynowego z aplikacjami dla rolnictwa precyzyjnego, oferując narzędzia do monitoringu upraw w skali pojedynczych roślin. Tematyka pracy jest bardzo aktualna i potrzebna, oryginalny dorobek autorki polega na:

- stworzeniu systemu automatyzującego ocenę kondycji kukurydzy, integrującego detekcję faz rozwojowych (skala BBCH), poziomów nawodnienia i porażenia patogenami z wykorzystaniem robota polowego,
- udowodnieniu przewagi obrazowania wielospektralnego nad RGB,
- zaproponowanie klasyfikatora głoszącego dla danych wielospektralnych co poprawiło stabilność predykcji poprzez agregację wyników z różnych kanałów spektralnych,

- detekcji patogenów wyłącznie na bazie obrazów RGB, pomijając kosztowne dane spektralne,
- stworzeniu praktycznej implementacji jako aplikacji i systemu eksportu wyników (CSV) gotowych do wdrożenia w gospodarstwach,
- stworzenie autorskich zbioru danych faz BBCH i poziomów nawodnienia, wypełniając lukę badawczą,
- walidację w warunkach zbliżonych do rzeczywistych przez testy na poletkach doświadczalnych z użyciem prototypu robota polowego.

Rozprawa doktorska uwidacznia wysoką ogólną wiedzę teoretyczną i praktyczną oraz umiejętność prowadzenia pracy naukowej mgr inż. Justyny Stypułkowskiej. Opracowała wprowadzenie do tematyki i przegląd literatury na temat architektur sieci neuronowych związanych z tematyką pracy. Rozprawa doktorska stanowi oryginalne rozwiązanie problemu naukowego. Zaproponowane metody mają duże znaczenie dla nauk technicznych oraz przemysłu, zarówno teoretyczne, jak i aplikacyjne.

Niezależnie od mojej dobrej oceny pracy, nasunęły mi się następujące pytania i uwagi:

- Podstawowe sieci typu MLP i CNN omówione są bardzo dokładnie, a wszystkie złożone architektury używane w badaniach bardzo pobieżnie.
- Brak jest szczegółowej dyskusji na temat geograficznej i sezonowej zmienności danych treningowych (np. czy uwzględniono różne regiony klimatyczne lub ekstremalne warunki pogodowe), co może ograniczać generalizację modeli w rzeczywistych warunkach polowych. Niewystarczająco opisano metody radzenia sobie z nie zrównoważonymi klasami (np. rzadkie fazy BBCH lub rzadkie patogeny), co mogło wpłynąć na wyniki metryk.
- Brak jest omówienia problemów z oznaczaniem obrazów RGB i wielospektralnych; kto dokonywał takiej adnotacji?
- Brak jest podania dokładnych meta parametrów użytych sieci neuronowych oraz szczegółów jak obrazy wielospektralne były przetwarzane przez warstwy splotowe.
- Część zawartości, np. Tabela 4.1 (wklejona jako obraz) oraz np. Tabela 4.4 pochodzą z pracy Stypułkowska, J., & Rokita, P. (2025). Classification of Maize Growth Stages Using Deep Neural Networks with Voting Classifier. Machine Graphics and Vision, 3-4):29–53, 2024, bez jej cytowania.
- Zazwyczaj zamiast pisać np. „algorytm HTC”, mówimy o modelu czy architekturze HTC.
- Do oznaczenia wymiaru czasami stosowany jest znak x, a czasami ×.

#### 4. Wnioski końcowe recenzji

Podsumowując recenzję stwierdzam, że rozprawa doktorska „Analiza obrazów wielospektralnych i RGB z zastosowaniem głębokich sieci neuronowych dla potrzeb automatycznej oceny parametrów kondycji kukurydzy przez robota polowego” prezentuje

oryginalne rezultaty stanowiące rozwiązanie problemu naukowego oraz wkład w rozwój dyscypliny informatyka techniczna i telekomunikacja. Pani Justyna Stypułkowska wykazała się umiejętnością samodzielnej pracy badawczej, znajomością literatury światowej i wiedzą w zakresie wizji komputerowej. Rozprawa doktorska prezentuje ogólną wiedzę teoretyczną kandydatki. Recenzowana praca spełnia wymagania Ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz.U. z 2022 r. poz. 574 z późn. zm.). w dyscyplinie naukowej informatyka techniczna i telekomunikacja. Wnoszę o jej przyjęcie i dopuszczenie do dalszych etapów postępowania doktorskiego.

