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 pisania rozprawy.

Podziękowania kieruję również moim przełożonym z Sieci Badawczej Łukasiewicz – Instytutu Lotnictwa: Panu dr inż. Pawłowi Stężyckiemu – Dyrektorowi Sieci Badawczej Łukasiewicz – Instytutu Lotnictwa, Panu mgr inż. Mariuszowi Kacprzakowi – Kierownikowi Działu Teledetekcji oraz Panu mgr inż. Karolowi Rotchimmelowi – Kierownikowi Sekcji Rozwoju Narzędzi Teledetekcyjnych, pełniącemu również rolę Opiekuna pracy. Dzięki ich wsparciu możliwa była realizacja prac badawczych oraz dostęp do niezbędnego sprzętu obliczeniowego i kamer do pozyskiwania danych.

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 wybranymi 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 przetestowane i wykorzystane w rzeczywistym projekcie pn. „Inteligentny robot spełniający wymogi rolnictwa precyzyjnego”, finansowanym przez NCBiR oraz Komisję Europejską. Ponadto, rozszerzone wyniki badań stały się również wkładem merytorycznym do innego projektu pn. „Fitoexport”, finansowanego przez 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 infestation of maize with selected pathogens. 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 infestation 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 tested and used in the project entitled "Intelligent Robot Meeting the Requirements of Precision Agriculture", funded by NCBiR and the European Commission. Furthermore, the extended research findings have also made a contribution to another project, entitled "Fitoexport", funded by NCBiR. These facts demonstrate the practical applicability of the developed solution and the existing demand for this type of research.

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

Spis treści

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 odgrywa coraz większą 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 detekcji i klasyfikacji obiektów, przetwarzania i analizy obrazów, a także podejmowania decyzji na podstawie analizy dużych zbiorów danych. W kontekście analizy obrazów szczególną rolę odgrywają 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 specyficznych zbiorów danych, które są niezbędne do trenowania

zaawansowanych modeli uczenia maszynowego. 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 danych z 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 opracowanych modeli. 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 ważną rolę, zwłaszcza w monitorowaniu upraw w czasie rzeczywistym oraz precyzyjnym zarządzaniu środkami ochrony roślin, nawozami, nawadnianiem 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 szersze 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 przeglądy badań i rozwiązań dotyczących zastosowania uczenia głębokiego w rolnictwie, które pokazują, że głębokie uczenie zapewnia wysoką dokładność, przewyższając istniejące powszechnie stosowane techniki przetwarzania obrazów [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 również 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 Minguo Liu i współautorzy [Liu, M., 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, M., 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 przy 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 w wiedzy, którą warto się zająć.

Celem niniejszej pracy jest wypełnienie wspomnianych luk badawczych poprzez opracowanie 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 zobrazowania 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 rzeczywistych projektach 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. Na rysunku 1.1. zaprezentowano wygląd prototypu robota polowego opracowanego w wyniku prac nad projektem *„Inteligentny robot spełniający wymogi rolnictwa precyzyjnego”*. Rozwiązania opracowane w ramach przygotowywania niniejszej rozprawy zostały przetestowane w tymże prototypie.

Rysunek 1.1. Prototyp robota polowego opracowanego w wyniku prac nad projektem *„Inteligentny robot spełniający wymogi rolnictwa precyzyjnego”*, autor zdjęcia: mgr inż. Jakub Szymański, Sieć Badawcza Łukasiewicz – Instytutu Lotnictwa.

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 projektowe i 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 algorytm 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.

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 wykorzystanych w ramach przeprowadzonych prac. Dokonano rzetelnego opisu przyjętej metodyki przygotowywania danych i prowadzenia badań, opisano zbiory zebranych 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 MicaSense RedEdge MX Dual. 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 Survey3 Red+Green+NIR. Przedstawiono schemat opracowanego rozwiązania, wyniki uzyskane dla danych wielospektralnych oraz wyniki uzyskane 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 z kamery MapIR Survey3 Red+Green+NIR i danych z kamery 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, analizę zarejestrowanych krzywych uczenia oraz porównanie wyników i wybór najlepszego rozwiązania.
- **Rozdział 8** opisuje szczegóły dotyczące prezentacji wyników działania opracowanego systemu w formie stworzonej aplikacji wykorzystującej bibliotekę Gradio oraz w formie pliku CSV agregującego wyniki z działania opracowanego zestawu wytrenowanych modeli głębokich sztucznych sieci neuronowych, a także zawierającego niezbędne parametry przetwarzania.
- **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.

2. Przegląd literatury i wykorzystywanych technik

W niniejszym rozdziale przedstawiono przegląd literatury i technologii wykorzystywanych jako podstawa teoretyczna dla opracowania rozwiązań będących przedmiotem niniejszej rozprawy. Szczególną uwagę poświęcono sztucznym sieciom neuronowym. Omówiono podstawy teoretyczne dotyczące konwolucyjnych sieci neuronowych oraz sieci typu transformer wraz z wyjaśnieniem 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 [He i in., 2016], ResNet101 [He i in., 2016], ResNeXt101 [Xie i in., 2017] oraz zaawansowane architektury, takie jak HTC [Chen i in., 2019], Mask2Former [Chen i in., 2021], ConvNeXt Small [Liu i in., 2022], EfficientNet-B3 [Tan, Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] oraz Swin Transformer [Liu i in., 2021].

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 niniejszej rozprawy, gdyż skupia się ona na automatycznym określaniu tychże parametrów kondycji roślin na podstawie analizy obrazów.

Rozdział spina przegląd podobnych prac badawczych wraz z analizą wykorzystywanych rozwiązań, co było wyjściem do podsumowania kluczowych wniosków, 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 właśnie sposobu

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 między neuronami. 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 tzw. aksonu. Warren McCulloch i Walter 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 warto rozpocząć od przedstawienia podstaw koncepcji perceptronu, które opierają się na modelu neuronu zaproponowanym przez Warrena McCullocha i Waltera Pittsa, którzy na łamach Bulletin of Mathemathical 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 później 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 czy nie. 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 ∅(z), która jest kombinacją wartości wejściowych x i powiązanego wektora wag w, gdzie z jest całkowitym pobudzeniem układu [Rashka, 2018]:

$$z = w_1 x_1 + \cdots + w_m x_m$$

Wektory te mają następującą postać [Rashka, 2018]:

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

W przypadku, gdy pobudzenie z dla wybranej próbki $x^{(i)}$ przekracza założoną wartość progową θ, można zaliczyć obiekt do klasy pozytywnej, a w przeciwnym przypadku do klasy negatywnej. W perceptronie funkcja aktywacji $\emptyset(\cdot)$ jest funkcją skoku jednostkowego [Rashka, 2018]:

$$\emptyset(z) = \begin{cases} 1 \; jeśli \; z \geq \theta \\ -1 \; jeśli \; z < \theta \end{cases}$$

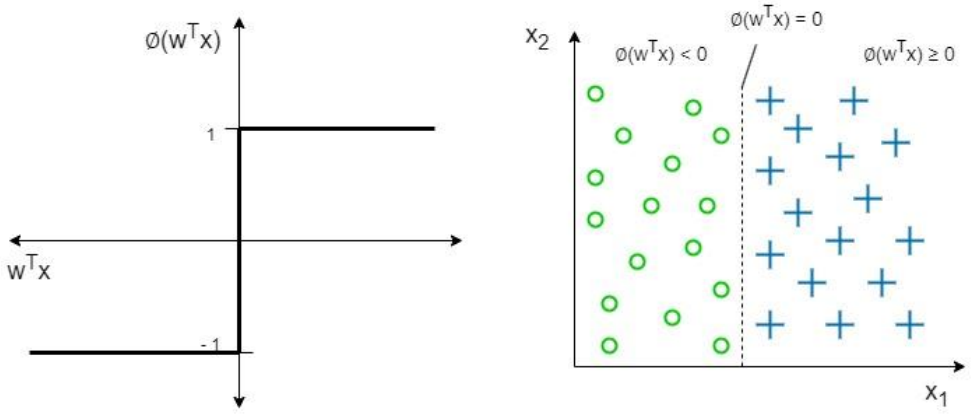
Aby uprościć ten zapis, należy wartość progową θ przenieść na lewą stronę równania, określając początkową wagę w postaci $w_0 = -\theta$ oraz $x_0 = 1$. Wówczas całkowite pobudzenie z można zapisać w następującej formie [Rashka, 2018]:

$$z = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = w^T x,$$

zakładając, że:

$$\emptyset(z) = \begin{cases} 1 \; jeśli \; z \geq 0 \\ -1 \; jeśli \; z < 0 \end{cases}.$$

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 uczeniu 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 nieaktywnym. Dzięki temu mechanizmowi reguła uczenia perceptronu, opracowana przez Franka Rosenblatta, charakteryzuje się prostotą i można ją opisać w następujących krokach [Rashka, 2018]:

1. inicjalizacja wag z wartościami równymi 0 lub losowymi, małymi wartościami.
2. wykonanie następujących czynności dla każdej próbki uczącej $x^{(i)}$:
   - obliczenie wartości wyjściowej $\hat{y}$
   - aktualizacja wag [Rashka, 2018].

Wartością uzyskiwaną na wyjściu jest etykieta klasy prognozowana przez określoną wcześniej funkcję skoku jednostkowego. Zmianę każdej wagi w_j w wektorze wag w można wyrazić jako [Rashka, 2018]:

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

Aktualizacja wagi Δw_j jest obliczana zgodnie z regułą uczenia perceptronu [Rashka, 2018]:

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

gdzie:

η - 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 $y^{(i)}$, dopóki nie zostaną odświeżone wszystkie wagi Δw_j [Rashka, 2018].

Uaktualnienia dla dwuwymiarowego zbioru danych można określić jako [Rashka, 2018]:

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

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

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

Przedstawiona reguła uczenia jest prosta i zawiera się w dwóch scenariuszach, w których perceptron we właściwy sposób przewiduje etykietę klas, a wagi nie ulegają zmianie [Rashka, 2018]:

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

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

W sytuacji błędnego przewidywania wagi są modyfikowane w kierunku pozytywnej bądź negatywnej klasy docelowej, zgodnie z następującymi równaniami [Rashka, 2018]:

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

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

W celu lepszego zrozumienia idei mnożnika $x_j^{(i)}$, należy przyjrzeć się następującemu przykładowi [Rashka, 2018]:

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

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

$$\eta = 1$$

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 próbka ta zostanie napotkana ponownie. W rezultacie badany obiekt z większym

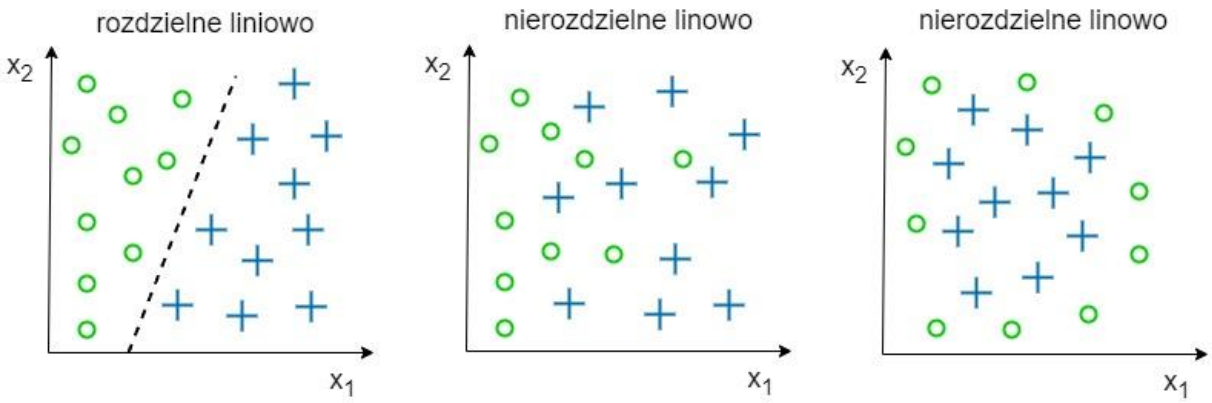
prawdopodobieństwem zostanie przypisany do klasy $+1$ po przekroczeniu wartości progowej funkcji skokowej [Rashka, 2018]:

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

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 jeszcze bardziej przesunąć granicę decyzyjną w celu poprawnej klasyfikacji próbki następnym razem [Rashka, 2018]:

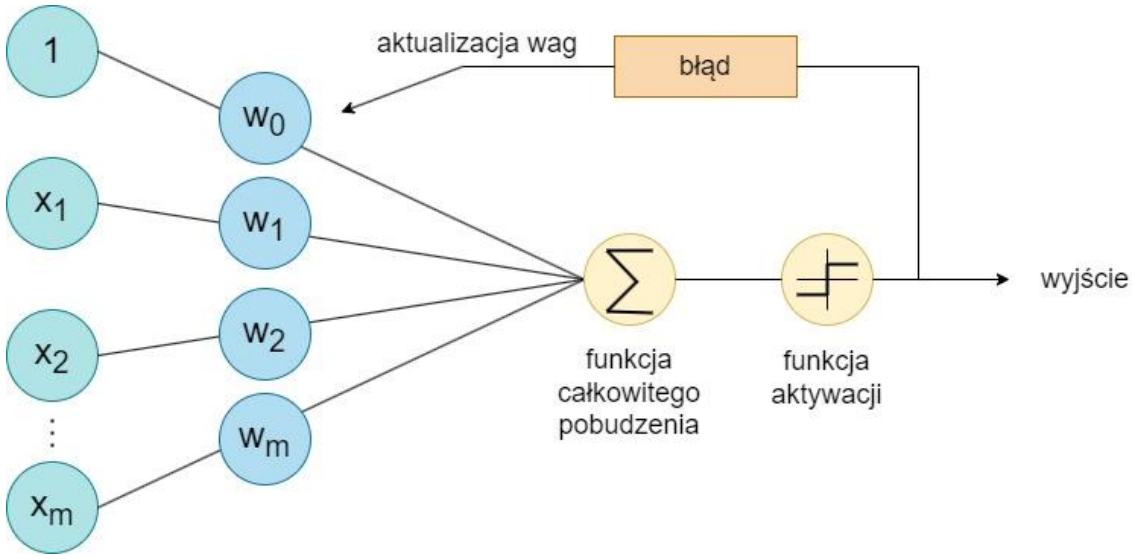
$$\Delta w_j = (1 - -1)2 = (2)2 = 4$$

Zbieżność algorytmu perceptronu jest gwarantowana jedynie w przypadku, gdy dwie klasy są liniowo separowalne (jak pokazano na rysunku 2.2.), a współczynnik uczenia jest odpowiednio niski. Jeśli nie jest możliwe rozdzielenie tych klas za pomocą liniowej granicy decyzyjnej, można ustalić wtedy 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 widocznego na rysunku 2.3.



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

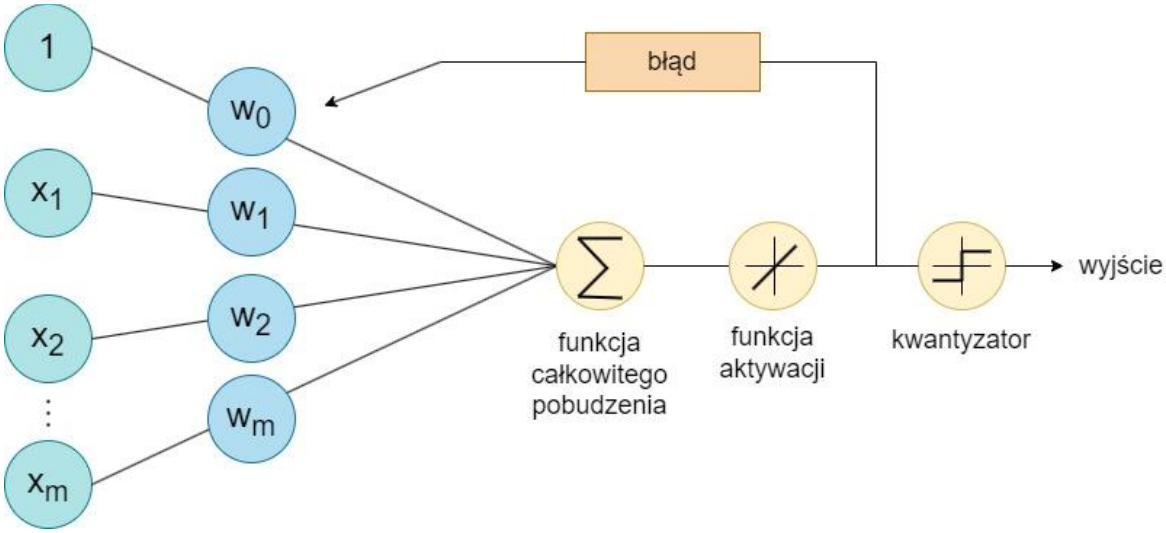
Rysunek 2.3 przedstawia schemat działania perceptronu, który otrzymuje dane wejściowe w postaci próbek x i łączy je z wagami w w celu obliczenia funkcji całkowitego pobudzenia. Wynik tej operacji przekazywany jest do funkcji aktywacji (w tym przypadku funkcji skoku jednostkowego), która generuje binarną 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 obrazów. Algorytm ten został wprowadzony kilka lat po perceptronie przez Bernarda Widrowa i Tedda Hoffa i można traktować go 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 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 [Rashka, 2018]:

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

Liniowa funkcja aktywacji służy do obliczania wag, a kwantyzator, zbliżony do funkcji skoku jednostkowego, jest używany do prognozowania etykiet klas, co pokazuje rysunek 2.4. [Rashka, 2018]:



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

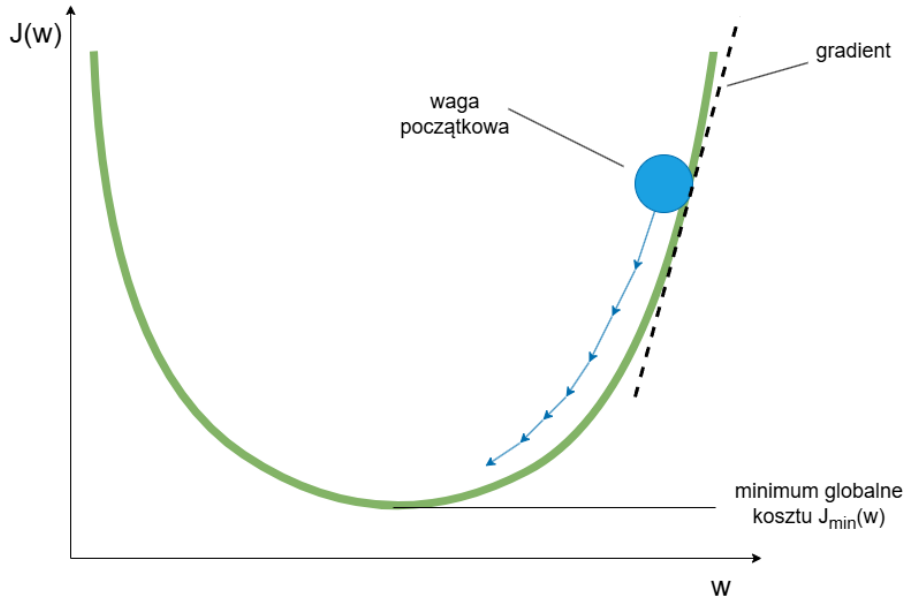
Porównując schemat perceptronu ze schematem Adaline, główna różnica polega na tym, że Adaline wykorzystuje ciągłe wartości, obliczane przy pomocy liniowej funkcji aktywacji, do wyznaczania błędu modelu i aktualizacji wag, zamiast binarnych etykiet klas, jak ma to miejsce w perceptronie. W kontekście 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. Zależność tą 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 rysunku 2.5. przedstawiono, jak algorytm gradientu prostego można porównać do procesu tzw. „schodzenia z górki" 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 również 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 [Rashka, 2018]:

$$w \coloneqq w + \Delta w$$

Zmianę wagi Δw zdefiniowano jako ujemny gradient pomnożony przez współczynnik uczenia η zgodnie z poniższym równaniem [Rashka, 2018]:

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

W celu wyliczenia gradientu funkcji kosztu należy obliczyć pochodną cząstkową tejże funkcji przy uwzględnieniu każdej z wag w_j [Rashka, 2018]

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

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 [Rashka, 2018]:

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

Jednocześnie są aktualizowane wszystkie wagi, więc reguła uczenia Adaline wygląda następująco [Rashka, 2018]:

$$w \coloneqq w + \Delta w$$

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].

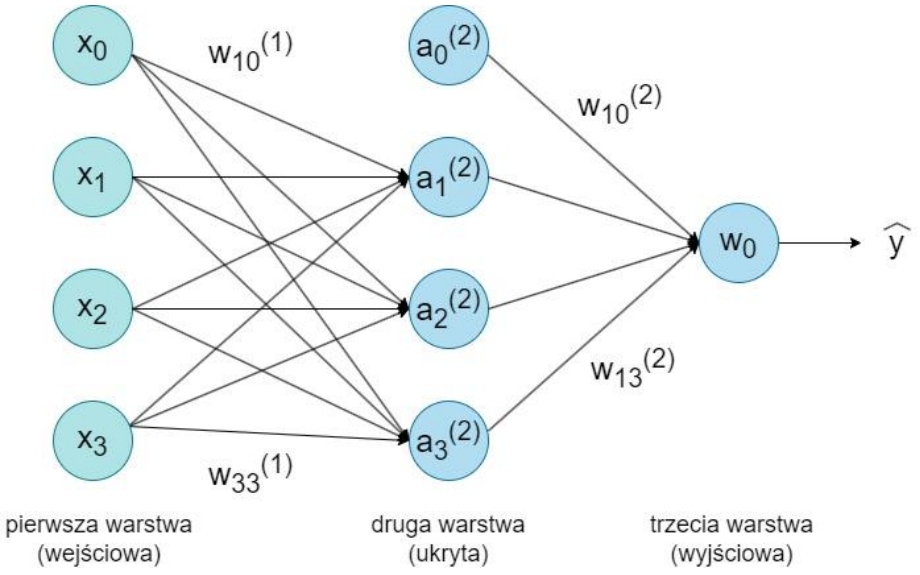
Po opracowaniu w latach pięćdziesiątych modelu perceptronu przez Franka Rosenblatta, bazującego na modelu McCollucha-Pittsa, zainteresowanie tym tematem znacząco zmalało ze względu na trudności związane z opracowaniem efektywnej metody uczenia wielowarstwowych sieci neuronowych. Przełom nastąpił w 1986 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 kolejnych 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 jednym z tematów 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 [Rashka, 2018].

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 tematyki 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 tą zależność [Rashka, 2018]:

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

Każda z jednostek znajdujących się w warstwie l jest połączona ze wszystkimi jednostkami w warstwie $l+1$ przy użyciu współczynnika wag. Dla przykładu, 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 [Rashka, 2018].

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 [Rashka, 2018]:

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

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 procesie uczenia 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 tychże 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 [Rashka, 2018]:

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

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

gdzie:

$z_1^{(2)}$ - to pobudzenie całkowite

$\phi(\cdot)$ - to funkcja aktywacji [Rashka, 2018].

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 sinusoidalna [Rashka, 2018]:

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

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 $z = 0$ [Rashka, 2018].

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 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 [Rashka, 2018].

W celu zwiększenia czytelności i efektywności kodu podczas implementacji, zalecane jest 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* [Rashka, 2018]:

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

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

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 [Rashka, 2018]:

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

gdzie:

$A^{(1)}$ - to $n \times [m + 1]$ – wymiarowa macierz [Rashka, 2018].

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 [Rashka, 2018]:

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

W podobny sposób można wyrazić zwektoryzowaną postać aktywacji warstwy wyjściowej [Rashka, 2018]:

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

Pomnożona została tutaj $t \times h$ - wymiarowa macierz $W^{(2)}$ (t to liczba jednostek wyjściowych) przez $h \times n$ – 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 zatem następująco [Rashka, 2018]:

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

2.1.2. Uczenie głębokie i propagacja wsteczna

W procesie trenowania sztucznej sieci neuronowej w rozpoznawaniu obrazów niezwykle istotne jest zrozumienie kluczowych koncepcji optymalizacji wag, takich jak logistyczna funkcja kosztu oraz algorytm wstecznej propagacji. Elementy te są fundamentalne dla skutecznego trenowania modelu, szczególnie w kontekście zadania klasyfikacji obrazów [Rashka, 2018].

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

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

gdzie:

$a^{(i)}$ - sigmoidalna aktywacja *i*-tej jednostki w jednej z warstw [Rashka, 2018].

Wartość $a^{(i)}$ obliczana jest jako funkcja propagacji jednokierunkowej [Rashka, 2018]:

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

Aby zminimalizować ryzyko przetrenowania modelu, dodaje się wyrażenie regularyzacji, które zmniejsza efekt przetrenowania. Regularyzację można przeprowadzić na dwa sposoby: L1 i L2, które są definiowane następująco (jednostki obciążenia nie są regularyzowane) [Rashka, 2018]:

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

Po dodaniu regularyzacji L2 do logistycznej funkcji kosztu otrzymuje się następujące równanie [Rashka, 2018]:

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

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 [Rashka, 2018]:

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

Funkcja kosztu (bez wyrażenia regularyzacji) wygląda następująco [Rashka, 2018]:

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

Uogólnione wyrażenie regularyzacji sprowadza się do zsumowania wszystkich wag w warstwie l (z wyjątkiem jednostki obciążenia), które zostały dodane do pierwszej kolumny [Rashka, 2018]:

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

Kara wyliczona z regularyzacji L2 przedstawia się za pomocą poniższego wyrażenia [Rashka, 2018]:

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

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 [Rashka, 2018]:

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

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 [Rashka, 2018].

Po wyjaśnieniu kluczowych koncepcji trenowania sztucznej sieci neuronowej, istotne jest omówienie algorytmu wstecznej propagacji, który odgrywa centralną rolę w optymalizacji wag. Algorytm ten jest jedną z najefektywniejszych technik trenowania sztucznych sieci neuronowych. Pozwala 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 minimum globalnego [Rashka, 2018].

Różniczkowanie złożonych, zagnieżdżonych funkcji jest możliwe dzięki tzw. regule łańcuchowej, która pozwala na rozbicie takich funkcji, jak $f\big(g(x)\big) = y$, na ich podstawowe składowe [Rashka, 2018]:

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

Z punktu widzenia algebry informatycznej opracowano wiele technik tzw. automatycznego różniczkowania, które rozwiązują tego rodzaju wyzwania. 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 [Rashka, 2018].

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, aby uzyskać wartości aktywacji warstwy wyjściowej, co można określić następująco [Rashka, 2018]:

- $Z^{(2)} = W^{(1)}\big[A^{(1)}\big]^T$ – 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 [Rashka, 2018].

Podczas działania algorytmu wstecznej propagacji błąd jest przesyłany od wyjścia do wejścia. Na początku obliczany jest błąd dla warstwy wyjściowej w postaci [Rashka, 2018]:

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

gdzie:

y – jest wektorem rzeczywistych etykiet klas [Rashka, 2018].

Następnie obliczany jest błąd dla warstwy ukrytej [Rashka, 2018]:

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

Pochodna sigmoidalnej funkcji aktywacji wygląda następująco [Rashka, 2018]:

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

Po obliczeniu wyrażeń δ, można zapisać wzór na pochodną funkcji kosztu w następującej formie [Rashka, 2018]:

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

Należy wyznaczyć również pochodną cząstkową dla każdego j-tego neuronu w warstwie i obliczyć i-ty błąd pochodzący z warstwy l+1. Operację tę można wykonać w formie zwektoryzowanej [Rashka, 2018]:

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

Po obliczeniu pochodnych cząstkowych można dodać wyrażenie regularyzacyjne [Rashka, 2018]:

$$\Delta^{(l)} \coloneqq \Delta^{(l)} + \lambda^{(l)}$$

W ostatnim etapie, po wyznaczeniu gradientów, wagi są modyfikowane poprzez wykonanie ruchu w kierunku przeciwnym do gradientu [Rashka, 2018]:

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

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. Technika ta 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 macierzy $W(1)$ i $W(2)$ w modelu wielowarstwowego perceptronu z jedną jednostkę ukrytą. Pochodna funkcji kosztu względem wagi $w_{i,j}^{(l)}$ jest wyrażona jako [Rashka, 2018]:

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

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 [Rashka, 2018]:

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

gdzie:

ε - przyjmuje wartości rzędu $1e^{-5}$ [Rashka, 2018].

Lepszą aproksymację gradientu można uzyskać, obliczając symetryczny iloraz różnicowy [Rashka, 2018]:

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

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 [Rashka, 2018]:

$$błąd = \|J_n' - J_a'\|_2$$

Dla większej dokładności zaleca się obliczanie znormalizowanej różnicy [Rashka, 2018]:

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

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 z propagacją w przód. Powstało jednak wiele innych architektur sieci neuronowych, spośród których warto 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 splotowych sieci neuronowych, tzw. CNN (od ang. *convolutional neural networks)* [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 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 splotowych 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 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, czyli 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 sieci 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 tzw. 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, jak już wcześniej wspomniano, 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 zasobach 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].

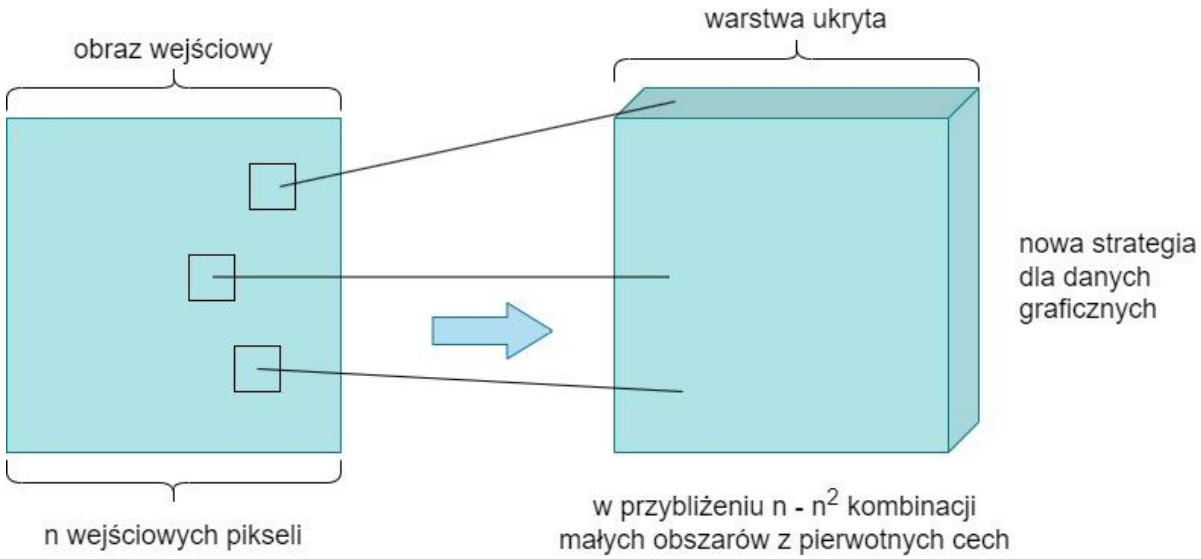
W kontekście sieci neuronowych i uczenia reprezentacji, sieci neuronowe przyjmują na wejściu dane obserwacyjne, gdzie każda obserwacja jest opisana przez zestaw określonej liczby

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].

Fakt ten znajduje zastosowanie tam, gdzie tworzenie kombinacji cech odbywa się w podobny sposób, jak dotychczas, jednak z większą liczbą tychże 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 [Weidman, 2019].



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], [Weidman, 2019].

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 wzorzec 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 [Weidman, 2019]:

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

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 wzorzec 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 wzorzec 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, to 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 wzorzec, a jego obecność w różnych miejscach obrazu wejściowego jest rejestrowana na 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], [Weidman, 2019].

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].

Warte uwagi jest to, 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 cechy te 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 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 × wysokość × 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 mającą m_1 kanałów 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 tychże 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 wzorzec 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 tychże 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 × wysokość obrazu × 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 procesie tym, 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 × wysokość obrazu × 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].

2.1.4. Architektury ResNet50, ResNet101, ResNeXt101

Wykorzystanie konwolucyjnych sieci neuronowych w uczeniu głębokim znacząco wpłynęło na obszar analizy obrazów. Zdolność tego typu sieci do przeprowadzania automatycznej ekstrakcji cech z obrazów umożliwiła ich zastosowanie zarówno w klasyfikacji i 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 krótce przybliżone sieci rezydualne, które zostały bezpośrednio użyte w badaniach przeprowadzonych w ramach realizowanego doktoratu. Będą nimi kolejno architektury: ResNet50 [He i in., 2016], ResNet101 [He i in., 2016] oraz ResNeXt101 [Xie i in., 2017].

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 i in., 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. połączeń skróconych (*shortcut connections*). 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 i in., 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 i in., 2016], [Simonyan, 2014].

Analogią to architektury ResNet50 [He i in., 2016] jest architektura ResNet101 [He i in., 2016] 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 i in., 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 [He i in., 2016].

Zupełnie nową, bardziej zaawansowaną wersją ResNet jest architektura typu ResNeXt101 [Xie i in., 2017], 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 [Xie i in., 2017]. W tym przypadku również wykorzystano bloki rezydualne, ale nowością jest wprowadzenie grupowych konwolucji. Dzięki temu ResNeXt101 jest zdolny do wykonywania równoległych operacji na wielu kanałach wejściowych zebranych w podgrupy. Co ważne, ResNeXt101 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 swego rodzaju podsumowania, architektura ResNet50 [He i in., 2016] zapewniła skuteczne trenowanie głębokich sztucznych sieci neuronowych unikając problemu z propagacją gradientu dzięki zastosowaniu bloków rezydualnych, architektura ResNet101 [He i in., 2016] pozwoliła na uzyskanie większej precyzji wyników dzięki wprowadzeniu większej liczby warstw, a z kolei architektura ResNeXt101 [Xie i in., 2017] wprowadziła grupowe konwolucje, które zapewniły wzrost elastyczności i skalowalności wynikowego modelu. Architektury te zapewniają dobrą wydajność przy efektywności obliczeniowej zachowanej na należytym poziomie [He i in., 2016], [Xie i in., 2017]. 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 i in., 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 i in., 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) [Chen i in., 2019], Mask2Former [Chen i in., 2021], ConvNeXt Small [Liu i in., 2022], EfficientNet B3 [Tan, Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] oraz Swin Transformer [Liu i in., 2021]. Są to przykłady kluczowych architektur, które odegrały istotną rolę w nowoczesnej analizie obrazów i zostaną pokrótce 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., 2019]. Osiąga ona wysoką dokładność na zbiorach danych COCO [Chen i in., 2019]. 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 [Chen i in., 2019].

HTC_x101 jako szkielet wykorzystuje ResNeXt101, HTC_r101 korzysta z ResNet101, zaś w przypadku HTC_r50 zastosowaną siecią jest ResNet50. Wszystkie trzy konfiguracje jako tzw. „szyję" (*neck*) wykorzystują mechanizm Feature Pyramid Network (FPN) pozwalający na efektywne wykrywanie obiektów o różnej skali [Chen i in., 2019]. 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 modułem 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., 2019]. 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 [Chen i in., 2019].

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. [Chen i in., 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 [Chen i in., 2021].

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. Komponent Dynamic Mask Prediction pozwala zaś 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 niezrównoważonych 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 architektura ConvNeXt Small [Liu i in., 2022] 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 [Liu i in., 2022].

EfficientNet B3 [Tan, Le, 2019] to architektura również zaprojektowana w celu optymalizacji 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 [Tan, Le, 2019] został zastosowany do klasyfikacji poziomów nawodnienia kukurydzy oraz klasyfikacji porażenia kukurydzy wybranymi patogenami, co pozwoliło na uzyskanie wyników o wysokiej precyzji.

Kolejną z analizowanych architektur jest Vision Transformer (ViT) [Dosovitskiy i in., 2020], 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 niniejszej rozprawy.

Przegląd wybranych zaawansowanych architektur zamyka Swin Transformer [Liu i in., 2021] oparty na modelu Vision Transformer [Dosovitskiy i in., 2020]. 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 [Liu i in., 2021]. Wprowadza ona mechanizm przesuwających się okienek (tzw. *shifted windows*), 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 kukurydzy oraz klasyfikacji porażenia kukurydzy wybranymi patogenami, co pozwoliło na uzyskanie wyników o wysokiej precyzji.

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, w wizji komputerowej, w cyfrowej fotografii oraz w 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 spektrum kolorów jest możliwe dzięki kombinacji tychże 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 szumy 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 dla ludzkiego oka. 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 rośliny inaczej odbijają światło widzialne i bliską podczerwień, niż chore rośliny 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. Jak 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 tychże 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 krótce 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, ze 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 krawędzie obrazu. Mają one duże znaczenie w przypadku obrazów wielospektralnych, w których precyzja w usuwaniu 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 w analizie obrazów, w szczególności konwolucyjne sztuczne sieci neuronowe i ich warianty w postaci ResNet [He i in., 2016], [Xie i in., 2017] czy EfficientNet [Tan, Le, 2019] 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 niezrównoważonych 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, ze 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. Technika walidacji hold-out

Proces walidacji modeli uczenia maszynowego jest kluczowy podczas trenowania modeli, a jego celem jest ocenienie zdolności modelu do generalizacji, a dokładniej 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 zestaw treningowy, walidacyjny i testowy (tożsamy ze zbiorem walidacyjnym). 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 tychże 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 i takie podejście zastosowano również w badaniach, których dotyczy niniejsza rozprawa.

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 niekorzystne, 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 różnorodnym zbiorem 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], [Goodfellow i in., 2016].

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 zbytnio 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. Można np. monitorować wykresy train_loss(steps) i val_acc(steps), co daje możliwość śledzenia postępów treningu i wykrycie oznak overfittingu [Goodfellow i in., 2016].

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, identyfikacja nawodnienia i identyfikacja porażenia patogenami 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.

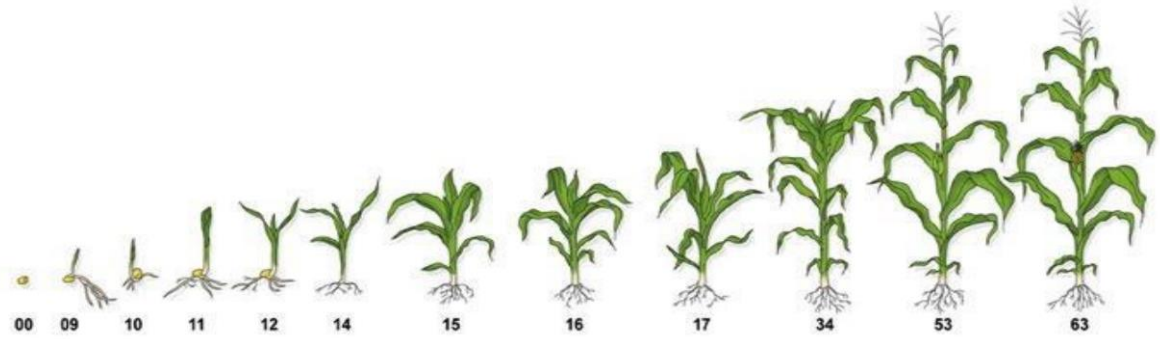
2.4.1. Skala BBCH i jej zastosowanie w analizie faz rozwojowych roślin

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 ramach przygotowywania niniejszej rozprawy 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 do liczby w pełni rozwiniętych liści, z wyłączeniem zalążków liści w postaci zwiniętej, które

znajdują się w początkowej fazie rozwoju. 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) [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 mięsistości liści, będącej wynikiem odpowiedniego turgoru. Pozwala to na klasyfikację roślin w trzy główne klasy: „nawodnienie małe", „nawodnienie średnie" oraz „nawodnienie duże". 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.

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 niezmiernie 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 i in., 2010].

W niniejszej rozprawie skupiono się na wykrywaniu objawów porażenia kukurydzy 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 Helmintosporioza objawia się charakterystycznymi przebarwieniami na liściach, które bardzo łatwo identyfikować można na obrazach RGB.

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 umożliwione 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 i in., 2010].

2.5. Przegląd podobnych rozwiązań

Określanie faz rozwojowych roślin, a w przypadku opisywanych badań - kukurydzy, jest istotne 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, a 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 reprezentacji na podstawie zadanych zbiorów obrazów [Lancashire i in., 1991]. Uczenie 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

Pierwszym z rozwiązań, 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 Mask 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 pomiarów 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, którą opublikowała Andrea Kirsten Mahlein w 2016 roku [Mahlein, 2016]. Badaczka zastosowała obrazy RGB i hiperspektralne 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 [Mahlein, 2016].

W przypadku analizy zdrowotności roślin podobne rozwiązanie zaprezentowali Chadwick H. Bock i współautorzy opracowania z 2010 roku [Bock i in., 2010]. Autorzy porównali skuteczność różnych metod analizy obrazów, począwszy od obrazów RGB po hiperspektralne, 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 i in.., 2010].

Jeszcze innym przypadkiem badania znalezionego 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

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

Kolejnym przykładem wartym uwagi jest praca Yihan Yao i współautorów [Yao i in., 2024], w której zaproponowano metodę klasyfikacji faz wzrostu kukurydzy przy użyciu cech fenotypowych i danych z BSP. Badacze połączyli indeksy wegetacyjne (VI), cechy teksturalne (TF) oraz parametry fenotypowe, takie jak zawartość chlorofilu w liściach (LCC), wskaźnik powierzchni liściowej (LAI), frakcja pokrycia wegetacyjnego (FVC) oraz wysokość baldachimu (CH). Najwyższą dokładność (95,1%) uzyskano z klasyfikatorem Random Forest przy użyciu parametrów LCC, LAI, FVC i CH. Badanie wykazało, że cechy fenotypowe przewyższają indeksy wegetacyjne, a integracja danych UAV z uczeniem maszynowym umożliwia precyzyjne monitorowanie faz rozwoju kukurydzy [Yao i in., 2024].

Warto wyróżnić również artykuł, który opublikował Avishek Bera i współautorzy [Bera i in., 2024]. Badacze zaproponowali wykorzystanie PND-Net, czyli systemu łączącego grafowe sieci konwolucyjne (GCN) z tradycyjnymi CNN do klasyfikacji niedoborów składników odżywczych i chorób roślin. Model integruje lokalne cechy obrazu liścia (Xception, ResNet-50, Inception-V3, MobileNet-V2) z relacjami przestrzennymi przechwyconymi przez GCN, wykorzystując przestrzenną piramidę (SPP) do agregacji cech wielkoskalowych. Testy na zestawach danych (banana, coffee, potato, PlantDoc) wykazały wysoką skuteczność: 90,00%, 90,54%, 96,18% i 84,30% [Bera i in., 2024].

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 pozyskiwanych z poziomu BSP, 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 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 przemyślenia 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 zobrazowań niż BSP. Chodzi tutaj o roboty polowe oraz 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 wiedzie 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 detekcji delikatnych zmian, które nie są jeszcze widoczne w spektrum światła widzialnego, a to czyni tą technikę bardzo obiecującą. Stąd też w prowadzonych badaniach starano się wypełnić tą 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 terenowych 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 zbiory danych 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 badań z użyciem uczenia głębokiego nie został jeszcze wyczerpany. Wciąż istnieje grupa algorytmów, które nie zostały zbadane pod kątem wykorzystania w zadaniach związanych z detekcją i klasyfikacją faz rozwojowych roślin, z detekcją poziomów nawodnienia oraz z detekcją porażenia wybranymi

patogenami. Jak pokazał przegląd literatury, niektóre z najpotężniejszych modeli, takich jak np. HTC i Mask2Former, nie były jeszcze badane w zadaniach tego typu.

Niniejsza praca ma na celu wypełnienie tychże luk badawczych. Istotnym celem staje się zbadanie, jaki rodzaj zobrazowania (RGB czy wielospektralne) jest bardziej odpowiedni do tego typu zastosowań oraz który z analizowanych algorytmów okaże się najskuteczniejszy. Przeprowadzenie badań pozwoli na wybór najbardziej efektywnej konfiguracji w formie: typ zobrazowania i wybrany algorytm spośród analizowanej puli algorytmów uczenia głębokiego.

3. Materiały i metody badawcze

W niniejszym rozdziale przedstawiono szczegółowe informacje dotyczące użytych danych, w tym procesu przygotowywania pól testowych, pozyskiwania danych, procesu ich oznaczania oraz zastosowanych metod badawczych. Opisano poszczególne rodzaje danych obrazowych w postaci obrazów RGB, obrazów wielospektralnych z kamery MicaSense RedEdge MX Dual oraz obrazów wielospektralnych z kamery MapIR Survey3 Red + Green + NIR, które posłużyły jako podstawa do przeprowadzenia dalszych badań i analiz. W kolejnej części tego rozdziału przybliżono architektury głębokich sieci neuronowych wykorzystanych do realizacji procesu detekcji i klasyfikacji obiektów na obrazach. Omówiono metodykę implementacji tychże architektur oraz opisano przyjęte parametry treningu i proces dostosowania 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, a także metod wykorzystywanych do prowadzenia procesu trenowania sztucznych sieci neuronowych w celu 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 zagadnień:

a) Utworzenie niezbędnych pól testowych w celu zapewnienia możliwości pobrania materiału badawczego w postaci obrazów

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 informacje 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 pozyskiwania 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 pozyskiwanie obrazów podczas okresu wegetacyjnego roślin. Drugim ważnym parametrem jest odpowiednia częstotliwość pozyskiwania obrazów, aby można było udokumentować zmieniające się parametry kondycji roślin takie jak wartość fazy rozwojowej, czy też poziom nawodnienia.

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

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

e) Dobór sprzętu obliczeniowego

Po etapie pozyskiwania materiału obrazowego z wcześniej przygotowanych pól testowych, niezmiernie ważny był dobór odpowiedniego sprzętu obliczeniowego, który posłużył do

gromadzenia i 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 dalszych przetworzeń i analiz.

g) Dobór algorytmów uczenia głębokiego

Algorytmy umożliwiające trenowanie sztucznych sieci neuronowych nie mogły być dobrane w sposób przypadkowy. Należało wybrać te algorytmy, które w dziedzinie rozpoznawania obrazów znajdują się w czołówce obecnych rozwiązań, czyli należących do 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 oraz obróbkę uzyskanych wyników w odpowiedni sposób.

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 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 będą 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 rozwiązania w rzeczywistym prototypie robota polowego.

Punktem podsumowującym metodologię badań były testy opracowanego rozwiązania w rzeczywistym prototypie robota polowego. Pozwalają one na zrewidowanie poprawności i dokładności opracowanego rozwiązania.

Poszczególne punkty przedstawionej metodologii badań zostały rozszerzone i uszczegółowione w kolejnych częściach niniejszej rozprawy.

3.2. Opis danych obrazowych

W części 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 MicaSense RedEdge MX Dual oraz danych z kamery wielospektralnej MapIR Survey3 Red + Green + NIR. 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 badawczych miały posłużyć realizacji trzech głównych zadań, 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 dotyczącego detekcji i klasyfikacji faz rozwoju kukurydzy miało miejsce na przestrzeni kilku sezonów wegetacyjnych, począwszy od 2021 roku. Poletka testowe przygotowano w szklarni oraz na otwartym polu uprawnym.

Rysunek 3.1. Widok szklarni z zewnątrz.

Rysunek 3.2. Widok poletka testowego, na którym prowadzona była uprawa kukurydzy.

Rysunek 3.3. Widok poletka testowego, na którym prowadzona była uprawa kukurydzy.

Kukurydza na poletkach testowych została posiana w odpowiednich odstępach między rzędami i między poszczególnymi roślinami, w sposób analogiczny do siewu, jaki prowadzi się na rzeczywistych polach uprawnych. Wszystko po to, aby wiernie zasymulować 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 obrazy z użyciem kamery RGB, ale zauważono, iż elementy konstrukcyjne szklarni zbytnio ingerują w wyniki działania wytrenowanych na ich podstawie modeli sztucznych sieci neuronowych. Pociągnęło to za sobą decyzję, że obrazy pozyskane z tego poletka testowego nie trafiły do żadnego z oznaczonych zbiorów danych, a badania były prowadzone wyłącznie na zewnętrznych poletkach testowych na otwartym polu, aby jeszcze lepiej zasymulować rzeczywiste warunki panujące na polach uprawnych. Na zewnętrznym poletku testowym wysiewana była kukurydza i pobierane w czasie jej wzrostu obrazy RGB i wielospektralne (z użyciem kamery wielospektralnej MicaSense RedEdge MX Dual), mające służyć do realizacji zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy. Następnie ponowiono wysiew kukurydzy i w czasie jej wzrostu pobierano obrazy RGB i wielospektralne (tym razem z użyciem kamery wielospektralnej MapIR Survey3 Red Green + NIR) mające służyć do realizacji zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy w skali BBCH 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 objawów porażenia kukurydzy. W tej sytuacji do przeprowadzenia badań posłużył zewnętrzny zbiór danych [Kaggle]. W zbiorze tym znajdują się obrazy kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami.

3.2.2. Krótkie wprowadzenie dotyczące użytych kamer

W ramach opisywanych badań posłużono się zobrazowaniem RGB i wielospektralnym. Do pozyskiwania zobrazowań RGB posłużył aparat cyfrowy, zaś do pozyskiwania zobrazowań wielospektralnych posłużyła kamera wielospektralna MicaSense RedEdge MX Dual.

W przypadku kolejnego wysiewu kukurydzy do pozyskiwania zobrazowań RGB również posłużył aparat cyfrowy, zaś do pozyskiwania zobrazowań wielospektralnych posłużyła kamera wielospektralna MapIR Survey3 Red + Green + NIR. Szczegółowe parametry techniczne użytych kamer wielospektralnych znajdują się w odpowiednich odnośnikach bibliograficznych [MicaSense, 2024a], [MicaSense, 2024b], [MapIR, 2024].

W przypadku kamery RGB kalibracja prowadzona była w sposób automatyczny, zaś w przypadku obu kamer wielospektralnych MicaSense RedEdge MX Dual oraz MapIR Survey3 Red + GGreen + NIR zastosowano standardową kalibrację z użyciem dedykowanych paneli kalibracyjnych. Poniżej zamieszczono rysunki pokazujące wygląd poszczególnych kamer wielospektralnych.

Rysunek 3.4. Wygląd kamery wielospektralnej MicaSense RedEdge MX Dual [MicaSense, 2024b].

Rysunek 3.5. Wygląd kamery wielospektralnej MapIR Survey3 Red + GGreen + NIR [MapIR, 2024].

3.2.3. Dane RGB

Na przygotowanych 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 zalążka pierwszego liścia w postaci zwiniętej struktury. Pozyskiwanie obrazów odbywało się codziennie, aż do momentu wykształcenia maksymalnych rozmiarów roślin. Zagwarantowało to pozyskanie danych z każdego etapu rozwoju poszczególnych roślin (szczególnie z faz BBCH 10-19).

Obrazy RGB pozyskiwane były o różnych porach dnia oraz przy różnych warunkach pogodowych, aby zwielokrotnić różnorodność zbioru treningowego. 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 nie trafiły zatem do żadnego z oznaczonych zbiorów danych. 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 skierowanej prostopadle do rzędu kukurydzy.

Poniżej przedstawiono przykłady pozyskanych obrazów kukurydzy pokazujące rośliny w różnych fazach rozwoju. Na rysunku 3.6. przedstawiono obrazy kukurydzy w jej początkowych fazach rozwoju, a na rysunku 3.7. widoczne są obrazy kukurydzy w jej wyższych fazach rozwoju, przy czym po lewej stronie widać kukurydzę pozbawioną chwastów, a po

prawej kukurydzę z wyrastającymi pomiędzy nią chwastami. Rysunek 3.8. przedstawia zaś obraz kukurydzy wykonany wieczorem przy mniejszej dostępności światła.

Rysunek 3.6. Obrazy kukurydzy będącej w jej początkowych fazach rozwojowych.

Rysunek 3.7. Obrazy kukurydzy niezachwaszczonej i zachwaszczonej będącej w jej wyższych fazach rozwojowych.

Rysunek 3.8. Obraz kukurydzy wykonany wieczorem.

Przy wykorzystaniu obrazów RGB kukurydzy pozyskanych w ramach wysiewu w początkowych sezonach wegetacyjnych powstał autorski zbiór danych zawierający obrazy kukurydzy w różnych fazach rozwojowych, zaś w ramach wysiewu w późniejszym sezonie wegetacyjnym powstały dwa autorskie zbiory danych: jeden dotyczący faz rozwojowych kukurydzy oraz drugi - dotyczący poziomów nawodnienia kukurydzy. Posłużyły one w kolejnych krokach do trenowania zestawu głębokich sztucznych sieci neuronowych.

Prowadzone badania ewoluowały na przestrzeni realizacji doktoratu i konieczne było stworzenie dwóch zbiorów danych zawierających obrazy RGB kukurydzy będącej w różnych fazach rozwojowych. Pierwszy z nich był referencyjny dla zbioru danych zawierającego obrazy wielospektralne kukurydzy pozyskiwane przy użyciu kamery MicaSense RedEdge MX Dual i przedstawiał te same rośliny, których zobrazowania wielospektralne pozyskiwano w tym samym czasie. Drugi zbiór danych zawierający obrazy RGB kukurydzy będącej w różnych fazach rozwojowych pozyskiwany w późniejszym sezonie wegetacyjnym był referencyjny dla zbioru danych zawierającego obrazy wielospektralne kukurydzy pozyskiwane przy użyciu kamery wielospektralnej MapIR Survey3 Red + Green + NIR (oba zbiory: RGB i wielospektralny dotyczą tych samych roślin i były pozyskiwane w tym samym czasie). Trzeci zbiór danych zawierający obrazy RGB kukurydzy charakteryzującej się różnymi poziomami nawodnienia pozyskiwane w tym samym późniejszym sezonie wegetacyjnym, co poprzedni zbiór, był referencyjny dla zbioru danych zawierającego obrazy wielospektralne pozyskiwane 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, a także pozyskiwanie obu typów zobrazowań: RGB i wielospektralnych w tym samym czasie, gwarantowało zachowanie tych samych warunków prowadzenia badań dla obu eksperymentów. Spełnienie tego warunku było kluczowe w momencie porównywania uzyskanych wyników i uzależniania ich wyłącznie od rodzaju zastosowanego zobrazowania: RGB oraz wielospektralnego.

Podsumowując, w wyniku tworzenia autorskich zbiorów danych RGB powstały następujące zbiory: zbiór obrazów RGB z początkowych sezonów wegetacyjnych odnoszący się do faz rozwoju kukurydzy, zbiór obrazów RGB z późniejszego sezonu wegetacyjnego również dotyczący faz rozwoju kukurydzy oraz zbiór obrazów RGB z tego samego późniejszego sezonu wegetacyjnego, odnoszący się do różnych poziomów nawodnienia kukurydzy.

Etapy przygotowywania poszczególnych zbiorów danych były następujące:

- 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,
- wybór narzędzia do oznaczania obrazów,
- dobór optymalnego sposobu oznaczania obiektów na obrazach oraz przeprowadzenie procesu oznaczania obrazów,
- przygotowanie plików wyjściowych po zakończeniu procesu oznaczania:
    - w przypadku oznaczania faz rozwojowych zastosowana była metoda wieloboków tzw. „polygon", a wyjściowe pliki zapisywane były w formacie COCO .JSON,
    - w przypadku oznaczania poziomów nawodnienia oraz rodzajów porażenia kukurydzy zastosowana była metoda „ground truth", a obrazy zostały przyporządkowane do odpowiednich folderów oznaczających poszczególne klasy obiektów.

Poniżej zamieszczono przykłady obrazów RGB dla kukurydzy o małym poziomie nawodnienia oraz dla kukurydzy o dużym poziomie nawodnienia.

Rysunek 3.9. Obrazy kukurydzy charakteryzującej się małym (po lewej) oraz dużym poziomem nawodnienia (po prawej).

Podczas przygotowań do realizacji zadania związanego z detekcją i klasyfikacją kukurydzy porażonej wybranymi patogenami, 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. Pomocny stał się zewnętrzny zbiór danych [Kaggle, 2022].

W zbiorze tym 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

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

3.2.4. Dane wielospektralne z kamery MicaSense RedEdge MX DUAL

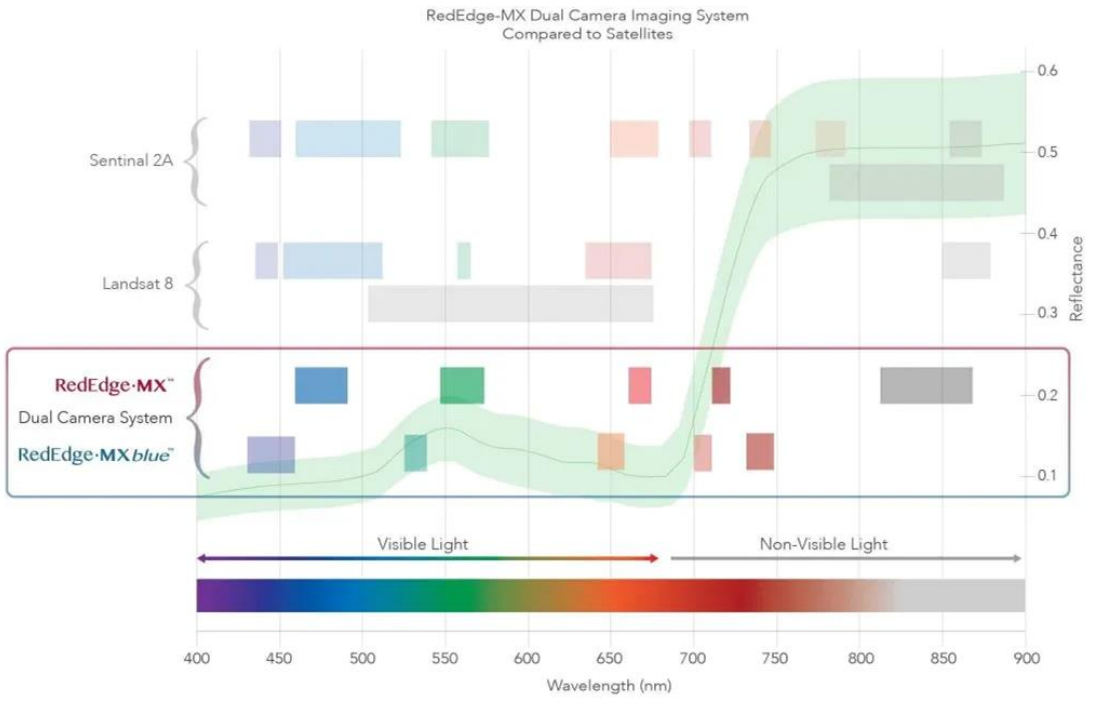
Obrazy wielospektralne z kamery MicaSense RedEdge MX DUAL pozyskiwane były na przygotowanych do tego poletkach testowych, tożsamych z poletkami, z których pozyskiwane były obrazy RGB, a sam proces ich pobierania był realizowany z identyczną częstotliwością, co proces pobierania obrazów RGB. Również w tym przypadku zapewniono maksymalną różnorodność obrazów pod względem ilości roślin, obecności wśród nich chwastów lub ich braku oraz różnej pory dnia i różnych warunków pogodowych.

Kamera wielospektralna MicaSense RedEdge MX DUAL może pobierać zobrazowania w 10 kanałach spektralnych charakteryzujących się następującymi parametrami [MicaSense, 2024a]:

- 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

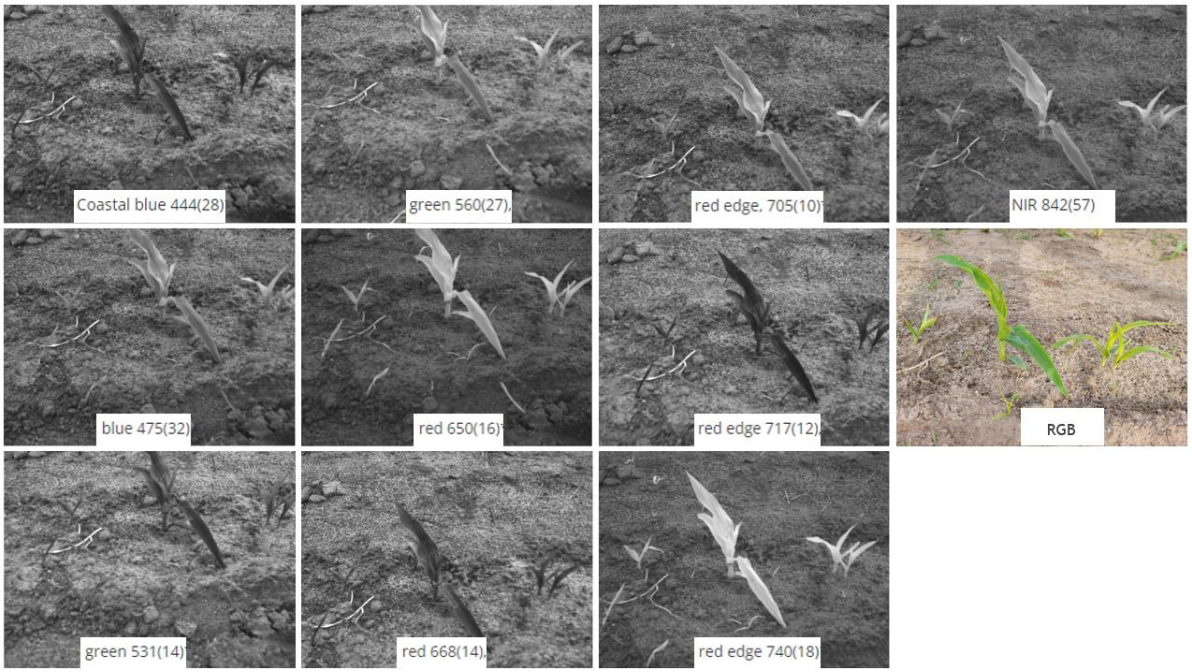
- 4 kanał spektralny – pasmo Near Infrared (bliska podczerwień), środek długości fali 842, zakres pasma 57
- 5 kanał spektralny – pasmo RedEdge („czerwona krawędź” 1), środek długości fali 717, zakres pasma 12
- 6 kanał spektralny – pasmo Coastal Blue (niebieski morski), środek długości fali 444, zakres 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 RedEdge („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

Na Rysunku 3.10 przedstawiono pasma poszczególnych kanałów kamery MicaSense RedEdge MX DUAL [MicaSense, 2024a].



Rysunek 3.10. Pasma poszczególnych kanałów kamery MicaSense RedEdge MX DUAL [MicaSense, 2024a].

Na rysunku 3.11 zamieszczono zaś przykłady obrazów wielospektralnych pozyskanych dla tej samej rośliny oraz zestawiony z nimi dla porównania obraz RGB. Obrazy z kamery MicaSense RedEdge-MX DUAL są generowane w formacie TIFF.



Rysunek 3.11. Obrazy kukurydzy zarejestrowane przez poszczególne obiektywy kamery wielospektralnej MicaSense RedEdge-MX DUAL oraz obraz zarejestrowany przez kamerę RGB.

Pozyskanie obrazów było punktem wyjścia do realizacji kolejnych zadań, w tym do przygotowania zbiorów danych z oznaczonymi obrazami, które w kolejnych krokach posłużyły do trenowania zestawu głębokich sztucznych sieci neuronowych. W przypadku obrazów wielospektralnych z kamery MicaSense RedEdge-MX DUAL zadaniem, do którego realizacji posłużyły te zobrazowania była detekcja i klasyfikacja faz rozwoju kukurydzy. Poszczególne etapy przygotowywania zbioru danych 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,
- wybór narzędzia do oznaczania obrazów,
- dobór optymalnego sposobu oznaczania obiektów na obrazach oraz przeprowadzenie procesu oznaczania obrazów,

- przygotowanie pliku wyjściowego po zakończeniu procesu oznaczania – podczas oznaczania zastosowana była metoda wieloboków tzw. „polygon”, a wyjściowy plik został zapisany w formacie COCO .JSON.

W podrozdziale *3.2.6. „Proces oznaczania obrazów”* zostanie przybliżony również proces oznaczania obrazów, w tym obrazów wielospektralnych z kamery MicaSense RedEdge MX-Dual oraz charakterystyka powstałych zbiorów danych pod względem liczności obrazów i obiektów należących do poszczególnych klas.

3.2.5. Dane wielospektralne z kamery MapIR Survey3 Red + Green + NIR

Obrazy wielospektralne z kamery MapIR Survey3 Red + Green + NIR pozyskiwane były w tym samym sezonie wegetacyjnym oraz na tych samych poletkach testowych, z których pozyskiwane były obrazy RGB, a sam proces ich pobierania był realizowany z identyczną częstotliwością, co proces pobierania obrazów 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 i różnych warunków pogodowych.

Na rysunku 3.12 zamieszczono przykłady obrazów wielospektralnych pozyskiwanych za pomocą kamery wielospektralnej MapIR Survey3 Red + Green + NIR. Obrazy te są generowane w formacie JPG.

Rysunek 3.12. Obrazy pozyskane za pomocą kamery MapIR Survey3 Red + Green + NIR.

Po pozyskaniu obrazów kukurydzy kolejnym krokiem w realizacji zamierzonych zadań było przygotowanie oznaczonych zbiorów danych, które w kolejnych krokach posłużyły do trenowania zestawu głębokich sztucznych sieci neuronowych. W przypadku obrazów wielospketralnych z kamery MapIR Survey3 Red + Green + NIR 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 zbiorów danych 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,
- wybór narzędzia do oznaczania obrazów,
- dobór optymalnego sposobu oznaczania obiektów na obrazach oraz przeprowadzenie procesu oznaczania,
- przygotowanie plików wyjściowych po zakończeniu procesu oznaczania:
    - w przypadku oznaczania faz rozwojowych zastosowana była metoda wieloboków tzw. „polygon", a wyjściowy plik został zapisany w formacie COCO .JSON,
    - w przypadku oznaczania poziomów nawodnienia zastosowana była metoda „ground truth", a obrazy zostały przyporządkowane do odpowiednich folderów oznaczających poszczególne klasy obiektów.

W podrozdziale *3.2.6. „Proces oznaczania obrazów"* zostanie przybliżony proces oznaczania obrazów, w tym obrazów wielospektralnych z kamery MapIR Survey3 Red + Green + NIR 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 koniecznych do wykonania przed realizacją procesu trenowania sztucznych sieci neuronowych i tworzenia na tej podstawie modeli zdolnych do realizowania określonych zadań. Mowa tutaj zarówno o obrazach RGB jak i obrazach wielospektralnych z kamery MicaSense RedEdge MX-Dual oraz z kamery MapIR Survey3 Red + Green + NIR.

Pierwszym punktem w procesie przygotowywania obrazów do procesu ich oznaczania był przegląd zebranych obrazów oraz odrzucenie wszelkich wadliwych przypadków, np. nie zawierających wymaganych obiektów, prześwietlonych, rozmytych, zbytnio przysłoniętych przez elementy nie będące kukurydzą itp.

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

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 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: „*mały”*, „*średni”* i „*duży”*.

W przypadku zadania związanego z detekcją i klasyfikacją porażenia kukurydzy patogenami przyjęto następujące klasy zgodnie z którymi oznaczono poszczególne obrazy: *zdrowa roślina, helmintosporioza, gąsienica spodoptera frugiperda, rdza pospolita* oraz *szara plamistość*.

Po wyborze klas dla poszczególnych eksperymentów kolejnym etapem był wybór metody oznaczania obiektów. W przypadku zadań związanych z detekcją i klasyfikacją poziomów nawodnienia kukurydzy oraz detekcją i klasyfikacją porażenia kukurydzy wybranymi patogenami zastosowano metodę oznaczania *ground truth*. Zgodnie z tą metodą poszczególne zbiory obrazów podzielono na foldery, w których zostały umieszczone obrazy przypisane poszczególnym klasom. Dodatkowo, obok podziału zbiorów na foldery oznaczające poszczególne klasy obiektów, dokonano podziału na foldery oznaczające zbiory treningowe i walidacyjne (testowe). Najlepiej wyjaśnią to zamieszczone poniżej struktury folderów odpowiadające poszczególnym zadaniom.

Dla zadania związanego z detekcją i klasyfikacją poziomów nawodnienia kukurydzy struktura folderów była następująca:

- folder ***train*** - zbiór treningowy
    - folder ***mały*** – zbiór treningowy obrazów kukurydzy charakteryzującej się małym poziomem nawodnienia
    - folder ***średni*** – zbiór treningowy obrazów kukurydzy charakteryzującej się średnim poziomem nawodnienia
    - folder ***duży*** – zbiór treningowy obrazów kukurydzy charakteryzującej się dużym poziomem nawodnienia
- folder ***val*** - zbiór walidacyjny (testowy)
    - ***mały*** – zbiór walidacyjny (testowy) obrazów kukurydzy charakteryzującej się małym poziomem nawodnienia
    - ***średni*** – zbiór walidacyjny (testowy) obrazów kukurydzy charakteryzującej się średnim poziomem nawodnienia
    - ***duży*** – zbiór walidacyjny (testowy) obrazów kukurydzy charakteryzującej się dużym poziomem nawodnienia.

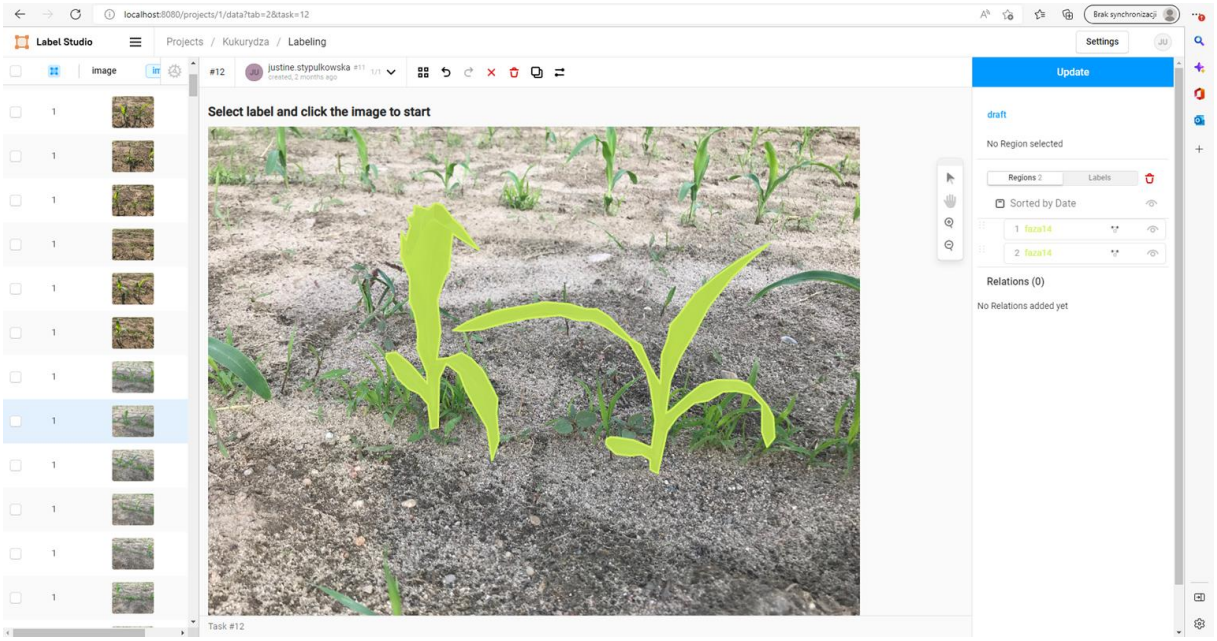
Dla zadania związanego z detekcją i klasyfikacją porażenia kukurydzy wybranymi patogenami struktura folderów była zaś następująca:

- folder ***train*** - zbiór treningowy
    - folder ***zdrowa roślina*** – zbiór treningowy obrazów zdrowej kukurydzy
    - folder ***helmintosporioza*** – zbiór treningowy obrazów kukurydzy charakteryzującej się objawami porażenia Helmintosporiozą
    - folder ***gąsienica spodoptera frugiperda*** – zbiór treningowy obrazów kukurydzy charakteryzującej się objawami porażenia gąsienicą Spodoptera frugiperda
    - folder ***rdza pospolita*** – zbiór treningowy obrazów kukurydzy charakteryzującej się objawami porażenia Rdzą pospolitą
    - folder ***szara plamistość*** – zbiór treningowy obrazów kukurydzy charakteryzującej się objawami porażenia Szarą plamistością
- folder ***val*** - zbiór walidacyjny (testowy)
    - folder ***zdrowa roślina*** – zbiór walidacyjny (testowy) obrazów zdrowej kukurydzy
    - folder ***helmintosporioza*** – zbiór walidacyjny (testowy) obrazów kukurydzy charakteryzującej się objawami porażenia Helmintosporiozą

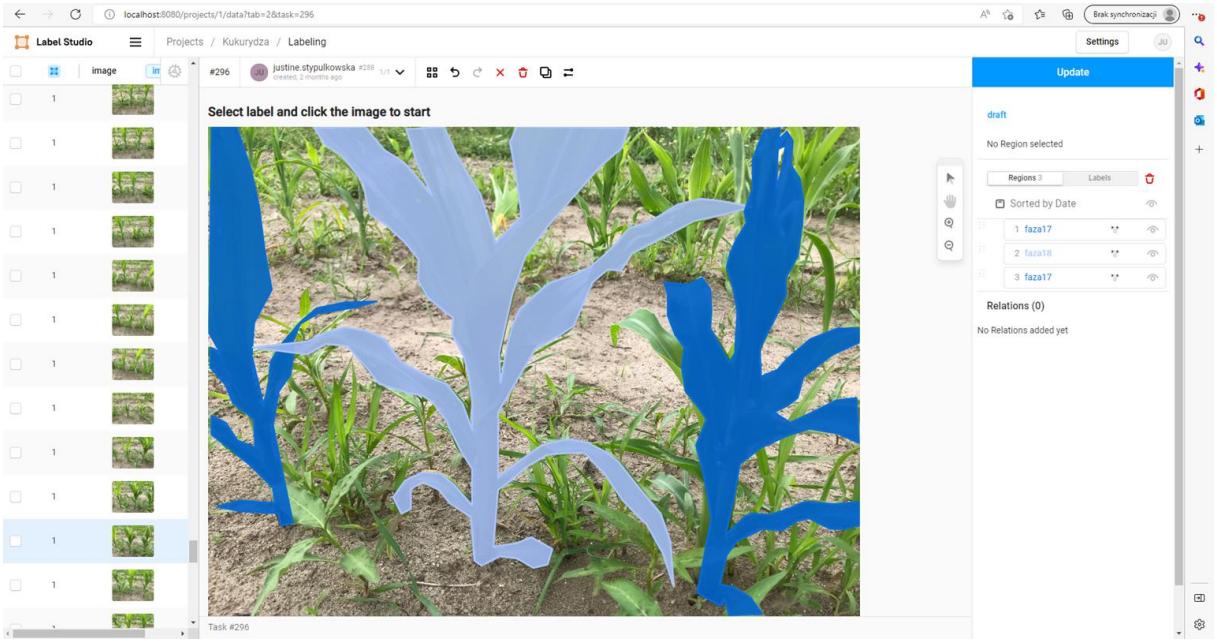
- folder ***gąsienica spodoptera frugiperda*** – zbiór walidacyjny (testowy) obrazów kukurydzy charakteryzującej się objawami porażenia gąsienicą Spodoptera frugiperda
- folder ***rdza pospolita*** – zbiór walidacyjny (testowy) obrazów kukurydzy charakteryzującej się objawami porażenia Rdzą pospolitą
- folder ***szara plamistość*** – zbiór walidacyjny (testowy) obrazów kukurydzy charakteryzującej się objawami porażenia Szarą plamistością.

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. Na podstawie testów najbardziej optymalnym rozwiązaniem okazało się środowisko Label Studio [Label Studio, 2024], które pozwala na wybór różnych technik oznaczania oraz umożliwia tworzenie różnych typów plików wyjściowych.

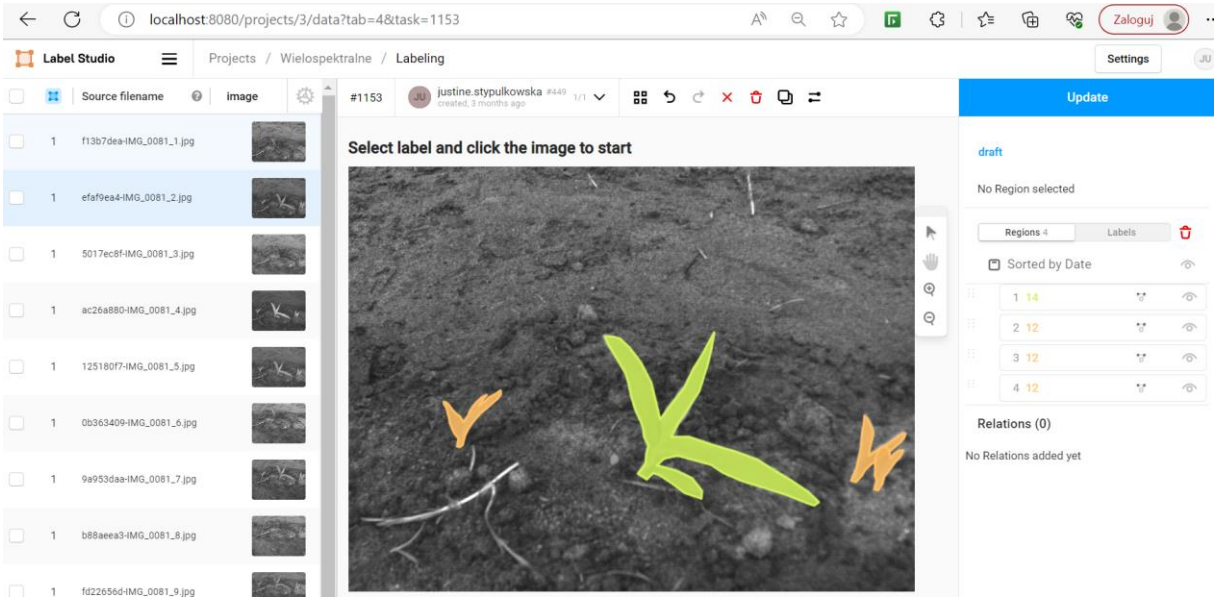
Po dokonaniu wyboru narzędzia do oznaczania kolejnym etapem był proces dostosowywania środowiska do realizowanego procesu oznaczania obrazów. W tym celu ustawiono w Label Studio następujące klasy oznaczanych obiektów: *faza10, faza11, faza12, faza13, faza14, faza15, faza16, faza17, faza18* oraz *faza19* 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 za pomocą wielokątów, tzw. *polygon,* co jest metodą wystarczająco precyzyjną. W kolejnym etapie rozpoczęto właściwy proces oznaczania obiektów na obrazach. Co warto zaznaczyć, dla poszczególnych zbiorów obrazów: RGB, obrazów wielospektralnych z kamery MicaSense RedEdge MX-Dual oraz obrazów wielospektralnych z kamery MapIR Survey3 Red + Green + NIR stworzono w Label Studio osobne projekty, a poszczególne obrazy były oznaczane osobno w ramach 3 projektów, co gwarantowało możliwość wygenerowania 3 osobnych plików wyjściowych po zakończeniu procesu oznaczania. Tworzenie tychże zbiorów obrazów z oznaczonymi obiektami było od siebie niezależne. Na rysunkach poniżej przedstawiono zrzuty ekranu z widoku narzędzia Label Studio podczas dokonywania procesu oznaczania.



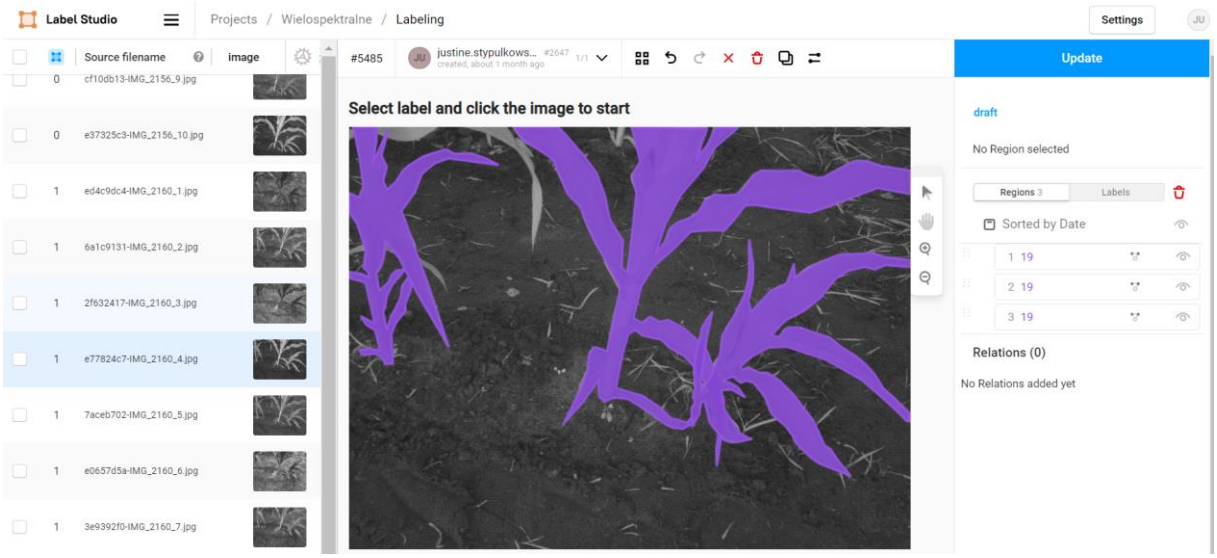
Rysunek 3.13. Widok środowiska Label Studio podczas procesu oznaczania obiektów w ramach projektu związanego ze zbiorem obrazów RGB. Na ekranie widać rośliny będące w fazie 14 oznaczone metodą wielokąta (ang. „polygon").



Rysunek 3.14. Widok środowiska Label Studio podczas procesu oznaczania obiektów w ramach projektu związanego ze zbiorem obrazów RGB. Na ekranie widać rośliny będące w fazie 17 i 18 oznaczone metodą wielokąta (ang. „polygon").



Rysunek 3.15. Widok środowiska Label Studio podczas procesu oznaczania obiektów w ramach projektu związanego ze zbiorem obrazów wielospektralnych z kamery MicaSense RedEdge MX-Dual. Na ekranie widać rośliny będące w fazie 12 i 14 oznaczone metodą wielokąta (ang. „polygon").



Rysunek 3.16. Widok środowiska Label Studio podczas procesu oznaczania obiektów w ramach projektu związanego ze zbiorem obrazów wielospektralnych z kamery MicaSense RedEdge MX-Dual. Na ekranie widać rośliny będące w fazie 19 oznaczone metodą wielokąta (ang. „polygon").

W zbiorach obrazów RGB pobieranych w początkowych sezonach wegetacyjnych znajduje się odpowiednio: 396 oznaczonych obrazów kukurydzy w mniejszym zbiorze oraz 700 obrazów w powiększonym zbiorze. Z kolei w zbiorze obrazów wielospektralnych z kamery MicaSense RedEdge MX-Dual pobieranych w tych samych sezonach wegetacyjnych 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.

Na każdym obrazie zostało zarejestrowanych od 1 do kilku roślin kukurydzy, co przełożyło się na łączną ilość 884 oznaczonych obiektów w mniejszym zbiorze obrazów RGB z początkowych sezonów wegetacyjnych, 1645 oznaczonych obiektów w powiększonym zbiorze obrazów RGB z początkowych sezonów wegetacyjnych, łączną ilość 9070 oznaczonych obiektów w zbiorze obrazów wielospektralnych z kamery MicaSense RedEdge MX-Dual (czyli 907 na każdy kanał spektralny) z początkowych sezonów wegetacyjnych, łączną ilość 1000 oznaczonych obiektów w zbiorze obrazów RGB z późniejszego sezonu wegetacyjnego oraz łączną ilość 1000 obiektów w zbiorze obrazów z kamery wielospektralnej MapIR Survey3 Red + Green + NIR z późniejszego sezonu wegetacyjnego. Dodatkowym celem było to, aby ilość obiektów danej klasy, czyli ilość roślin będących w danej fazie rozwojowej, była zrównoważona w każdym ze zbiorów. Osiągnięta została następującą liczność obiektów w poszczególnych klasach:

- w mniejszym zbiorze obrazów RGB pobieranych w początkowych sezonach wegetacyjnych, stanowiącym zbiór referencyjny dla zbioru obrazów wielospektralnych z kamery MicaSense RedEdge MX-Dual (również pobieranych w początkowych sezonach wegetacyjnych):

faza 10: 83 obiekty, faza 11: 83 obiekty, faza 12: 137 obiektów, faza 13: 83 obiekty, faza 14: 83 obiekty, faza 15: 83 obiekty, faza 16: 83 obiekty, faza 17: 83 obiekty, faza 18: 83 obiekty, faza 19: 83 obiekty,

- w powiększonym zbiorze obrazów RGB pobieranych w początkowych sezonach wegetacyjnych:

faza 10: 83 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 MicaSense RedEdge MX-Dual pobieranych w początkowych sezonach wegetacyjnych, dla każdego pojedynczego kanału spektralnego:

faza 10: 82 obiekty, faza 11: 86 obiektów, faza 12: 162 obiekty, faza 13: 82 obiekty, faza 14: 82 obiekty, faza 15: 83 obiekty, faza 16: 84 obiekty, faza 17: 83 obiekty, faza 18: 83 obiekty, faza 19: 80 obiektów

- w zbiorze obrazów RGB pobieranych w późniejszym sezonie wegetacyjnym:

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,

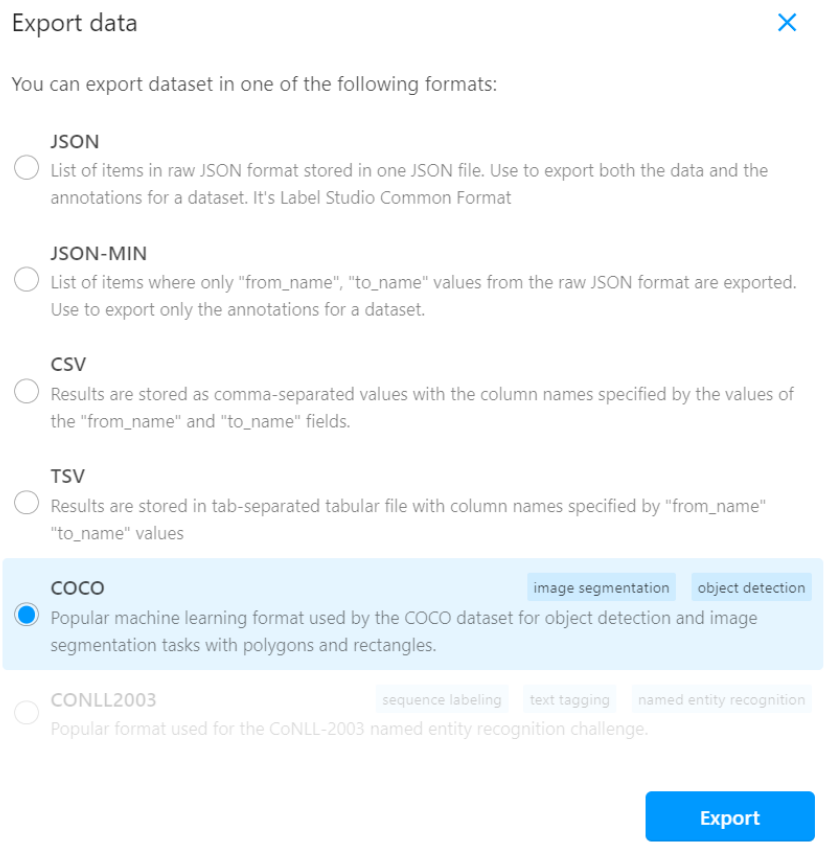
- w zbiorze obrazów wielospektralnych z kamery MapIR Survey3 Red + Green + NIR pobieranych w późniejszym sezonie wegetacyjnym:

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.

Co ważne - zbiór obrazów RGB pobieranych w początkowych sezonach wegetacyjnych przygotowano w dwóch wersjach: w wersji mniejszej - referencyjnej względem zbioru obrazów wielospektralnych z kamery MicaSense RedEdge MX-Dual również pobieranych w początkowych sezonach wegetacyjnych oraz w wersji powiększonej, czyli o większej liczbie obrazów i o większej liczbie oznaczonych obiektów.

Stworzenie referencyjnych zbiorów obrazów RGB i wielospektralnych było podstawą do porównania wyników uzyskanych przez modele głębokich sieci neuronowych wytrenowanych na tychże zbiorach. Z kolei większy zbiór RGB posłużył do porównania wyników osiąganych przez modele trenowane na tym zbiorze z wynikami modeli trenowanych na mniejszym zbiorze. RGB. Dzięki temu możliwe było sprawdzenie postępów w uczeniu poszczególnych modeli oraz wybór najlepszego z uzyskanych rozwiązań.

Po zakończeniu procesu oznaczania obrazów należało przejść do kolejnego etapu, czyli przygotowania tzw. *outputów*, czyli plików wyjściowych z wynikami wszystkich przeprowadzonych oznaczeń. Pliki te są niezbędne do tego, aby przełożyć logikę oznaczania przeprowadzoną przez człowieka w aplikacji komputerowej na język zrozumiały przez poszczególne algorytmy uczenia maszynowego. Środowisko Label Studio oferuje kilka typów formatów plików wyjściowych (jak pokazano na rysunku 3.17), 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 oznaczania.



Rysunek 3.17. Widok panelu z wyborem formatu pliku wyjściowego w środowisku Label Studio.

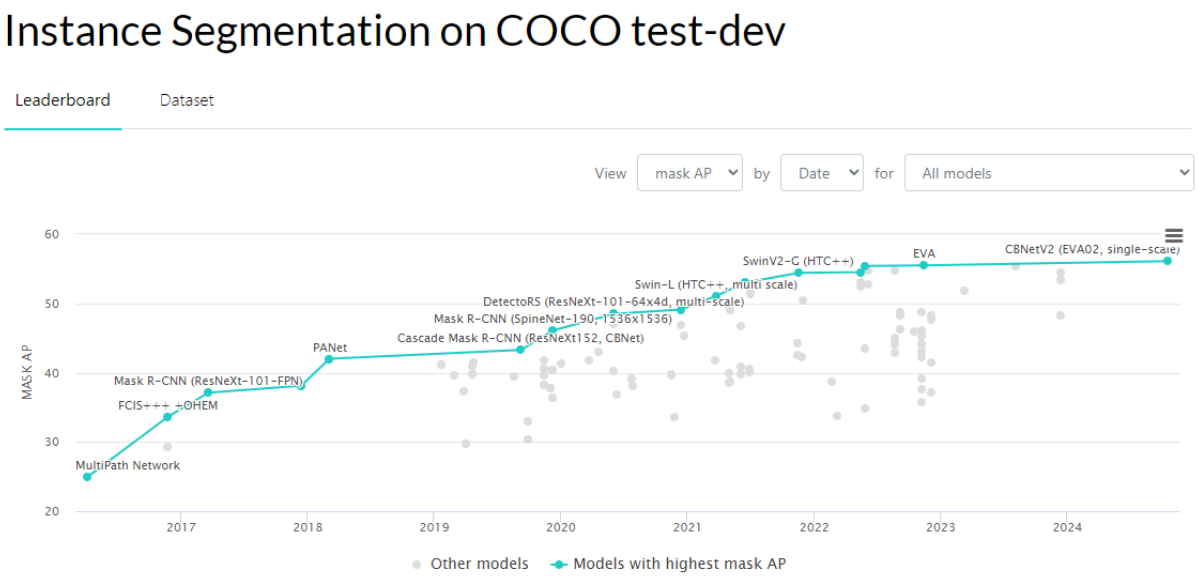
Po wygenerowaniu plików wyjściowych w formacie COCO dane w poszczególnych zbiorach odnoszących się do konkretnych zadań zostały podzielone na zbiory treningowe, które stanowiły około 70% poszczególnych zbiorów oraz zbiory walidacyjne (tożsame ze zbiorami

testowymi) stanowiące około 30% poszczególnych zbiorów. Podział został dokonany w sposób losowy, a stratyfikacja zapewniła równomierny podział obiektów z każdej fazy BBCH w każdym z tychże zbiorów.

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 reprezentacji 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 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 kukurydzy porażonej wybranymi patogenami. Szczególną uwagę poświęcono modelom HTC (Hybrid Task Cascade) [Chen i in., 2019], 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. Poniżej umieszczono zestawienie najskuteczniejszych architektur służących do przeprowadzania zadania semantycznej segmentacji na zbiorze COCO. Widać tutaj m.in. wysoką pozycję wyników algorytmu HTC wykorzystywanego podczas realizacji badań i przygotowywania rozprawy.



Rysunek 3.18. 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 szkielet sieci, czyli tzw. *backbone*), HTC_r101 (z ResNet101 jako szkielet sieci) i HTC_r50 (z ResNet50 jako szkielet sieci). 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 we frameworku MMDetection [MMDetection, 2020], [Chen i in., 2019]. Kluczowym elementem implementacji było dostosowanie szkieletów sieci, czyli tzw. 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].

Szkielet 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], [Chen i in., 2019].

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, 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 [Chen i in., 2021].

Implementacja Mask2Former została zrealizowana z wykorzystaniem ResNet50 jako szkieletu sieci, 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 [He i in., 2016].

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 obejmuje także modyfikacje liczby zapytań w mechanizmie Transformera, co pozwala 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 klasyfikacją porażenia kukurydzy wybranymi patogenami, w niniejszej pracy zaimplementowano i przetestowano pięć zaawansowanych architektur sieci neuronowych: 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]. Każdy z tych modeli został dostosowany do specyficznych wymagań badania, uwzględniając charakterystykę danych obrazowych oraz cele analizy.

ConvNeXt Small jest nowoczesną architekturą, która łączy tradycyjne podejście do konwolucyjnych sieci neuronowych z nowymi technikami przetwarzania obrazu, inspirowanymi architekturami Transformerów. 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 na dwa sposoby: do klasyfikacji trzech klas związanych z poziomami nawodnienia kukurydzy oraz do klasyfikacji pięciu klas związanych z porażeniem kukurydzy wybranymi patogenami (w tym kukurydzy zdrowej). Kluczowe znaczenie miały techniki augmentacji, takie jak rotacje, odbicia lustrzane, zmiany jasności oraz kontrastu. Techniki te pozwoliły na poprawę zdolności generalizacyjnych modelu, co było kluczowe dla uzyskania stabilnych i precyzyjnych wyników w zróżnicowanych warunkach środowiskowych [Liu i in., 2022].

ConvNeXt Small wykorzystuje zaawansowane operacje konwolucyjne, które są odpowiednio skalowane, aby maksymalnie wykorzystać dostępne zasoby obliczeniowe. Model ten został wyposażony w 384 jednostki konwolucyjne przypadające 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].

Kolejna architektura to ResNet50, który 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 oznak porażenia roślin wybranymi patogenami.

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 technik augmentacji danych, które miały na celu zwiększenie zdolności generalizacyjnych oraz stabilności modelu [He i in., 2016]. Model ten charakteryzuje się wysoką wydajnością obliczeniową, co pozwala na przetwarzanie dużych ilości danych obrazowych w stosunkowo krótkim czasie. W niniejszej pracy szczególną uwagę zwrócono również na optymalizację procesu trenowania modelu, obejmującą dobór odpowiednich hiperparametrów, takich jak współczynnik uczenia, momentum oraz wartość regularyzacji wagi [He i in., 2016].

EfficientNet-B3 to z kolei 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 oznak porażenia roślin wybranymi patogenami. Implementacja 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 realizowanych zadań, w tym z klasyfikacją trzech klas w przypadku zadania związanego z poziomami nawodnienia oraz z klasyfikacją pięciu klas w przypadku zadania związanego z porażeniem kukurydzy wybranymi patogenami. Zastosowane zostały również techniki augmentacji, takie jak skalowanie, obracanie, zmiana jasności i kontrastu [Tan, Le, 2019]. Dzięki zastosowaniu architektury EfficientNet-B3 możliwe było uzyskanie wysokiej precyzji, 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].

Kolejną architekturą jest Vision Transformer (ViT), czyli innowacyjna architektura, która przenosi mechanizmy Transformerów 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 oznak porażenia kukurydzy wybranymi patogenami. Model ten charakteryzuje się nowatorskim podejściem do analizy obrazów, gdzie obrazy są dzielone na mniejsze fragmenty (tzw. *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 związanymi z poziomami nawodnienia kukurydzy oraz do pracy z pięcioma klasami związanymi z porażeniem kukurydzy wybranymi patogenami (w tym ze zdrową kukurydzą). Zastosowano również techniki augmentacji danych, takie jak skalowanie, rotacja, zmiana jasności i kontrastu, co miało na celu zwiększenie zdolności generalizacyjnych oraz stabilności modelu. Dzięki zastosowaniu mechanizmów Transformerów, 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].

Kolejną wykorzystywaną architekturą jest Swin Transformer, który jest oparty na architekturze Vision Transformer. Wprowadza on mechanizm przesuwających się okienek (tzw. *shifted windows*), co pozwala na efektywne przetwarzanie obrazów o dużej rozdzielczości [Liu i in., 2021]. Implementacja Swin Transformer obejmowała konfigurację modelu do pracy z trzema klasami związanymi z poziomami nawodnienia kukurydzy oraz do pracy z pięcioma klasami związanymi z porażeniem kukurydzy wybranymi patogenami (w tym ze zdrową kukurydzą). Zastosowane zostały również zaawansowane techniki augmentacji danych, takie jak losowe rotacje, odbicia lustrzane, zmiany jasności oraz kontrastu [Liu i in., 2021]. Model ten został zoptymalizowany pod kątem efektywnego wykorzystania zasobów obliczeniowych, co jest istotne w kontekście analizy dużych zbiorów danych obrazowych [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.

3.4. Proces trenowania modeli

W ramach przygotowywania niniejszej pracy wykorzystano i odpowiednio dostosowano różnorodne architektury głębokich sieci neuronowych, takie jak: Hybrid Task Cascade (HTC) [Chen i in., 2019], 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]. Celem procesu trenowania było uzyskanie stabilnych i dokładnych wyników, co wymagało starannej optymalizacji parametrów, takich jak: liczba epok, wielkość porcji danych (tzw. *batch*), funkcja kosztu, dobór

algorytmu optymalizacji, a także modyfikacje architektury modeli. W kolejnych podrozdziałach omówiono szczegóły techniczne tego procesu.

3.4.1. Trenowanie modeli HTC i Mask2Former

Trenowanie modeli HTC i Mask2Former odbyło się w środowisku opartym na bibliotece PyTorch, z wykorzystaniem funkcji i modułów dostępnych we frameworku MMDetection [Chen i in., 2019], [Chen i in., 2021], [Ge i in., 2021], [MMDetection, 2020]. Każdy model został przystosowany do specyficznych zadań związanych z detekcją obrazów kukurydzy.

Proces trenowania rozpoczęto od podziału dostępnych danych na zbiory treningowe i walidacyjne. Dane wejściowe były poddawane 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, momentum równym 0,9 oraz regularyzacją wagi na poziomie 0,0001 [Kingma & Ba, 2015]. Zastosowany harmonogram uczenia oparty na krokach (tzw. *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, accuracy, precision, recall, F1-score 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-B3, Vision Transformer, Swin Transformer

Modele 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] były trenowane w środowisku opartym na bibliotece *timm*, z zastosowaniem optymalizatora Adam, z początkową wartością współczynnika uczenia wynoszącą 0,00001 [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 tychże 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 [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] zastosowano różne metryki ewaluacyjne, takie jak accuracy, precision, recall oraz F1-score. 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].

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 dobrej 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 umożliwiła 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 z wykorzystaniem wytrenowanych modeli można analizować na komputerze stacjonarnym tak, jakby były wykonywane na rzeczywistym robocie podczas jego pracy w polu.

Wszystkie eksperymenty były przeprowadzane w środowisku programistycznym opartym na języku Python, z wykorzystaniem popularnych narzędzi do głębokiego uczenia takich jak: PyTorch [Paszke i in., 2019], Timm [Wightman, 2019] czy też 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 zastosowanej architektury, typu analizowanego obrazowania oraz złożoności poszczególnych zadań. 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. 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. Zasoby obliczeniowe były w pełni wykorzystywane podczas trenowania, z GPU pracującym na maksymalnej wydajności [NVIDIA, 2020].

3.5. Proces detekcji i klasyfikacji obiektów

Podsumowując, cały złożony 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 rozpoznawanie i klasyfikacja różnych faz rozwoju kukurydzy zgodnie z międzynarodową skalą BBCH, a także ocena poziomów nawodnienia kukurydzy oraz identyfikacja kukurydzy zdrowej i porażonej wybranymi patogenami.

Pierwszym krokiem było przygotowanie i dostosowanie modeli głębokich sieci neuronowych do specyfiki określonego zadania. W pracy zastosowano dwa zestawy zaawansowanych modeli, dostosowanych do odpowiednich celów badawczych. Do zadania polegającego na detekcji i klasyfikacji faz rozwojowych kukurydzy zastosowano modele HTC_x101, HTC_r101, HTC_r50 [Chen i in., 2019] oraz Mask2Former [Chen i in., 2021]. Każdy z nich został zaimplementowany z wykorzystaniem kodów źródłowych pochodzących z repozytorium MMDetection [MMDetection, 2020], które następnie odpowiednio

zmodyfikowano, aby działały optymalnie w kontekście prowadzonej analizy obrazów kukurydzy.

Modele HTC (Hybrid Task Cascade), a w szczególności HTC_x101, HTC_r101 i HTC_r50 [Chen i in., 2019], wybrano ze względu na ich zdolność do jednoczesnej segmentacji semantycznej, instancyjnej oraz detekcji obiektów. Implementacja tychże 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 [Chen i in., 2021], oparty na architekturze typu 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 obrazach rolniczych.

Drugi zestaw modeli – 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] – zastosowano do zadań związanych z określaniem poziomów nawodnienia kukurydzy oraz klasyfikacją porażenia kukurydzy wybranymi patogenami. Modele te zostały wybrane ze względu na ich zdolność do efektywnego przetwarzania złożonych wzorców na obrazach oraz analizy danych wielospektralnych. ConvNeXt Small [Liu i in., 2022], 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 [He i in., 2016], znany ze swojego mechanizmu *residual connections*, umożliwił skuteczne trenowanie głębokiej sieci przy zachowaniu stabilności procesu uczenia. EfficientNet-B3 [Tan & Le, 2019], czyli model znany ze zdolności do efektywnego skalowania, pozwolił na uzyskanie wysokiej dokładności przy optymalnym zużyciu zasobów obliczeniowych. Vision Transformer [Dosovitskiy i in., 2020] i Swin Transformer [Liu i in., 2021], reprezentujące nowoczesne podejście do przetwarzania obrazów z wykorzystaniem mechanizmów Transformerów, 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 [Chen i in., 2019] oraz Mask2Former [Chen i in., 2021] 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 [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] zawierały obrazy pozwalające na ocenę poziomów nawodnienia kukurydzy oraz identyfikację objawów porażenia kukurydzy wybranymi patogenami. 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 (tzw. *step-based learning rate schedule*). Każdy model był trenowany przez określoną liczbę epok, a w trakcie treningu monitorowana była jego wydajność, aby zapewnić optymalne dostrojenie parametrów.

Modele HTC [Chen i in., 2019] i Mask2Former [Chen i in., 2021] zastosowane zostały do wykrywania obiektów na obrazach w zadaniu dotyczącym 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 te, dzięki swoim zaawansowanym mechanizmom segmentacji instancyjnej, umożliwiły precyzyjne rozdzielenie i klasyfikację różnych faz rozwoju roślin nawet w złożonych obrazach.

Modele 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] były z kolei stosowane do zadań związanych z oceną poziomów nawodnienia kukurydzy oraz identyfikacją objawów porażenia kukurydzy wybranymi patogenami. Dzięki zaawansowanym technikom przetwarzania danych, takim 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 cechy roślin.

Wyniki detekcji i klasyfikacji zostały przeanalizowane, aby ocenić skuteczność i precyzję działania wytrenowanych modeli. Analiza ta obejmowała ocenę jakości predykcji na podstawie wskaźników takich jak Accuracy, Precision, Recall oraz F1-score, co pozwoliło na pełną ocenę zdolności modeli do wykrywania i klasyfikacji zarówno faz rozwoju kukurydzy, jak i poziomów nawodnienia oraz porażenia roślin wybranymi patogenami. Szczegółowe dane na temat uzyskanych wyników zostały omówione z rozdziałach 4, 5, 6 i 7.

3.5.1. Zastosowanie modeli

W ramach niniejszej pracy implementacja modeli do detekcji i klasyfikacji faz rozwoju kukurydzy, poziomów nawodnienia oraz objawów porażenia kukurydzy wybranymi patogenami została przeprowadzona w specjalnie przygotowanym środowisku badawczym. Aby zapewnić optymalną wydajność oraz dokładność analiz, wykorzystano narzędzia takie jak PyTorch oraz środowisko obliczeniowe oparte na jednostkach GPU. Wykorzystana została w tym celu karta graficzna 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 [NVIDIA, 2020]. Dzięki temu możliwe było szybkie przetwarzanie dużych zbiorów danych obrazowych, co znacznie przyspieszyło proces implementacji i trenowania modeli.

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 [NVIDIA, 2020], [MMDetection, 2020].

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 porcji danych (tzw. *batchy*), co umożliwiało równoczesne przetwarzanie wielu obrazów, maksymalizując efektywność obliczeń na GPU.

Podczas implementacji napotkano 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 porcji danych lub obrazów o wysokiej rozdzielczości. Aby zminimalizować ryzyko wystąpienia tego problemu, zastosowane zostały techniki optymalizacji pamięci, takie jak zmniejszenie rozmiaru porcji danych.

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 objawy porażenia kukurydzy wybranymi patogenami (każde z tych zadań realizowane było oddzielnie przez osobne modele). To z kolei umożliwiło osiągnięcie wysokiej precyzji i skuteczności, co stanowiło istotny element w realizacji postawionych celów.

3.5.2. Klasyfikator głosujący i jego zastosowanie

Dla danych ze zbioru obrazów wielospektralnych zarejestrowanych przez kamerę MicaSense RedEdge 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 pozyskiwania obrazów powodowały zmianę kształtu rejestrowanych obiektów. Podczas gdy z jednego obiektywu liść był dość prosty, inny obiektyw

rejestrował go jako liść 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 MicaSense RedEdge MX-Dual wykorzystywać jednocześnie wszystkie kanały spektralne. Naturalnym rozwiązaniem okazało się przyjęcie metody głosowania poszczególnych modeli wytrenowanych na pojedynczych kanałach spektralnych. W opracowanym rozwiązaniu ocenie podlegały dwa podejścia wyboru klasy:

- pojedynczy model - model wytrenowany na pojedynczym kanale spektralnym,
- połączenie wielu pojedynczych modeli - 10 modeli wytrenowanych na różnych pojedynczych kanałach spektralnych głosuje nad wyborem klasy („faza10”, „faza11”, … „faza19”) i wygrywa klasa większościowa.

Klasyfikator głosujący to metoda agregacji wyników predykcji pochodzących z różnych modeli, która może poprawić skuteczność klasyfikacji w systemach uczenia maszynowego. W badaniach opisanych w niniejszej rozprawie klasyfikator głosujący został zastosowany do analizy wyników modeli uczenia głębokiego wytrenowanych na pojedynczych kanałach spektralnych. W celu realizacji tego podejścia stworzono 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, która jest odpowiedzialna za ekstrakcję prawdopodobieństw predykcji z wyników modeli zapisanych w plikach 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, podczas agregacji predykcji i wyników wykrycia w postaci poligonów, istotne znaczenie miały predykcje dotyczące poszczególnych roślin, a nie całych obrazów. Wszystkie predykcje zostały powiązane i zagregowane na podstawie unikalnego identyfikatora rośliny. Ponieważ dla każdej rośliny zarejestrowano wiele predykcji i odpowiadających im wyników w postaci poligonów, konieczne było wybranie jednej klasy reprezentatywnej dla każdej rośliny.

Wybór przewidywanej klasy z wyników wykrycia w postaci poligonów opierał się na zasadzie większości. Spośród klas zwróconych przez model wybierano tę, która występowała

najczęściej. W przypadku remisu decydowała klasa z najwyższą średnią wartością przewidywaną dla danego poligonu. Jeśli wykryte poligony przypisano do więcej niż jednej klasy, zastosowano analogiczne podejście: wybierano klasę występującą najczęściej, a w przypadku remisu decydowała klasa o największej średniej powierzchni poligonów.

Każdy z dziesięciu modeli głosujących był wytrenowany na jednym z dziesięciu zbiorów obrazów odpowiadających poszczególnym kanałom spektralnym kamery MicaSense RedEdge MX-Dual. Modele te zostały wytrenowane z wykorzystaniem algorytmu, który osiągnął najlepsze wyniki podczas badań prowadzonych w ramach przygotowywania niniejszej rozprawy.

Klasyfikator głosujący działa na zasadzie agregacji predykcji z poszczególnych klasyfikatorów podstawowych. Każdy z tych klasyfikatorów dokonuje predykcji dla danej rośliny, przy czym obrazy są wykonywane z różnych kątów nachylenia za pomocą różnych kamer. Identyfikacja kolejnych roślin jest możliwa dzięki unikalnym identyfikatorom (id), które są przypisane do nazw plików poszczególnych obrazów. Po zebraniu predykcji wybierana jest klasa większościowa, czyli taka, która uzyskała najwięcej głosów z poszczególnych klasyfikatorów. Klasyfikator głosujący agreguje wyniki i zwraca klasę większościową jako uzyskaną decyzję.

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 wyników w klasyfikatorze głosującym polega na łączeniu predykcji z każdego z pojedynczych klasyfikatorów, odpowiadających kolejnym kanałom spektralnym. Wyniki zwracane przez klasyfikatory mają strukturę:

- obraz_1 :
    - klasa_1:
        - polygon_1: score
        - polygon_2: score
        - …
    - klasa_2:

    - ...
- obraz_2
- ...

Klasyfikator głosujący agreguje wyniki, wybierając najwyższy score spośród wszystkich klas dla danego obrazu. Klasa o najwyższym score zostaje przypisana jako wynikowa klasa dla tego obrazu. W opracowanym podejściu największy score spośród wszystkich klas determinuje klasę wynikową. Otrzymane wyniki mają strukturę:

- obraz_1: klasa_6
- obraz_2: klasa_5
- ...

Agregacja oznaczeń w zbiorze walidacyjnym do klas polega na tym, że dla każdego kanału spektralnego definiowane są poligony przypisane do różnych klas. W niektórych przypadkach na jednym obrazie w danym kanale może występować więcej niż jeden poligon dla tej samej klasy, a także poligony różnych klas w różnych częściach obrazu. Aby w ramach eksperymentu przypisać każdemu obrazowi jedną konkretną klasę, wybrano poligon o największej powierzchni jako reprezentatywny dla danego obrazu.

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 z rzeczywistymi klasami. Predykcje są grupowane dla każdej rośliny (file_id) w taki sposób, że spośród wszystkich kanałów spektralnych wybierana jest klasa najczęściej występująca, czyli statystyczna moda. W sytuacji, gdy dwie lub więcej klas występuje równie często, porównywane są dodatkowe wartości:

- „area" dla wykrytych poligonów – wybierana jest klasa o największej średniej powierzchni poligonów.
- „score" dla predykcji – wybierana jest klasa o najwyższym średnim score.

Po wyznaczeniu zagregowanych klas można obliczyć typowe dla klasyfikacji metryki, takie jak: accuracy (stosunek liczby poprawnie sklasyfikowanych obiektów do całkowitej liczby obiektów), precision (stosunek liczby poprawnie sklasyfikowanych obiektów danej klasy do liczby wszystkich obiektów sklasyfikowanych jako należące do tej klasy), recall (stosunek liczby poprawnie sklasyfikowanych obiektów danej klasy do liczby wszystkich obiektów rzeczywiście należących do tej klasy) oraz F1-score (średnia harmoniczna metryk precision i recall). Metryki te umożliwiają kompleksową ocenę jakości klasyfikacji [Powers, 2011].

Opracowanie takiego podejścia pociągnęło za sobą dodatkowy cel badawczy, 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 poszczególnych kanałów spektralnych kamery wielospektralnej MicaSense RedEdge MX Dual.

Po przetworzeniu danych z obu źródeł, tj. predykcji oraz adnotacji, następuje etap porównania i oceny wyników, realizowany przez dedykowaną funkcję. Jest ona odpowiedzialna za obliczenie kluczowych miar jakości klasyfikacji, takich jak wspomniane miary: accuracy, precision, recall oraz 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].

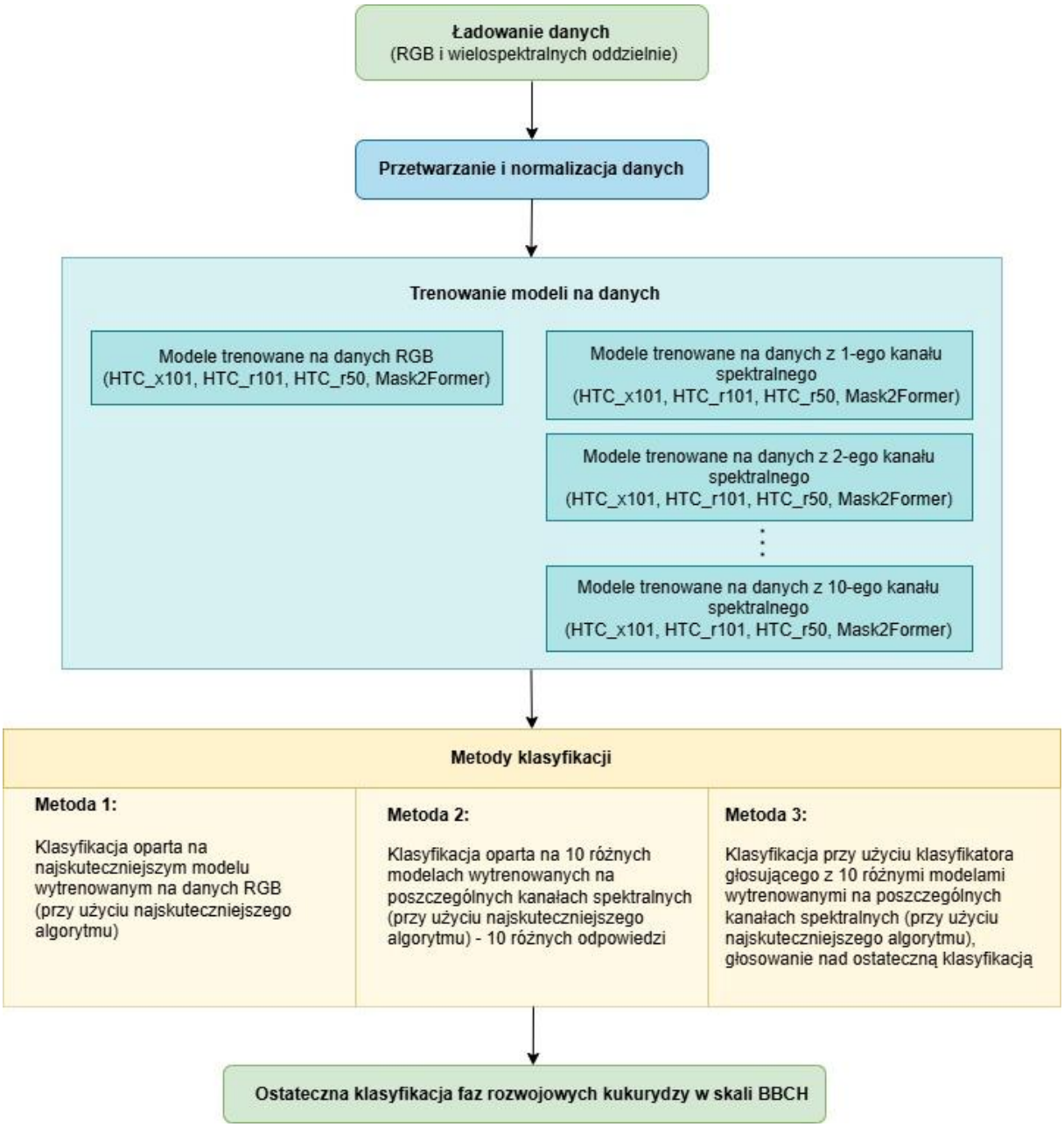
Główny proces walidacji wyników jest realizowany przez odpowiednią funkcję, która łączy wyniki predykcji z danymi adnotacji, a następnie wywołuje inną funkcję, która oblicza miary jakości klasyfikacji. Potem następuje dostosowanie sposobu przetwarzania danych oraz ich oceny w zależności od typu klasyfikatora. W tym przypadku jest to klasyfikator głosujący oparty na modelach wytrenowanych dla poszczególnych pojedynczym kanałów spektralnych.

4. Detekcja i klasyfikacja faz rozwoju kukurydzy w skali BBCH z wykorzystaniem obrazowania RGB i obrazowania z kamery wielospektralnej MicaSense RedEdge MX Dual

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 [Chen i in., 2019] oraz Mask2Former [Chen i in., 2021]. Architektury te mają optymalne parametry pod względem przeprowadzania procesu trenowania i nie wymagają zbyt dużych zasobów obliczeniowych do trenowania przy zastosowaniu stosunkowo mało rozbudowanych zbiorów treningowych. Zbadana została ich efektywność i skuteczność w przypadku trenowania na zbiorze RGB oraz na zbiorze obrazów wielospektralnych z kamery MicaSense RedEdge MX Dual. W przypadku obrazów wielospektralnych badania zostały przeprowadzone dzieląc zbiór danych wielospektralnych na poszczególne pojedyncze kanały spektralne. To umożliwiło uzyskanie odpowiedzi, który z badanych algorytmów radzi sobie najlepiej na wskazanych zbiorach danych (RGB lub wielospektralnych) 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. Badania te rozszerzają świat analizy danych o inne spektra niż popularne dotychczas zobrazowanie RGB. Prezentowane prace otwierają pole do zbadania nowych zagadnień i były inspiracją do kontynuacji badań z wykorzystaniem nowego zbioru obrazów wielospektralnych oraz użycia w tym celu również innej kamery wielospektralnej MapIR Survey3 Red + Green + NIR, 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 dwóch typów zobrazowań: zobrazowania RGB oraz zobrazowania pozyskiwanego za pomocą kamery wielospektralnej MicaSense RedEdge MX Dual. Co warto podkreślić, z uwagi na przesunięcie i różny kształt

obiektów rejestrowanych przez poszczególne kanały kamery wielospektralnej, wynikające z pozyskiwania obrazów z bliskiej odległości i rożnych kątów patrzenia poszczególnych kamer odpowiadających różnym kanałom spektralnym, niemożliwe było połączenie tychże pojedynczych zobrazowań w jedno wspólne zobrazowanie. Konieczna była zatem analiza wyników generowanych przez modele wytrenowane na obrazach z pojedynczych kanałów spektralnych. Aby jednak wykorzystać wspólny potencjał, który tkwi w zobrazowaniach generowanych przez zastosowaną kamerę wielospektralną, opracowano metodę klasyfikacji opartą o prosty 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 kukurydzy w skali BBCH. Zaprezentowane zostanie porównanie wyników uzyskanych na danych RGB, wyników uzyskanych na danych wielospketralnych 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.

W tabeli 4.1. przedstawione zostały wyniki klasyfikacji dla badanych kanałów spektralnych zarejestrowane dla każdego z badanych algorytmów. Model wytrenowany na podstawie danych z kanału nr 10 wyraźnie odbiega skutecznością klasyfikacji od modeli wytrenowanych na bazie pozostałych kanałów spektralnych. Zaprezentowana tabela ilustruje efektywność poszczególnych metod klasyfikacji w oparciu o metryki: accuracy, precision, recall i F1-score.

kanał	accuracy				precision			
	htc_r101	htc_r50	htc_x101	mask2former	htc_r101	htc_r50	htc_x101	mask2former
01	0.486957	0.552174	0.547826	0.539130	0.486508	0.592528	0.582530	0.550819
02	0.568282	0.559471	0.559471	0.555066	0.594116	0.563848	0.586652	0.611719
03	0.548246	0.550661	0.530702	0.508772	0.587350	0.573304	0.570517	0.567900
04	0.567100	0.549784	0.541126	0.510823	0.603094	0.587433	0.563036	0.502541
05	0.575893	0.580357	0.558036	0.531250	0.596516	0.605068	0.578664	0.545954
06	0.547085	0.551570	0.581081	0.520179	0.570063	0.577274	0.591761	0.545100
07	0.582609	0.565217	0.569565	0.495652	0.606866	0.581716	0.593181	0.566532
08	0.538117	0.522321	0.540179	0.486607	0.575040	0.556559	0.572584	0.504101
09	0.551570	0.569507	0.549550	0.524664	0.586433	0.584708	0.561682	0.540334
10	0.219298	0.223684	0.214912	0.232456	0.161993	0.175143	0.162701	0.174677
kanał	recall				F1-score			
	htc_r101	htc_r50	htc_x101	mask2former	htc_r101	htc_r50	htc_x101	mask2former
01	0.546268	0.608676	0.582942	0.547557	0.482994	0.558260	0.543616	0.506817
02	0.611719	0.619491	0.589024	0.599922	0.563584	0.554428	0.564948	0.543282
03	0.593651	0.591402	0.566799	0.549762	0.552418	0.557723	0.521853	0.509900
04	0.626295	0.600151	0.566362	0.557919	0.566393	0.555305	0.539768	0.494669
05	0.621161	0.630494	0.587758	0.571906	0.585859	0.589184	0.552485	0.523784
06	0.571536	0.585762	0.610729	0.558193	0.546741	0.556209	0.591596	0.521969
07	0.627316	0.587195	0.602076	0.541441	0.587952	0.560104	0.570215	0.474963
08	0.561341	0.553343	0.555533	0.498973	0.542855	0.529349	0.552871	0.476158
09	0.611539	0.596285	0.565288	0.541253	0.557820	0.570818	0.544688	0.527224
10	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.

Analizując powyższe wyniki można wyciągnąć istotne wnioski. Przy wykorzystaniu algorytmu HTC z ResNet101 jako szkielet sieci najlepsze wyniki uzyskano dla kanału

spektralnego nr 07 (red edge 705 (10)). Przy wykorzystaniu algorytmu HTC z ResNet50 jako szkielet sieci najlepsze wyniki uzyskano dla kanału spektralnego nr 05 (red 650 (16)). Przy wykorzystaniu algorytmu HTC z ResNeXt101 jako szkielet sieci 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 szkielet sieci, 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 szkielet sieci (według metryk accuracy i recall), Mask2Former (według metryki precision) oraz HTC z ResNeXt101 jako szkielet sieci (według metryki F1-score).
- Dla kanału spektralnego nr 03 (green 531 (14)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmów: HTC z ResNet50 jako szkielet sieci (według metryk accuracy i F1-score) oraz HTC z ResNet101 jako szkielet sieci (według metryk precision i recall).
- Dla kanału spektralnego nr 04 (green 560 (27)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmu HTC z ResNet101 jako szkielet sieci, zgodnie ze wszystkimi miarami użytymi do oceny wydajności modeli klasyfikacyjnych (accuracy, precision, recall, F1-score).
- Dla kanału spektralnego nr 05 (red 650 (16)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmu HTC z ResNet50 jako szkielet sieci, 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 szkielet sieci, 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 szkielet sieci, 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 szkielet sieci (według metryk precision i recall) oraz HTC z ResNeXt101 jako szkielet sieci (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 szkielet sieci (według metryk accuracy i F1-score) oraz HTC z ResNet101 jako szkielet sieci (według metryk precision i recall).
- Dla kanału spektralnego nr 10 (NIR 842 (57)) najlepsze wyniki klasyfikacji osiągnięto przy wykorzystaniu algorytmów: Mask2Former (według metryk accuracy i recall) oraz HTC z ResNet50 jako szkielet sieci (według metryk precision i F1-score).

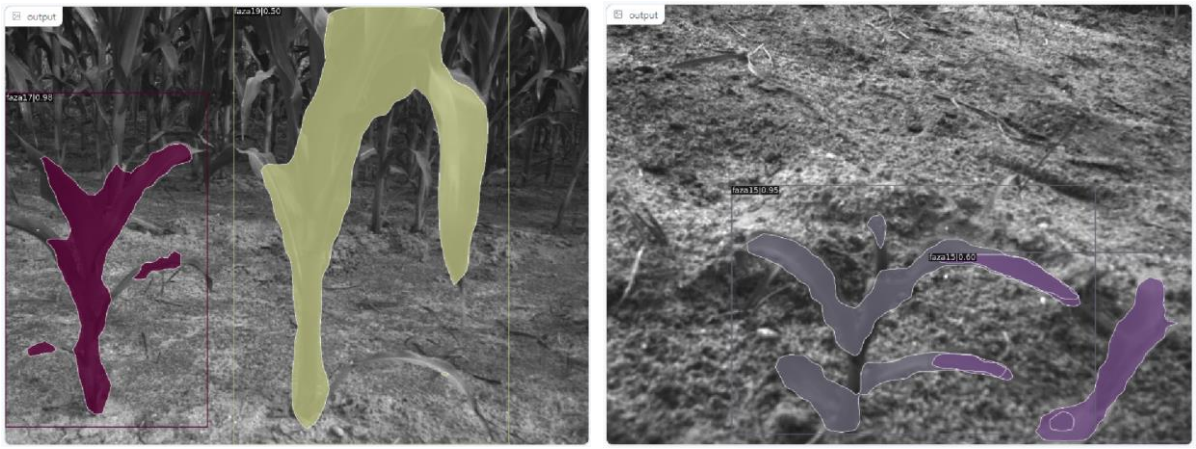
W Tabeli 4.2. przedstawiono wyniki klasyfikacji dla każdego badanego algorytmu. Tabela ta ilustruje dokładność poszczególnych metod klasyfikacyjnych na danych z poszczególnych kanałów spektralnych w oparciu o metryki oceny takie jak: accuracy, precision, recall oraz F1-score.

Kanał spektralny	Długość fali (nm)	Najlepszy algorytm	Metryki oceny
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 4.2. Wyniki klasyfikacji na danych z poszczególnych kanałów spektralnych wraz ze wskazaniem najskuteczniejszych algorytmów według poszczególnych metryk oceny.

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.

Na rysunku 4.2. przedstawiono przykładowe wyniki działania modelu wytrenowanego w ramach omawianego zadania. Widać, że sieć w sposób poprawny wykryła obiekty w postaci kukurydzy i prawidłowo sklasyfikowała fazy rozwojowe w skali BBCH.



Rysunek 4.2. Wynik predykcji modelu wytrenowanego na bazie obrazów wielospketralnych.

4.3. Wyniki uzyskane dla danych RGB

Oprócz badań z wykorzystaniem obrazów wielospketralnych przeprowadzone zostały również badania z użyciem obrazów RGB. Jak zostało to omówione w rozdziale *3.2.3. Dane RGB*, przygotowano dwa zbiory danych RGB: pierwszy (mniej liczny) będący zbiorem referencyjnym dla zbioru obrazów wielospektralnych z kamery MicaSense RedEdge MX Dual oraz drugi (bardziej rozbudowany), który stworzono w celu obserwacji postępów w uczeniu poszczególnych algorytmów w zależności od liczności zbioru treningowego. 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. W tabeli 4.3. zamieszczono 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. Do analizy wykorzystano metryki oceny: accuracy, precision, recall i F1-score.

		accuracy	precision	recall	F1-score
htc_r101	rgb	0.706667	0.460286	0.496550	0.465737
	rgb_większy_zbiór_danych	0.673469	0.695714	0.683929	0.654048
htc_r50	rgb	0.680000	0.424074	0.404996	0.406152
	rgb_większy_zbiór_danych	0.612245	0.635736	0.617857	0.598121
htc_x101	rgb	0.760000	0.525599	0.580694	0.524339
	rgb_większy_zbiór_danych	0.591837	0.646688	0.628571	0.605777
mask2former	rgb	0.800000	0.596212	0.660516	0.600132
	rgb_większy_zbiór_danych	0.551020	0.657080	0.581548	0.525582

Tabela 4.3. 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.

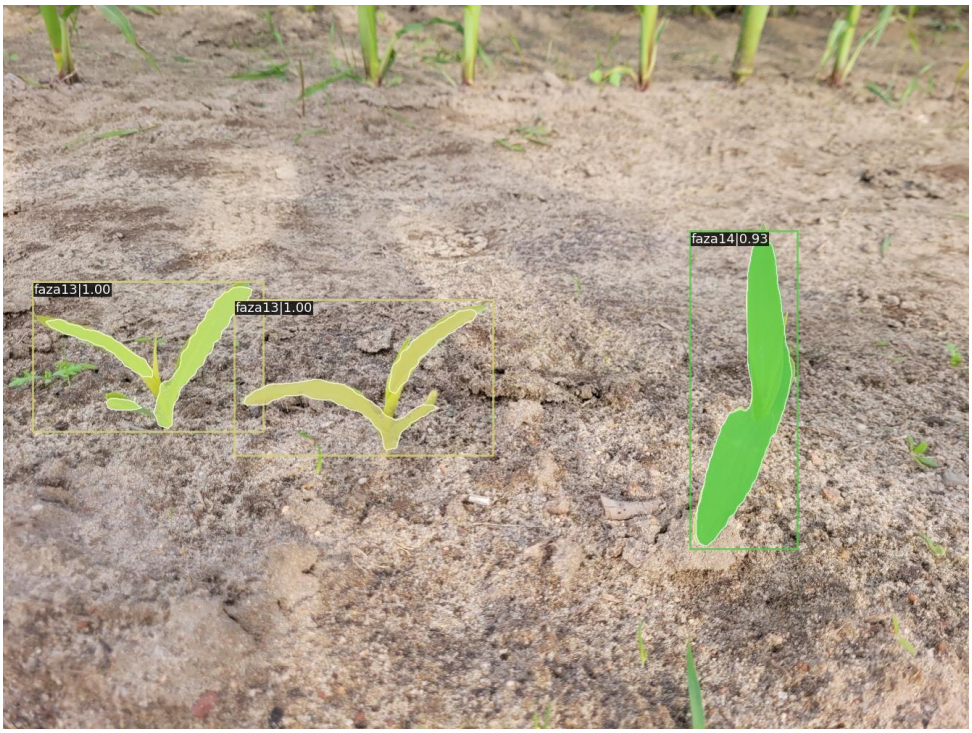
Najlepsze wyniki klasyfikacji w oparciu o trening na zbiorze RGB (referencyjnym względem danych z RedEdge) uzyskał algorytm Mask2Former. W przypadku trenowania na większym zbiorze obrazów RGB najlepsze wyniki uzyskał algorytm HTC z ResNet101 jako szkielet sieci i wyniki te były lepsze niż wyniki podczas treningu na mniejszym zbiorze danych, co świadczy o postępach w uczeniu wraz ze wzrostem liczności zbioru treningowego. Fakt ten

pozwolił wyciągnąć istotny wniosek badawczy i projektowy, wskazując, że do procesu klasyfikacji obrazów RGB spośród badanych modeli najlepiej stosować model oparty o algorytm HTC z ResNet101 jako szkielet sieci wytrenowany na większym zbiorze RGB.

Na Rysunku 4.3. widać wyniki pierwszych eksperymentów, które prowadzone były na jeszcze mniejszym zbiorze treningowym, niż docelowe dwa zbiory danych RGB. Model 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 wytrenowanego modelu – pierwsze eksperymenty.

Po przeprowadzeniu dalszych treningów, rozszerzeniu zbioru treningowego oraz po przeanalizowaniu większej ilości algorytmów opartych o głębokie sieci neuronowe uzyskano dalszą poprawę wyników, a przetestowanie wielu różnego rodzaju obrazów kukurydzy pokazało, że opracowane rozwiązanie poprawnie wykrywa obiekty w postaci kukurydzy oraz poprawnie przypisuje im fazy rozwojowe w skali BBCH, zgodnie ze zbiorem: *faza10, faza11, faza12, faza13, faza14, faza15, faza16, faza17, faza18, faza19*. Na Rysunku 4.4. zaprezentowano wynik predykcji modelu wytrenowanego na powiększonym zbiorze obrazów RGB. Kukurydza została prawidłowo wykryta, algorytm prawidłowo zaklasyfikował wartości fazy rozwojowej, a wykryty obszar praktycznie idealnie pokrywa się z obszarem obrazu, na którym znajduje się kukurydza.



Rysunek 4.4. Wynik predykcji modelu wytrenowanego na powiększonym zbiorze obrazów RGB.

Poniżej przedstawiono kilka dodatkowych obrazów, na których widać wyniki działania modelu wytrenowanego przy użyciu algorytmu HTC z ResNeXt101 jako szkielet sieci.

Rysunek 4.5. Wynik predykcji modelu wytrenowanego na powiększonym zbiorze obrazów RGB.

Rysunek 4.6. Wynik predykcji modelu wytrenowanego na powiększonym zbiorze obrazów RGB.

Rysunek 4.7. Wynik predykcji modelu wytrenowanego na powiększonym zbiorze obrazów RGB.

4.4. Wyniki uzyskane przy zastosowaniu klasyfikatora głosującego

W Tabeli 4.4. zebrane zostały wyniki dotyczące metryk oceny 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 pojedynczych kanałach spektralnych. Do oceny skuteczności poszczególnych metod posłużyły metryki oceny: accuracy, precision, recall oraz F1-score.

		accuracy	precision	recall	F1-score
htc_r101	klasyfikator_głosujący	0.651466	0.684678	0.686132	0.643477
	rgb	0.706667	0.460286	0.496550	0.465737
htc_r50	klasyfikator_głosujący	0.661238	0.690290	0.699036	0.659793
	rgb	0.680000	0.424074	0.404996	0.406152
htc_x101	klasyfikator_głosujący	0.657980	0.663664	0.676181	0.652508
	rgb	0.760000	0.525599	0.580694	0.524339
mask2former	klasyfikator_głosujący	0.625407	0.615539	0.630647	0.592380
	rgb	0.800000	0.596212	0.660516	0.600132

Tabela 4.4. Porównanie wyników klasyfikacji osiąganych z użyciem poszczególnych modeli wytrenowanych na zbiorze obrazów RGB z wynikami klasyfikacji osiąganymi przez te modele po ich wytrenowaniu z użyciem obrazów wielospektralnych oraz zastosowaniu metody klasyfikatora głosującego.

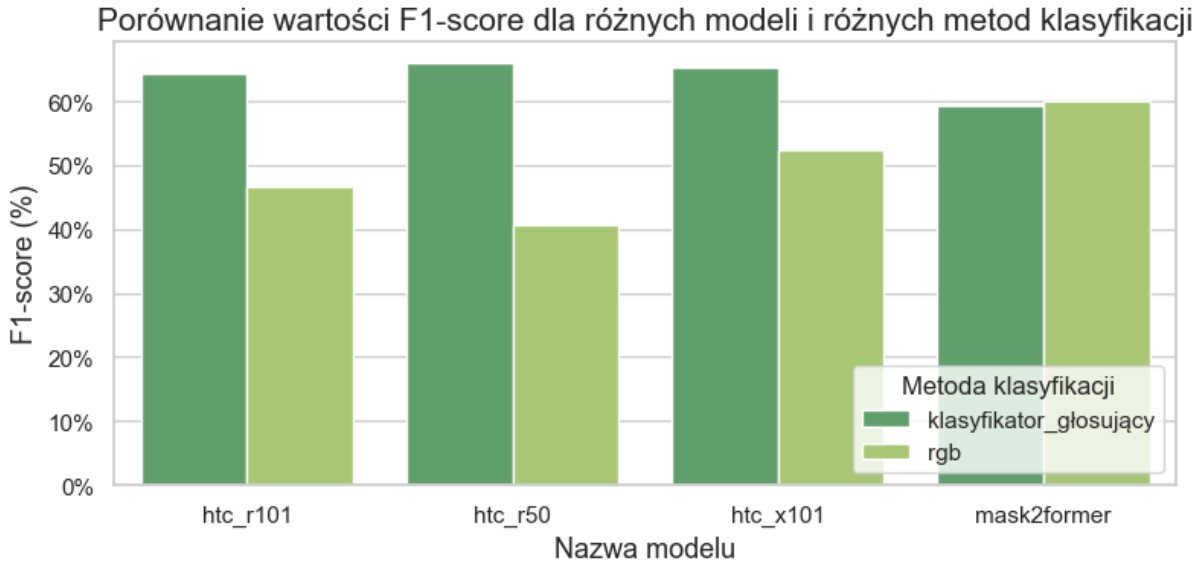
Analizując poszczególne wyniki, można wyciągnąć istotne wnioski. W przypadku algorytmu HTC z ResNet101 jako szkielet sieci, model wytrenowany na podstawie zbioru obrazów RGB okazał się lepszy od klasyfikatora głosującego jedynie pod względem metryki accuracy. Metryki precision, recall i F1-score wskazują natomiast na lepsze wyniki w przypadku zaproponowanej metody klasyfikatora głosującego, niż w przypadku trenowania na zbiorze obrazów RGB.

W przypadku algorytmu HTC z ResNet50 jako szkielet sieci, model wytrenowany na podstawie zbioru obrazów RGB także okazał się lepszy od zaproponowanej metody głosowania jedynie na podstawie metryki accuracy. Natomiast według metryk precision, recall i F1-score, to zaproponowana metoda klasyfikatora głosującego pozwala uzyskać lepsze wyniki.

W przypadku algorytmu HTC z ResNeXt101 jako szkielet sieci, model wytrenowany na podstawie zbioru obrazów RGB tylko pod względem metryki accuracy okazał się lepszy od metody z wykorzystaniem klasyfikatora głosującego. Z kolei metryki precision, recall i F1-score pokazały, że jest zaproponowana metoda głosowania pojedynczych modeli wytrenowanych na podstawie obrazów z pojedynczych kanałów spektralnych jest lepsza niż model wytrenowany na podstawie obrazów RGB.

W przypadku algorytmu Mask2Former, zaproponowana metoda klasyfikatora głosującego okazała się bardziej skuteczna, niż model wytrenowany na zbiorze obrazów RGB jedynie według metryki precision. Metryki accuracy, recall i F1-score pokazały zaś, że model wytrenowany na zbiorze obrazów RGB osiągnął wyniki lepsze niż klasyfikator głosujący.

Rysunek 4.8. przedstawia wykres porównujący wyniki uzyskane przez modele wytrenowane na obrazach RGB oraz wyniki uzyskane dzięki zastosowaniu metody 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 uzyskane przez modele wytrenowane na obrazach RGB oraz wyniki uzyskane dzięki zastosowaniu metody klasyfikatora głosującego, w której modele wytrenowane na pojedynczych kanałach spektralnych głosują nad wynikiem klasyfikacji.

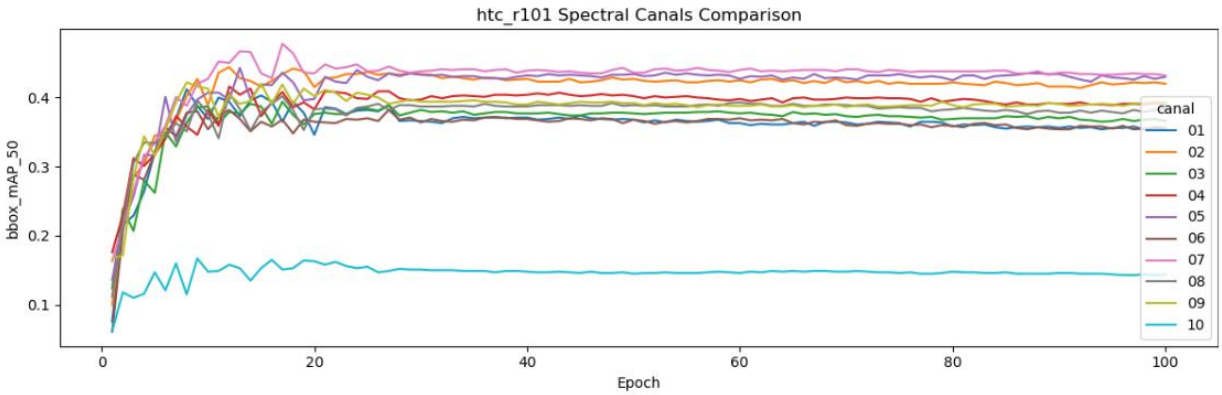
Jak widać na bazie z wykresu, w przypadku algorytmów HTC z ResNet101, HTC z ResNet50 oraz HTC z ResNeXt101 jako szkielet sieci, model wytrenowany na obrazach wielospektralnych i przy zastosowaniu klasyfikatora głosującego okazał się lepszy niż model wytrenowany na obrazach RGB. W przypadku algorytmu Mask2Former metoda z użyciem klasyfikatora głosującego okazała się zaś nieco gorsza, niż model wytrenowany na obrazach RGB.

Pojedyncze modele wytrenowane na danych z pojedynczych kanałów spektralnych mają indywidualnie gorszą jakość, niż modele wytrenowane na obrazach RGB. Przyczyną tego jest fakt, iż zostały one wytrenowane na jednokanałowych obrazach, podczas gdy modele RGB zostały wytrenowane na obrazach trójkanałowych. Zastosowanie techniki głosowania poprawiło jakość sumarycznej predykcji i pozwoliło na przewyższenie wyników uzyskanych przez klasyfikator głosujący w porównaniu do wyników uzyskanych z użyciem modeli wytrenowanych na obrazach RGB.

4.5. Krzywe uczenia dla danych wielospektralnych i RGB

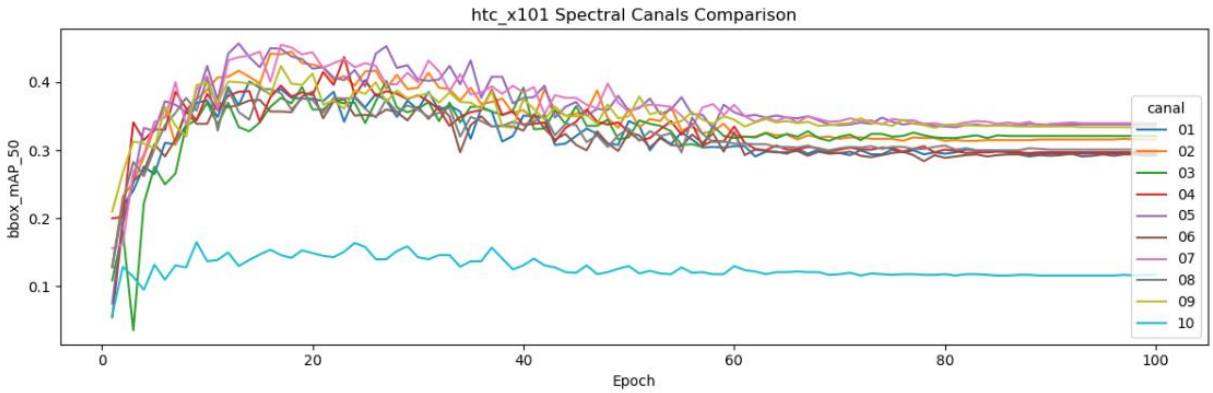
W niniejszym rozdziale zaprezentowane zostały krzywe uczenia zarejestrowane podczas trenowania poszczególnych modeli.

Rysunek 4.9. przedstawia porównanie wyników modelu HTC_r101 trenowanego na poszczególnych kanałach spektralnych w trakcie treningu, który trwał 100 epok. Oś pozioma reprezentuje kolejne epoki, zaś oś pionowa to miara 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.

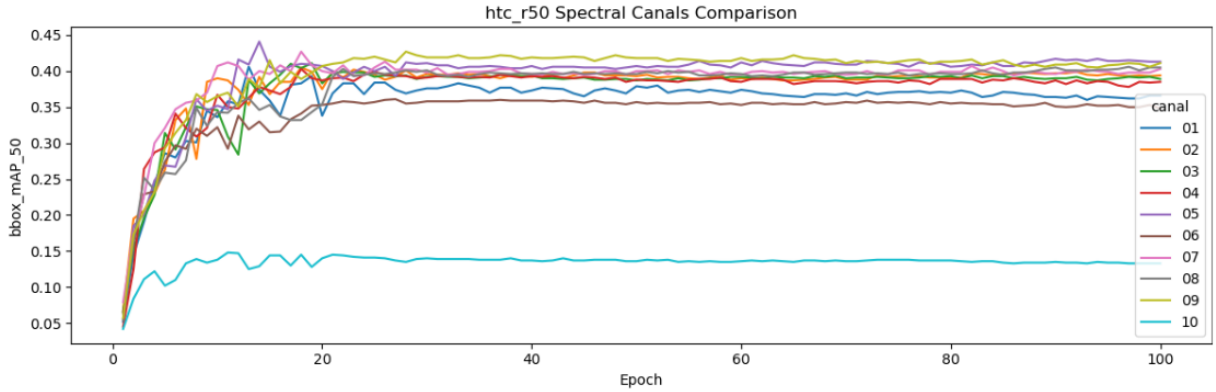
Warto również przyjrzeć się również rysunkowi 4.10. zawierającemu krzywe uczenia modelu opartego o algorytm HTC_x101 dla poszczególnych kanałów spektralnych. Również w tym przypadku wyniki dla kanału nr 10 są najniższe w porównaniu do innych kanałów. Wartości bbox_mAP_50 dla tychże kanałów oscylują wokół 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 5 przy bardzo zbliżonych wynikach również dla kanału nr 7.



Rysunek 4.10. Porównanie wyników modelu HTC_x101 trenowanego na poszczególnych kanałach spektralnych.

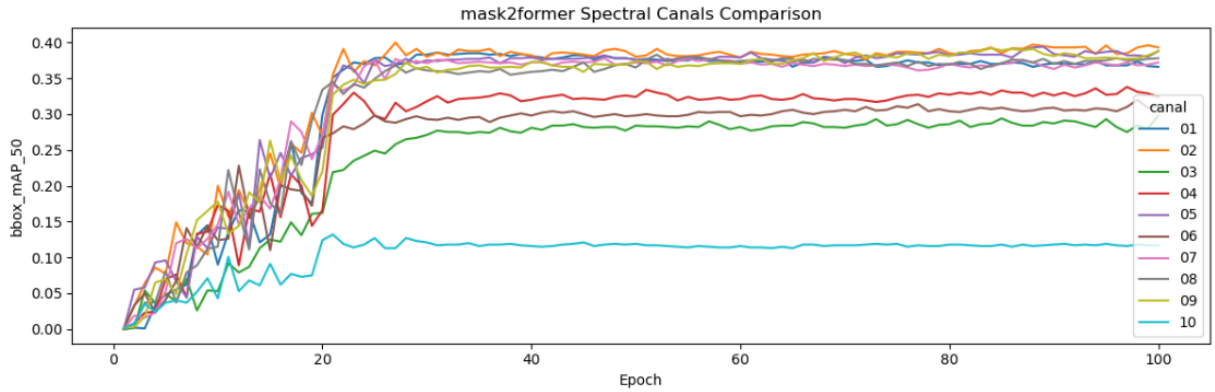
W przypadku krzywych uczenia dla algorytmu HTC_r50 (rysunek 4.11) zaobserwowane wyniki są podobne do krzywych uczenia dla algorytmu HTC_r101. W tym przypadku również wyniki dla kanału nr 10 są najniższe w porównaniu do innych kanałów. Pozostałe modele

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, aż do końca zakresu obserwacji, czyli 100 epok. Wartości dla kanałów 1-9 są bardziej stabilne po osiągnięciu maksimum, co wskazuje na lepszą efektywność w wykrywaniu obiektów przez modele wytrenowane przy ich użyciu.



Rysunek 4.11. Porównanie wyników modelu HTC_r50 trenowanego na poszczególnych kanałach spektralnych.

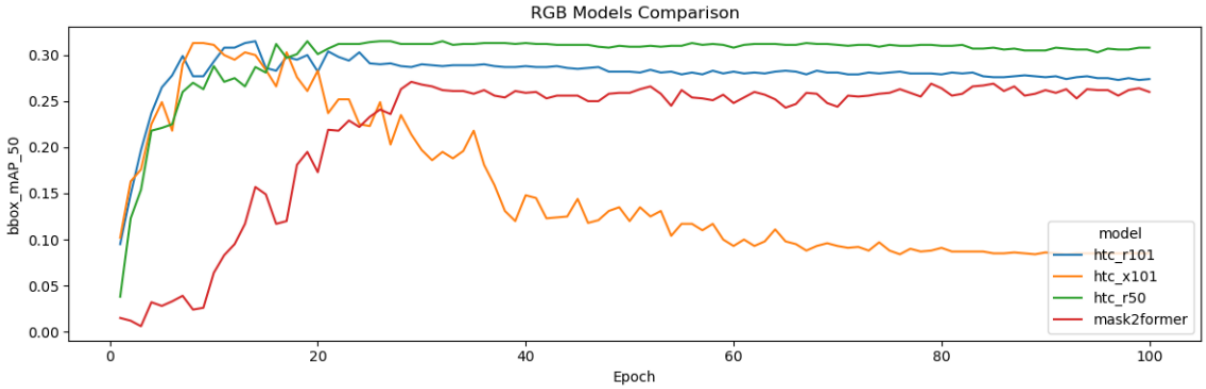
Jak widać na rysunku 4.12. dla algorytmu Mask2Former również można zaobserwować najniższe wyniki dla kanału nr 10, gdzie bbox_mAP_50 oscyluje wokół 0,125 po pierwszych 20 epokach i utrzymuje się na tym poziomie do końca obserwacji, czyli do końca 100 epok. Pozostałe kanały osiągają wyższe wartości bbox_mAP_50, lecz nie stanowią już tak zwartej grupy jak w przypadku poprzednich algorytmów. Spośród wszystkich kanałów najwyższe wartości bbox_mAP osiąga kanał nr 2.



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

Przedstawione wyniki pokazują, iż dla różnych algorytmów trenowanie modeli na obrazach z różnych kanałów spektralnych pozwala osiągać 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 modeli trenowanych na obrazach RGB. Rysunek 4.13 przedstawia porównanie wyników trenowania poszczególnych modeli na zbiorze danych RGB w ciągu 100 epok. Oś pozioma reprezentuje kolejne epoki, zaś oś pionowa przedstawia wartość bbox_mAP_50, czyli miarę dokładności wykrywania obiektów.



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

Model HTC_r101 osiąga stabilne wyniki od około 20 epoki, z wartością bbox_mAP_50 wynoszącą około 0,27. Wartość ta pozostaje stosunkowo stabilna do końca treningu, wykazując niewielkie wahania.

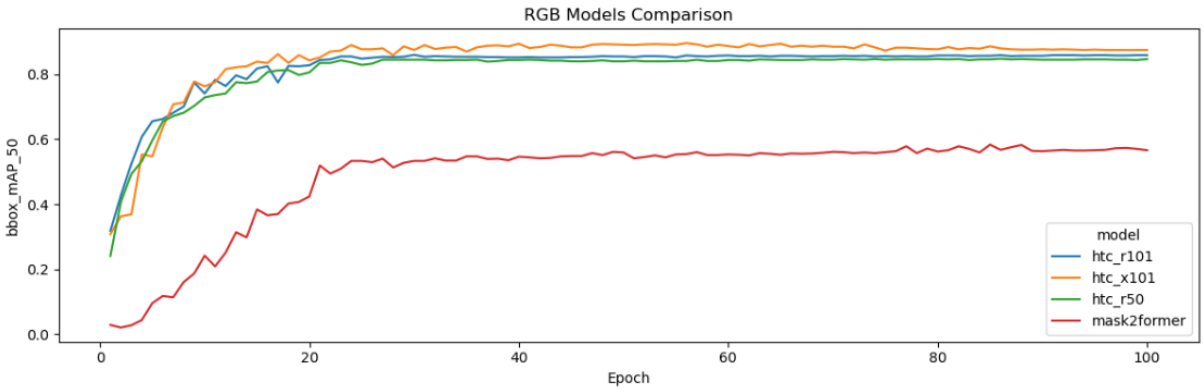
Model wytrenowany na podstawie algorytmu HTC_x101 charakteryzuje się początkowym wzrostem wartości bbox_mAP_50, jednak od około 30 epoki widoczne są wyraźne spadki wydajności, które utrzymują się na poziomie poniżej 0,2 w kolejnych epokach.

Model HTC_r50 osiąga najwyższą stabilność i najlepsze wyniki wśród porównywanych modeli. Już po około 10 epokach wartość bbox_mAP_50 wzrasta do poziomu 0,3 i pozostaje stabilna do końca treningu.

Model Mask2Former uzyskuje najniższe wyniki spośród wszystkich porównywanych modeli. Jego wartość bbox_mAP_50 rośnie powoli, osiągając poziom w przybliżeniu 0,2 około 40 epoki, po czym stabilizuje się z większymi wahaniami.

Analizując wyniki, można zauważyć, iż na analizowanym zbiorze danych modele HTC_r101 i HTC_r50 są bardziej efektywne w wykrywaniu obiektów w porównaniu do modeli HTC_x101 i Mask2Former, które wykazują znacznie większe spadki i wahania. Model HTC_r50 osiąga najwyższą stabilność i najlepsze wyniki spośród wszystkich porównywanych algorytmów.

Warto prześledzić również krzywe uczenia modeli trenowanych na powiększonym zbiorze obrazów RGB. Rysunek 4.14 przedstawia porównanie wyników poszczególnych modeli na powiększonym zbiorze danych RGB w ciągu 100 epok.



Rysunek 4.14. Porównanie przebiegu trenowania poszczególnych modeli trenowanych na powiększonym zbiorze danych RGB.

Model HTC_r101 osiąga dobre wyniki już od początku treningu, a wartość bbox_mAP_50 rośnie dość szybko osiągając poziom w przybliżeniu równy 0,8 już po około 20 epokach i stabilizuje się na tym poziomie do końca treningu.

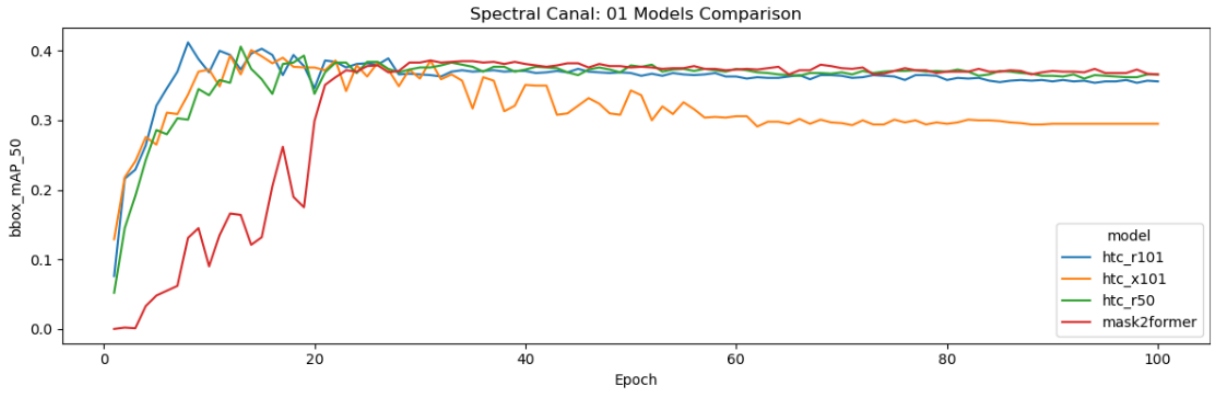
Model wytrenowany na podstawie algorytmu HTC_x101 osiąga najlepsze wyniki spośród wszystkich porównywanych modeli. Wartości bbox_mAP_50 również rosną dość szybko, osiągając poziom nieco powyżej 0,8 po około 20 epokach i utrzymując się na tym poziomie do końca treningu.

Model wytrenowany na bazie algorytmu HTC_r50 osiąga wyniki zbliżone do HTC_r101 i HTC_x101. Wartości bbox_mAP_50 rosną szybko do poziomu w okolicach 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, niż w przypadku innych algorytmów, osiągając poziom w przybliżeniu równy 0,6 po około 60 epokach i utrzymują się na tym poziomie do końca treningu, wykazując większe wahania w porównaniu do innych modeli.

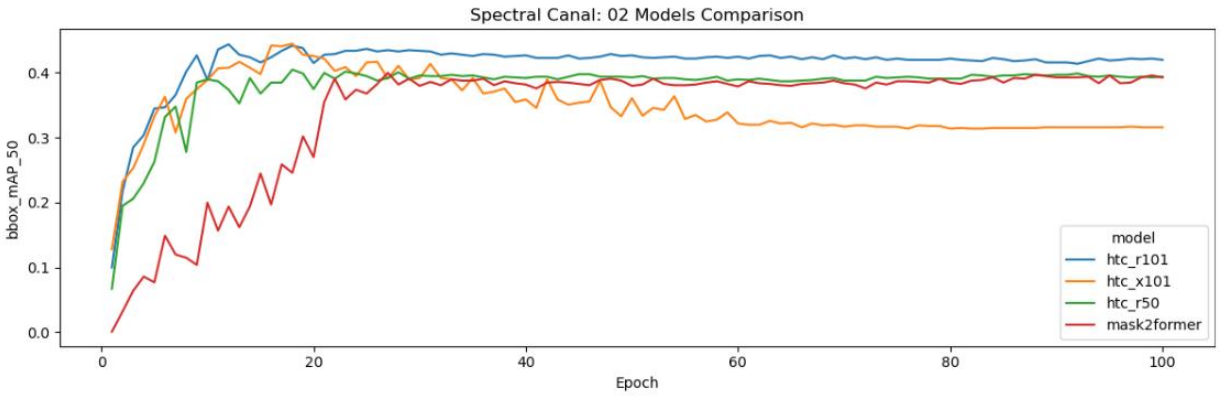
Na podstawie tychże wyników można wywnioskować, iż modele HTC_r101, HTC_x101 i HTC_r50 są bardziej efektywne w wykrywaniu obiektów na zbiorze RGB niż model Mask2Former. Porównując zaś wyniki uzyskiwane przez poszczególne algorytmy trenowane na większym zbiorze danych RGB w porównaniu do ich wyników uzyskiwanych podczas trenowania na mniejszym zbiorze danych RGB można zauważyć, iż zwiększenie wielkości zbioru danych RGB znacząco poprawia efektywność trenowania algorytmów, co przekłada się na wyższą jakość detekcji obiektów oraz większą stabilność wyników. To potwierdza, że dostępność większego zbioru danych jest kluczowa dla poprawnego trenowania algorytmów detekcji obiektów.

Po omówieniu krzywych uczenia dla modeli trenowanych na zbiorze obrazów RGB można przejść do krzywych uczenia dla modeli trenowanych na obrazach z pojedynczych kanałów spektralnych. Poniższe wykresy przedstawiają porównanie wyników dla czterech modeli trenowanych na zbiorach danych odpowiadających poszczególnym kanałom spektralnym, w ciągu 100 epok. Na wszystkich wykresach oś pozioma reprezentuje poszczególne epoki, zaś oś pionowa przedstawia wartości bbox_mAP_50, czyli miary dokładności wykrywania obiektów.



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

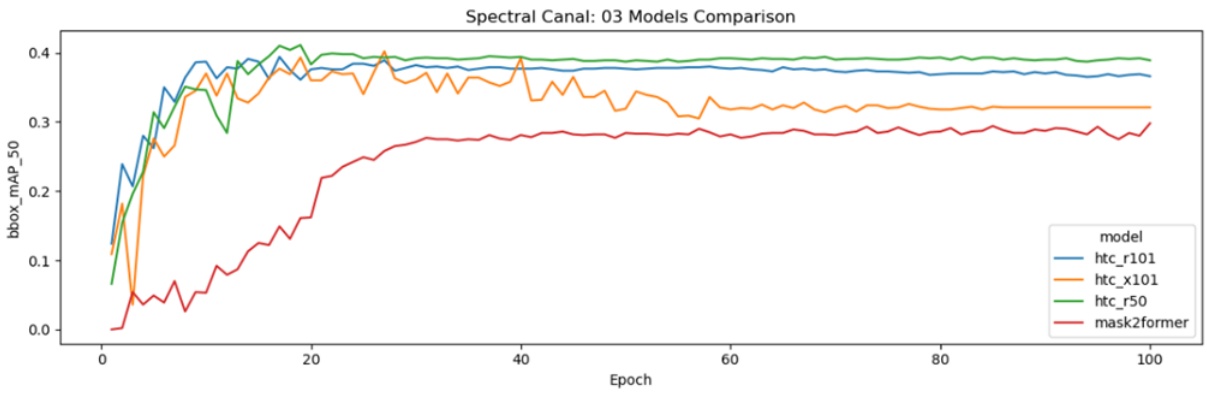
Rysunek 4.15 przedstawia wyniki przebiegu trenowania dla poszczególnych modeli zarejestrowane dla kanału spektralnego nr 1. Modele HTC_r101 i HTC_r50 charakteryzują się szybkim wzrostem w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 wynoszącym około 0,4. Model Mask2Former wykazuje bardziej stopniowy wzrost, osiągając stabilizację po około 20 epokach. Jego końcowa wartość bbox_mAP_50 jest zbliżona do wyników HTC_r101 i HTC_r50, również na poziomie około 0,4, co wskazuje na porównywalną skuteczność w końcowej fazie trenowania. Model HTC_x101 początkowo osiąga wyniki bliskie wartości 0,4 w pierwszych 10-20 epokach, jednak w dalszych etapach treningu jego wydajność spada. Stabilizuje się na niższym poziomie bbox_mAP_50 wynoszącym około 0,3, co świadczy o mniejszej efektywności w porównaniu do pozostałych modeli.



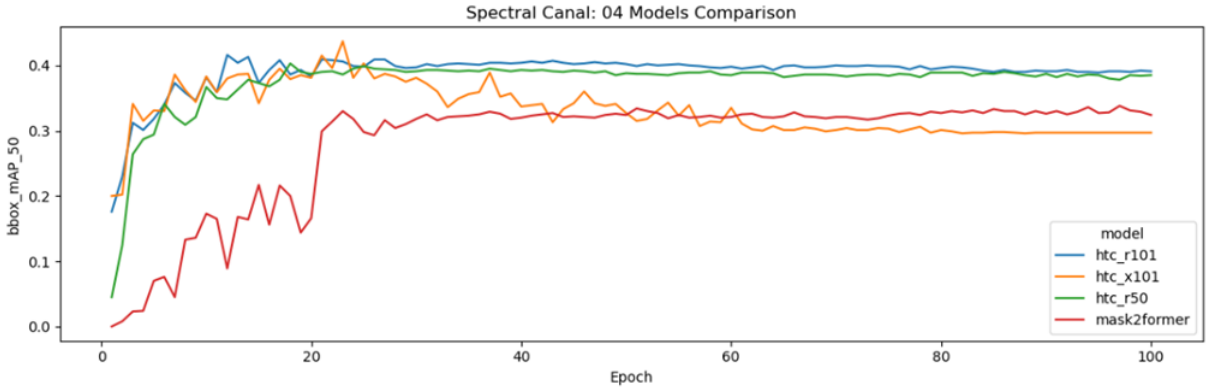
Rysunek 4.16. Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z drugiego kanału spektralnego.

Analizując wyniki dla kanału spektralnego nr 2 (rysunek 4.16) można zauważyć, że wykresy dla algorytmów HTC_r101 i HTC_r50 charakteryzują się szybką tendencją wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 wynoszącym około 0,4, co świadczy o ich stabilności. Model Mask2Former również osiąga podobny poziom stabilizacji (około 0,4), ale jego wzrost jest bardziej stopniowy i trwa do około 20 epok. Z kolei model HTC_x101 początkowo osiąga wartość bbox_mAP_50 równą około 0,4 w pierwszych 20 epokach, jednak w dalszych etapach treningu jego wyniki spadają i stabilizują się na poziomie około 0,3, co oznacza znaczną utratę skuteczności. Pod względem końcowych wyników, najwyższą wartość bbox_mAP_50 osiąga model HTC_r101, przewyższając pozostałe algorytmy. Modele HTC_r50 i Mask2Former uzyskują porównywalne wyniki końcowe (około 0,4), natomiast najniższą wartość stabilizacji wykazuje HTC_x101, który kończy trenowanie na poziomie około 0,3.

W wynikach dla kanału spektralnego nr 3 (rysunek 4.17) również można zauważyć, że algorytmy HTC_r101 i HTC_r50 wykazują szybki wzrost wyników w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 wynoszącym około 0,4, co wskazuje na ich stabilność i skuteczność w analizowanym zadaniu. Model Mask2Former charakteryzuje się bardziej stopniowym wzrostem, który trwa do około 20 epok. Po tym czasie stabilizuje się na poziomie bbox_mAP_50 równym około 0,3, co jest wynikiem niższym niż dla HTC_r101 i HTC_r50. Z kolei model HTC_x101 osiąga poziom bbox_mAP_50 wynoszący około 0,4 już w pierwszych 10 epokach, jednak w późniejszych etapach jego wyniki spadają i stabilizują się na poziomie około 0,3, podobnie jak w przypadku Mask2Former. Pod względem końcowych wyników, HTC_r101 i HTC_r50 osiągają najwyższą wartość bbox_mAP_50 (około 0,4), przewyższając pozostałe modele. Modele Mask2Former i HTC_x101 kończą trenowanie na niższym poziomie, stabilizując się na poziomie około 0,3.

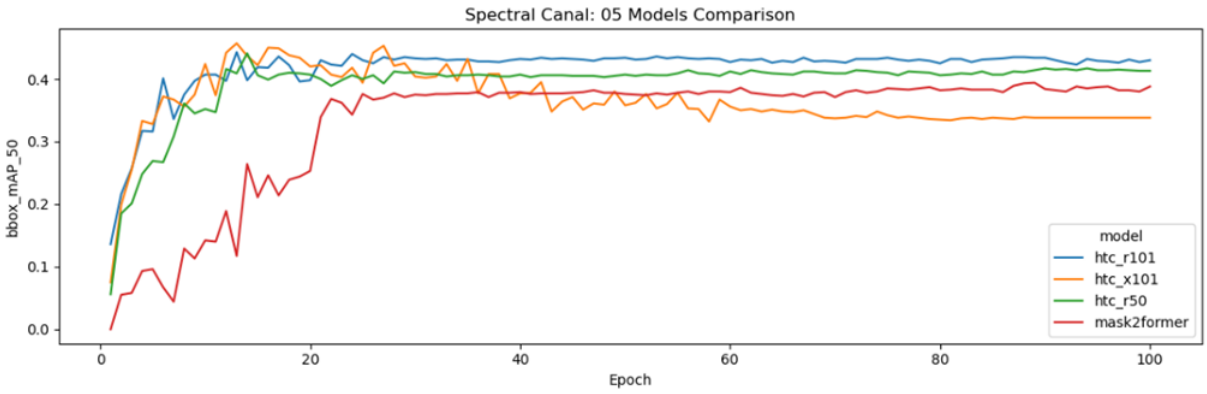


Rysunek 4.17. Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z trzeciego kanału spektralnego.



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

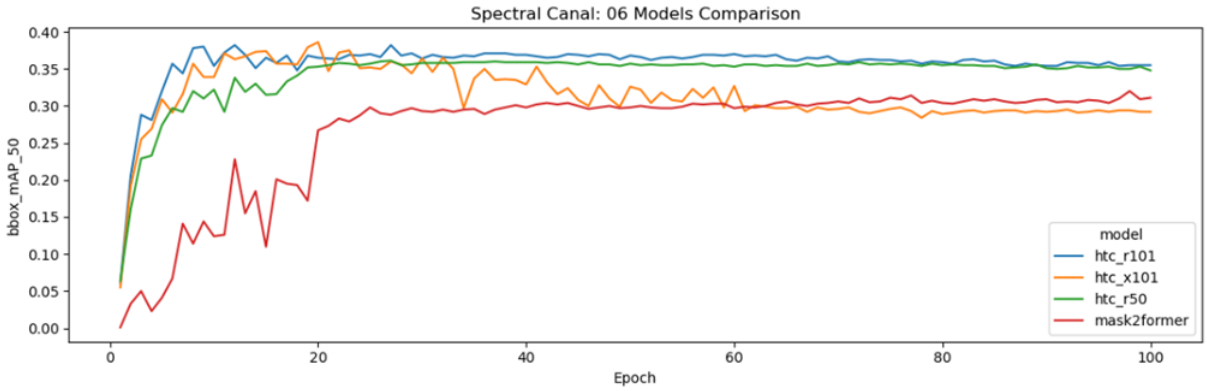
Przechodząc do analizy wyników dla kanału spektralnego nr 4 (rysunek 4.18) również można zauważyć, że wykresy dla algorytmów HTC_r101 i HTC_r50 wykazują szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 wynoszącym odpowiednio około 0,4. Ich przebieg jest stabilny i konsekwentny w dalszej części trenowania. Model Mask2Former charakteryzuje się bardziej stopniowym wzrostem w pierwszych 20 epokach, po czym stabilizuje się na poziomie bbox_mAP_50 równym około 0,3. Z kolei model HTC_x101 osiąga poziom bbox_mAP_50 wynoszący około 0,4 już w pierwszych 10 epokach, jednak później jego wyniki spadają, stabilizując się na poziomie około 0,3, podobnie jak w przypadku Mask2Former. Pod względem końcowych wyników, HTC_r101 osiąga najwyższą wartość bbox_mAP_50, przewyższając pozostałe modele. HTC_r50 uzyskuje nieco niższe, ale nadal konkurencyjne wyniki. Najniższe końcowe wartości osiągają modele Mask2Former i HTC_x101, które stabilizują się na poziomie około 0,3, z tym że HTC_x101 wykazuje wyraźny spadek po początkowym etapie trenowania.



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

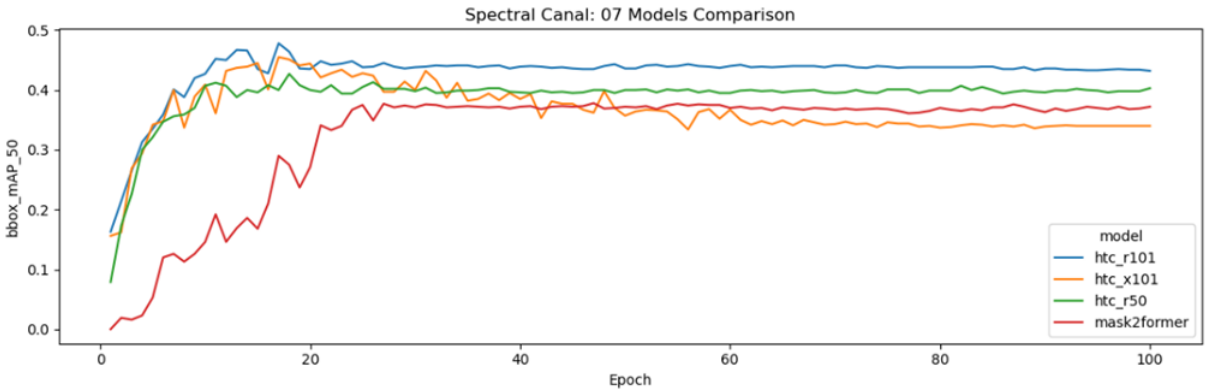
Poddając analizie wyniki dla kanału spektralnego nr 5 (rysunek 4.19) można zauważyć, iż wykresy dla algorytmów HTC_r101 i HTC_r50 charakteryzują się szybką tendencją wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 wynoszącym odpowiednio około 0,4. Model Mask2Former osiąga podobny poziom stabilizacji, lecz jego wzrost jest bardziej stopniowy i trwa do około 20 epok. W przypadku HTC_x101 widoczny jest dynamiczny wzrost w pierwszych 10 epokach, z osiągnięciem wartości bbox_mAP_50 bliskiej 0,4, jednak później następuje spadek wydajności. Model stabilizuje się na poziomie około 0,3 i utrzymuje tę wartość do końca treningu. Pod względem końcowych wyników, najwyższą wartość bbox_mAP_50 uzyskuje HTC_r101, przewyższając inne modele. Na drugim miejscu znajduje się HTC_r50, którego wyniki są zbliżone do

HTC_r101. Modele Mask2Former i HTC_x101 kończą trenowanie na najniższym poziomie, przy czym HTC_x101 wykazuje wyraźny spadek po osiągnięciu początkowo wysokich wyników.



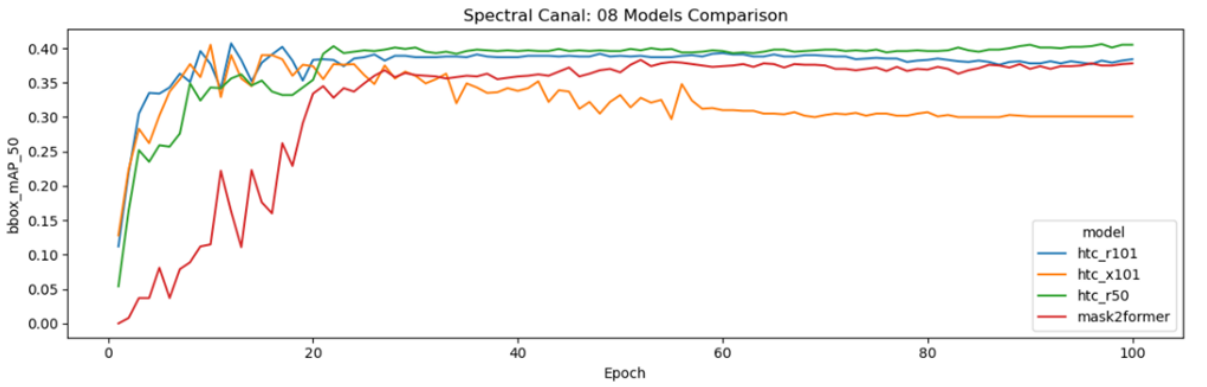
Rysunek 4.20. Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z szóstego kanału spektralnego.

Analizując wyniki dla kanału spektralnego nr 6 (rysunek 4.20) można zauważyć, iż wykresy dla algorytmów HTC_r101 i HTC_r50 charakteryzują się szybką tendencją wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 wynoszącym odpowiednio około 0,4 dla HTC_r101 i nieco poniżej 0,4 dla HTC_r50. Model Mask2Former wykazuje bardziej stopniowy wzrost, osiągając stabilizację dopiero po około 20 epokach na poziomie bbox_mAP_50 równym około 0,3. Natomiast HTC_x101 w początkowej fazie treningu (pierwsze 10 epok) osiąga wartość bbox_mAP_50 równą około 0,38, jednak w późniejszej fazie obserwuje się spadek wyników. Model stabilizuje się na poziomie około 0,3, podobnie jak Mask2Former. Pod względem końcowych wyników HTC_r101 uzyskuje najwyższą wartość bbox_mAP_50, przewyższając pozostałe modele. HTC_r50 zajmuje drugie miejsce z wynikami bliskimi 0,4, natomiast najniższe końcowe wartości osiągają modele Mask2Former i HTC_x101, które stabilizują się na poziomie około 0,3.



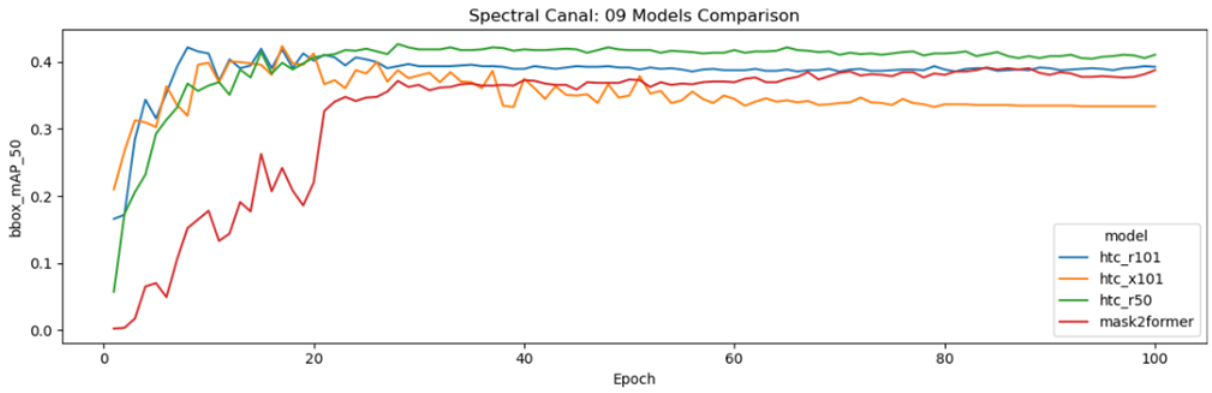
Rysunek 4.21. Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z siódmego kanału spektralnego.

Analizując wyniki dla kanału spektralnego nr 7 (rysunek 4.21) można zauważyć, iż wykresy dla algorytmów HTC_r101 i HTC_r50 charakteryzują się szybką tendencją wzrostową w pierwszych 10 epokach. Model HTC_r101 stabilizuje się na najwyższym poziomie bbox_mAP_50, wynoszącym około 0,45, natomiast HTC_r50 osiąga stabilny poziom wynoszący około 0,4. Algorytm Mask2Former wykazuje bardziej stopniowy wzrost, stabilizując się po 20 epokach na poziomie bbox_mAP_50 równym około 0,4, co czyni go porównywalnym z modelem HTC_r50. W przypadku algorytmu HTC_x101 początkowa faza trenowania jest dynamiczna i osiąga on wartość bbox_mAP_50 równą około 0,45 w pierwszych 10 epokach, jednak później wyniki spadają. Model stabilizuje się na poziomie około 0,3 i utrzymuje tę wartość do końca trenowania. Pod względem końcowych wyników, HTC_r101 uzyskuje najwyższą wartość bbox_mAP_50, przewyższając pozostałe modele. HTC_r50 oraz Mask2Former osiągają porównywalne wyniki, stabilizując się na poziomie około 0,4, natomiast najniższe końcowe wyniki uzyskuje HTC_x101, który kończy trenowanie na poziomie 0,3.



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

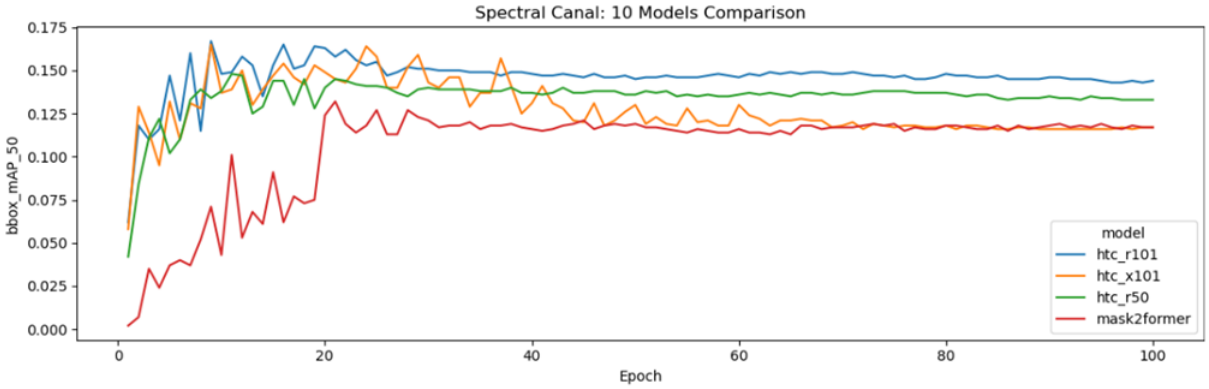
Analizując wyniki dla kanału spektralnego nr 8 (rysunek 4.22) można zauważyć, że wykresy dla algorytmów HTC_r101 i HTC_r50 charakteryzują się szybką tendencją wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 równym około 0,4, co wskazuje na ich skuteczność i stabilność w omawianym zadaniu. Algorytm Mask2Former wykazuje bardziej stopniowy wzrost, osiągając poziom bbox_mAP_50 około 0,4 dopiero po 20 epokach, gdzie stabilizuje się i utrzymuje tę wartość do końca treningu. W przypadku algorytmu HTC_x101 można zaobserwować dynamiczny wzrost w pierwszych 10 epokach, osiągając wartość bbox_mAP_50 równą około 0,4, jednak w kolejnych epokach następuje spadek wyników. Model stabilizuje się na niższym poziomie, wynoszącym około 0,3 i utrzymuje tę wartość do końca treningu. Pod względem końcowych wyników najlepszą skuteczność osiągają algorytmy HTC_r101, HTC_r50 i Mask2Former, które stabilizują się na poziomie 0,4. Algorytm HTC_x101, pomimo dobrych wyników w początkowej fazie trenowania, ostatecznie osiąga najniższą wartość końcową wynoszącą 0,3.



Rysunek 4.23. Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z dziewiątego kanału spektralnego.

Poddając analizie wyniki dla kanału spektralnego nr 9 (rysunek 4.23) widać, iż algorytmy HTC_r101 i HTC_r50 wykazują szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 równym około 0,4, co wskazuje na ich wysoką i stabilną skuteczność. Algorytm Mask2Former osiąga podobny poziom stabilizacji (około 0,4), jednak wzrost jego wydajności trwa dłużej, bo około 20 epok. W przypadku algorytmu HTC_x101 początkowa faza trenowania jest bardzo dynamiczna, osiągając poziom bbox_mAP_50 równy około 0,4 już po kilku epokach, po czym następuje zauważalny spadek. Po około 20 epokach stabilizuje się na niższym poziomie, wynoszącym około 0,3 i pozostaje na tym poziomie do końca treningu. Pod względem końcowych wyników najwyższe rezultaty

osiągają algorytmy HTC_r50, HTC_r101 i Mask2Former, które uzyskują podobny poziom skuteczności równy około 0,4. Algorytm HTC_x101, pomimo bardzo obiecujących wyników w początkowej fazie trenowania, osiąga najniższy końcowy wynik na poziomie około 0,3, co może wynikać z jego trudności w utrzymaniu efektywności podczas dalszego procesu uczenia.



Rysunek 4.24. Porównanie przebiegu trenowania poszczególnych modeli na zbiorze danych z dziesiątego kanału spektralnego.

Analizując zaś wyniki dla kanału spektralnego nr 10 (rysunek 4.24) można zaobserwować, iż algorytmy HTC_r101 i HTC_r50 wykazują szybką tendencję wzrostową w pierwszych 10 epokach, po czym stabilizują się na poziomie bbox_mAP_50 równym około 0,15, co wskazuje na ich dobrą stabilność i skuteczność w detekcji. W przypadku algorytmu Mask2Former można zauważyć, że w pierwszych 20 epokach wykres rośnie dynamicznie, osiągając wartość bbox_mAP_50 na poziomie około 0,125, po czym stabilizuje się na tym poziomie do końca trenowania. Algorytm HTC_x101 charakteryzuje się nietypowym przebiegiem, gdyż osiąga najwyższą wartość skokową bbox_mAP_50 w pierwszych 10 epokach równą około 0,4, lecz w kolejnych epokach obserwuje się znaczny spadek. W końcowej fazie trenowania jego wydajność stabilizuje się na poziomie bbox_mAP_50 równym około 0,125, co plasuje go na poziomie porównywalnym z Mask2Former. Pod względem stabilności i końcowych wyników, algorytm HTC_r101 osiąga najlepsze rezultaty, ustępując niewiele algorytmowi HTC_r50. Algorytmy Mask2Former i HTC_x101 osiągają podobne wartości w końcowej fazie trenowania, lecz HTC_x101 wykazuje nieefektywność w utrzymaniu początkowych wyników, co znacząco wpływa na jego końcową ocenę.

Jak widać na podstawie krzywych uczenia, w przeważającej liczbie kanałów spektralnych najsłabsze wyniki końcowe można zaobserwować w przypadku modelu

wytrenowanego z użyciem algorytmu HTC_x101, czyli tego, który osiągnął najlepsze wyniki w przypadku trenowania na powiększonym 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 1 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 zaproponowanie nowatorskiego podejścia do automatyzacji precyzyjnego monitorowania wzrostu kukurydzy, które wykorzystuje zaawansowane techniki analizy obrazów oraz sztucznej inteligencji, polegające na zastosowaniu głębokich sieci neuronowych do analizy zarówno obrazów RGB, jak i obrazów wielospektralnych, w tym z dodatkowym wykorzystaniem klasyfikatora głosującego. Przeprowadzone w tym kierunku badania miały 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 iż 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, 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 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 pod kątem skuteczności detekcji i klasyfikacji zarówno w zależności od użytego zbioru obrazów treningowych, jak i w zależności od zastosowanego algorytmu.

Na bazie obserwacji można zauważyć, iż lepsze wyniki osiągnął klasyfikator głosujący, w którym modele wytrenowane na pojedynczych kanałach spektralnych głosowały nad wynikiem dotyczącym wyboru konkretnej fazy rozwojowej wykrytych roślin kukurydzy.

Pojedyncze modele wytrenowane na danych spektralnych mają gorszą jakość, niż model wytrenowany na danych RGB, ponieważ zostały wytrenowane na jednokanałowych obrazach, podczas gdy model RGB został wytrenowane na trójkanałowych danych. Zastosowanie techniki głosowania poprawiło jakość sumarycznej predykcji dla danych wielospektralnych i pozwoliło na uzyskanie lepszych wyników, niż w przypadku modelu trenowanego na danych RGB.

Dodatkowymi cennymi wnioskami z przeprowadzonych badań jest obserwacja zachowań różnych algorytmów i uzyskiwanych przez nie wyników w zależności od zastosowanego zbioru treningowego. Spośród badanych algorytmów trenowanych na referencyjnym zbiorze danych RGB najlepsze wyniki uzyskał algorytm Mask2Former, zaś podczas trenowania tychże algorytmów na zwiększonym zbiorze danych RGB najlepsze wyniki osiągnął algorytm HTC_r101. Z kolei spośród wszystkich badanych algorytmów trenowanych na zbiorze danych wielospektralnych i przy zastosowaniu klasyfikatora głosującego najlepsze wyniki uzyskał algorytm HTC_r50, a tuż za nim z minimalną różnicą był algorytm HTC_x101. Porównując zaś wszystkie podejścia, czyli: zastosowanie mniejszego zbioru obrazów RGB, zastosowanie referencyjnego do niego zbioru obrazów wielospektralnych z kamery MicaSense RedEdge MX Dual z użyciem techniki klasyfikatora głosującego oraz zastosowanie zwiększonego zbioru obrazów RGB, można wyciągnąć wniosek iż najlepsze wyniki uzyskał algorytm HTC_r50 trenowany na danych wielospektralnych i przy zastosowaniu klasyfikatora głosującego.

Porównując zaś wyniki poszczególnych algorytmów w zależności od zastosowanego zbioru treningowego (mowa tutaj o mniejszym zbiorze RGB i analogicznym do niego zbiorze obrazów wielospektralnych, aby porównanie było rzetelne) można zauważyć, że algorytmy HTC_r50, HTC_r101 oraz HTC_x101 uzyskują lepsze wyniki dla zbioru treningowego

składającego się z obrazów pochodzących z pojedynczych kanałów spektralnych i przy zastosowaniu klasyfikatora głosującego. Z kolei algorytm Mask2Former uzyskał wyższe wyniki przy trenowaniu na zbiorze obrazów RGB (przy czym w przypadku algorytmu Mask2Former różnica ta nie jest duża).

Analizując poszczególne pojedyncze kanały spektralne można zauważyć, że:

- Algorytm HTC_r101 osiągnął najlepsze wyniki spośród badanych algorytmów dla zbiorów treningowych odpowiadających następującym kanałom spektralnym: kanał 2, kanał 4, kanał 5, kanał 6, kanał 7, kanał 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: kanał 1, kanał 2, kanał 4, kanał 5, kanał 6, kanał 7, kanał 8, kanał 9, kanał 10.
- Algorytm HTC_r50 osiągnął najlepsze wyniki spośród badanych algorytmów dla następujących zbiorów treningowych odpowiadających poszczególnym kanałom spektralnym: kanał 3, kanał 8, kanał 9 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 nr 1. Algorytm ten osiągnął zaś najgorsze wyniki spośród analizowanych algorytmów dla zbioru treningowego obrazów RGB oraz dla zbiorów treningowych odpowiadających następującym kanałom spektralnym: kanał 3, kanał 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 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 kanał 7, dla algorytmu HTC_x101 najlepszymi kanałami okazały się kanał 5 i kanał 7, dla algorytmu HTC_r50 najlepszym kanałem okazał się kanał 5, zaś dla algorytmu Mask2Former najlepszym kanałem okazał się kanał 2.

Zaprezentowane rozwiązanie jest unikatowe w kontekście automatycznego i precyzyjnego monitorowania faz rozwoju kukurydzy przy użyciu obrazów wielospektralnych i obrazów RGB. Dzięki prezentowanej metodzie, możliwe jest dokładne śledzenie stanu roślin i podejmowanie trafnych decyzji w rolnictwie precyzyjnym, gdzie dokładna identyfikacja etapów rozwoju roślin może pomóc w optymalizacji nawadniania, nawożenia oraz stosowania środków 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 RGB oraz odpowiedniej jednostki obliczeniowej. Jest to rozwiązanie tańsze, niż użycie kamery wielospektralnej typu MicaSense RedEdge MX Dual, lecz przy nieco gorszych wynikach klasyfikacji w porównaniu do trenowania na analogicznym zbiorze obrazów wielospektralnych pochodzących z tejże kamery wielospektralnej. Jak pokazały badania, powiększenie zbioru treningowego obrazów RGB pozwoliło na polepszenie wyników i uzyskanie lepszych efektów, niż przy wykorzystaniu mniejszego zbioru danych RGB. Dzięki opracowanemu rozwiązaniu rolnicy i przedsiębiorcy mogą na bieżąco monitorować etapy rozwoju kukurydzy i podejmować odpowiednie działania w czasie rzeczywistym, co może przyczynić się do zwiększenia efektywności i wydajności upraw.

5. Detekcja i klasyfikacja faz rozwoju kukurydzy w skali BBCH przy wykorzystaniu obrazowania RGB i obrazowania z kamery wielospektralnej MapIR Survey3 Red+Green+NIR

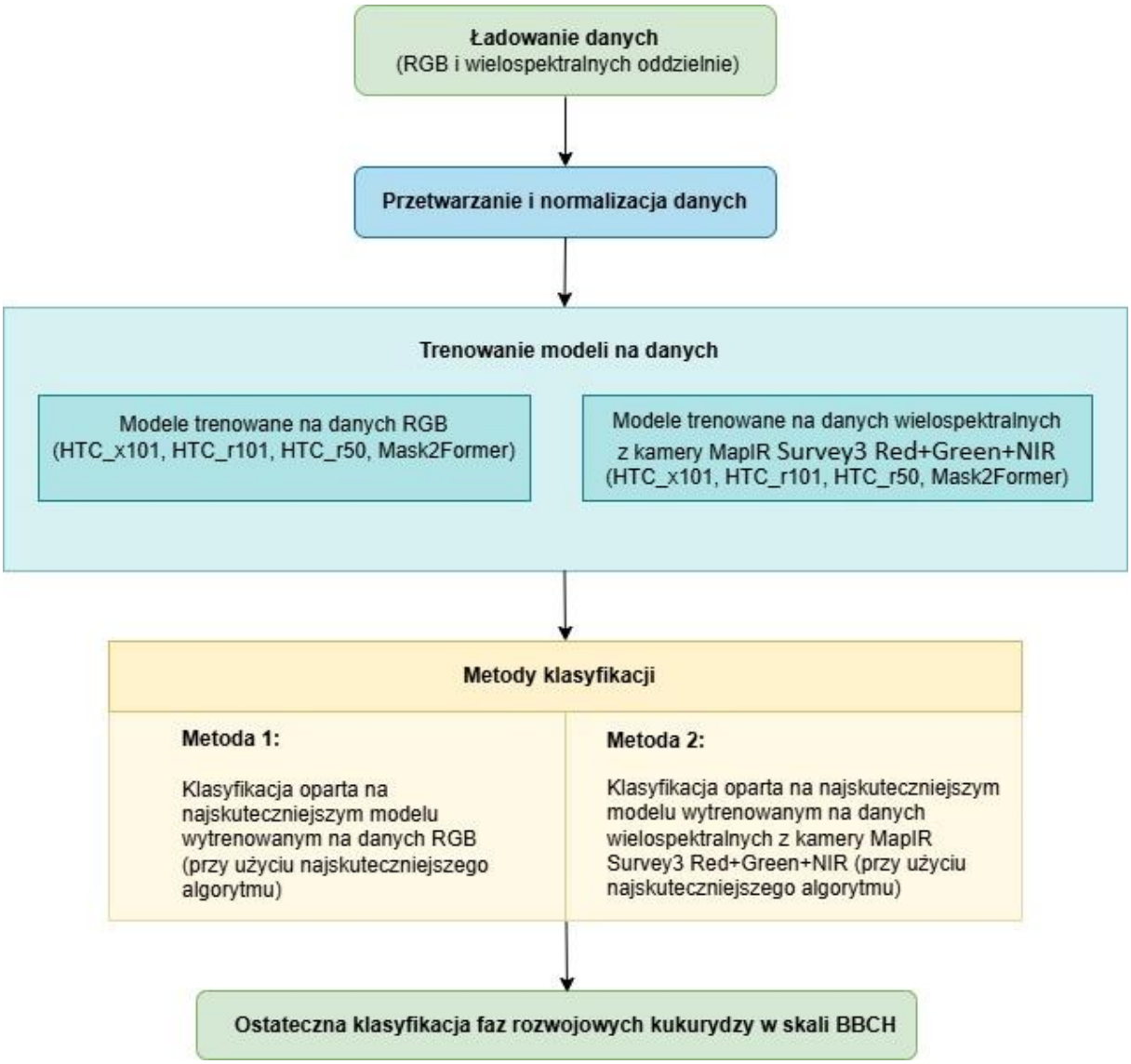
Niniejszy rozdział skupia się na przedstawieniu wyników uzyskanych w ramach zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH z zastosowaniem obrazowania RGB, obrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR oraz następujących architektur uczenia głębokiego: HTC_r101, HTC_r50, HTC_x101 oraz Mask2Former.

Do realizacji prac została wykorzystana kolejna kamera wielospektralna w postaci kamery 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 MX Dual, gdzie ta sama roślina rejestrowana przez różne obiektywy kamery wielospektralnej miała inny kształt w każdym z rejestrowanych kanałów spektralnych. Pozyskiwanie obrazów z kamery MapIR Survey3 Red+Green+NIR 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 Survey3 Red+Green+NIR, aby zachować identyczne warunki badawcze oraz aby jedyną zmienną podczas eksperymentu był rodzaj kamery, a co za tym idzie 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 z użyciem obrazowania RGB oraz obrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR. 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 Survey3 Red+Green+NIR.

Schemat przedstawia dwa główne podejścia zastosowane w opracowanym systemie: klasyfikacja na obrazach RGB oraz klasyfikacja na obrazach pochodzących z kamery wielospektralnej MapIR Survey3 Red+Green+NIR. Rozwiązania te zaproponowano, aby sprawdzić i porównać wyniki otrzymane przy zastosowaniu standardowego obrazowania RGB oraz zobrazowania pozyskiwanego za pomocą kamery wielospektralnej MapIR Survey3 Red+Green+NIR, 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 zdobytej w ten sposób wiedzy również 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 użyciem obrazów wielospektralnych z kamery MapIR Survey3 Red+Green+NIR. Finalna klasyfikacja jest wynikiem wyboru najskuteczniejszej metody opartej o jedno ze zobrazowań: RGB lub wielospektralne z kamery MapIR Survey3 Red+Green+NIR oraz o algorytm, który osiągnął najlepsze wyniki spośród analizowanych algorytmów.

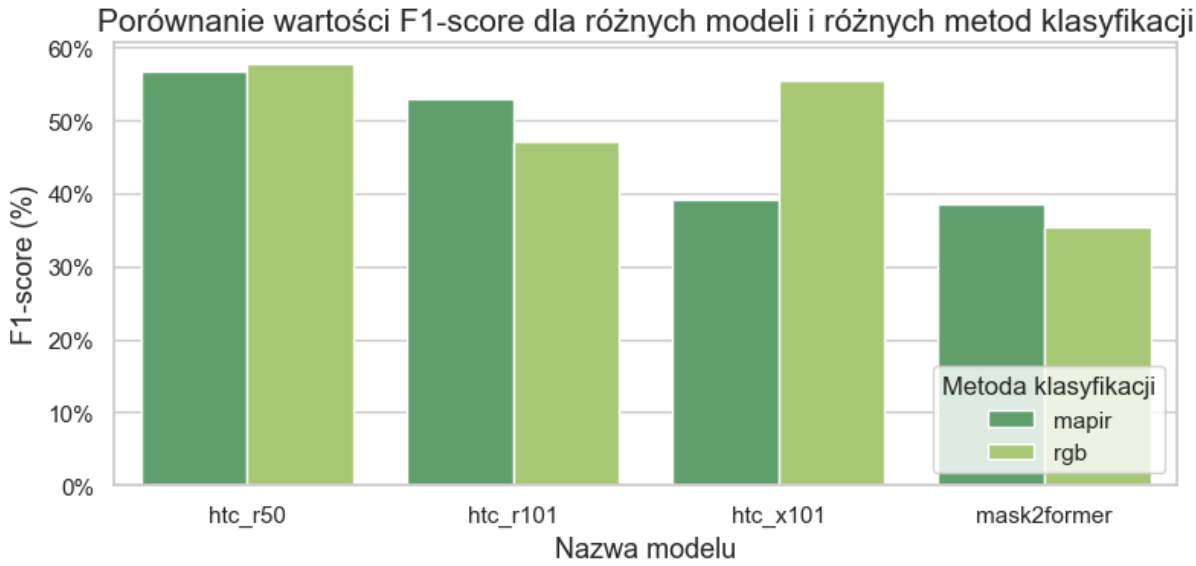
5.2. Wyniki uzyskane dla danych wielospektralnych i RGB

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, dla porównania wykorzystano metryki accuracy, precision, recall oraz F1-score. 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 wielospketralnych z kamery MapIR Survey3 Red+Green+NIR 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.

		accuracy	precision	recall	F1-score
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 Survey3 Red+Green+NIR.

Rysunek 5.2 przedstawia z kolei wykres wartości parametru F1-score dla poszczególnych algorytmów oraz analizowanych typów zobrazowania: RGB oraz zobrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR.



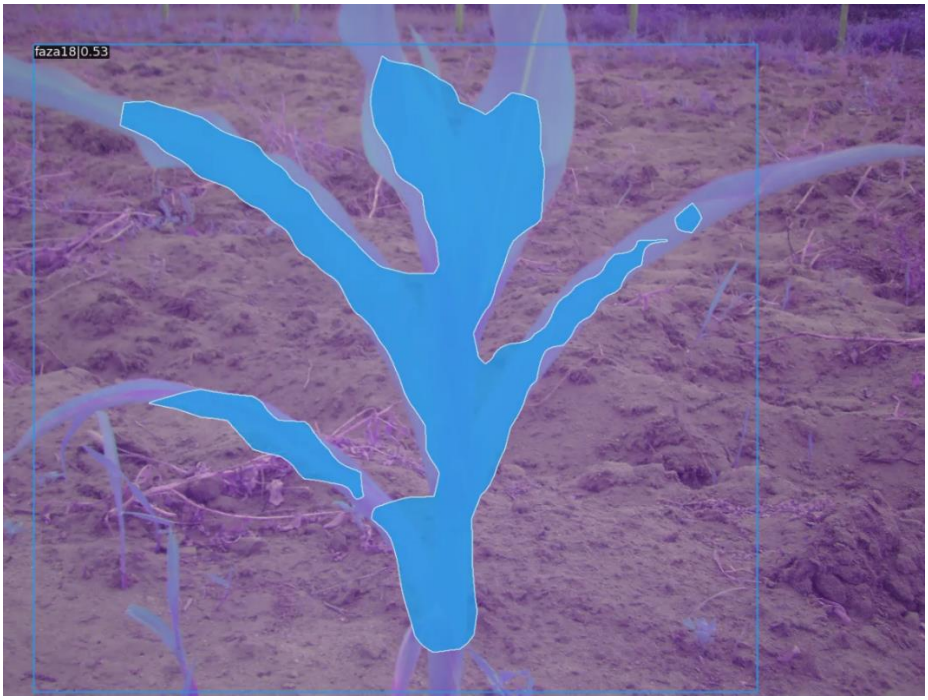
Rysunek 5.2. Wykres przedstawiający wartości parametru F1-score dla poszczególnych algorytmów oraz analizowanych typów zobrazowania: RGB oraz wielospektralnego z kamery MapIR Survey3 Red+Green+NIR.

Na rysunkach 5.3. - 5.5. zaprezentowano wyniki działania modelu HTC_r50 wytrenowanego na obrazach RGB pochodzących z późniejszego sezonu wegetacyjnego. Z kolei na rysunkach 5.6. – 5.8. zaprezentowano wyniki działania modelu HTC_r50 wytrenowanego na danych wielospektralnych z kamery MapIR Survey3 Red+Green+NIR pochodzących z tego samego sezonu wegetacyjnego.

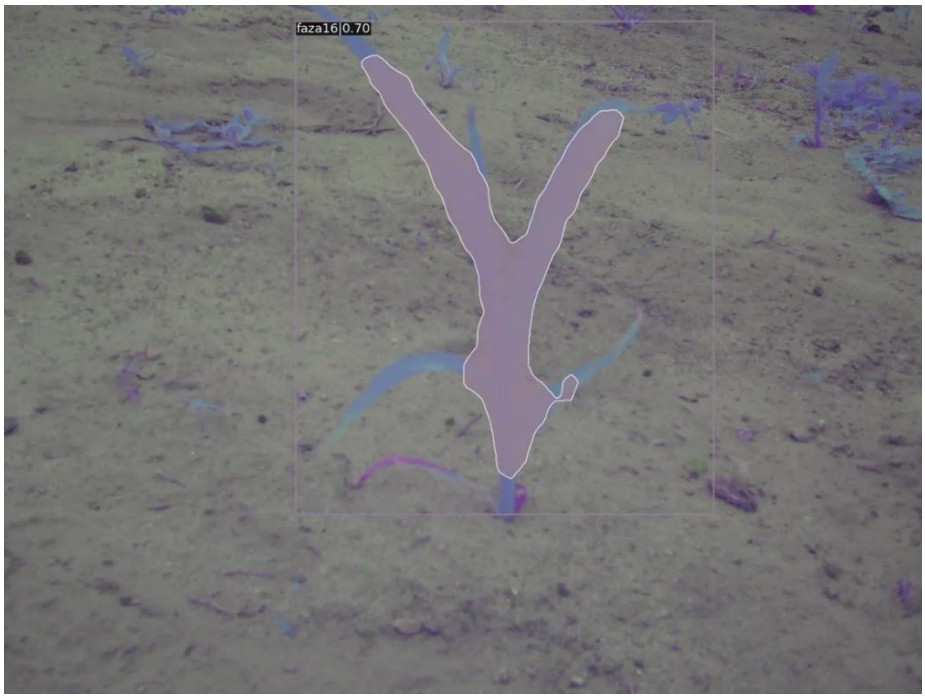
Rysunek 5.3. Wynik działania modelu HTC_r50 wytrenowanego na obrazach RGB.

Rysunek 5.4. Wynik działania modelu HTC_r50 wytrenowanego na obrazach RGB.

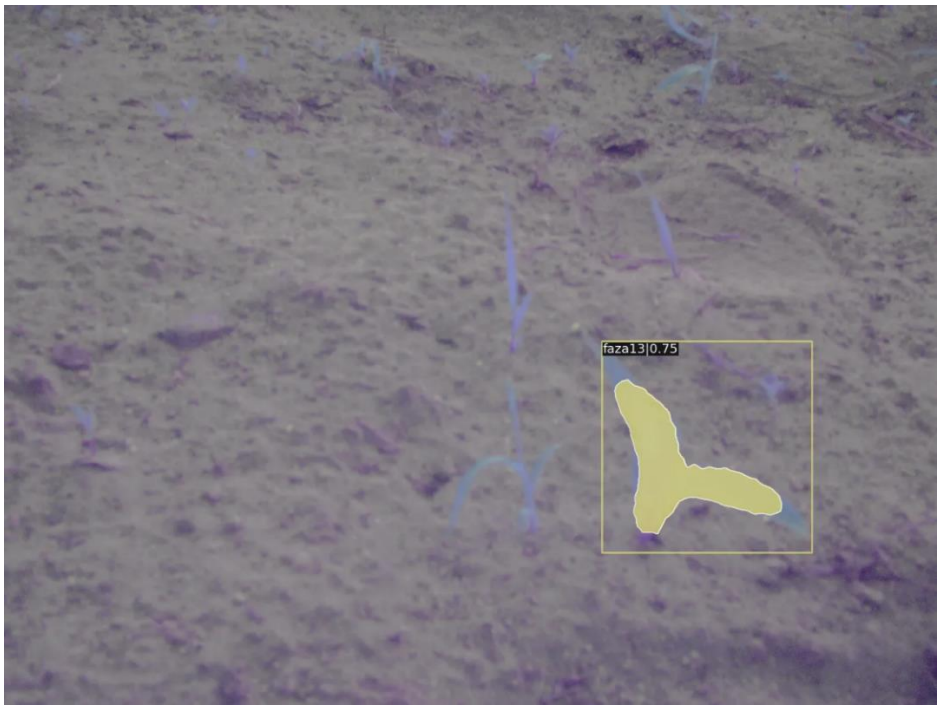
Rysunek 5.5. Wynik działania modelu HTC_r50 wytrenowanego na obrazach RGB.



Rysunek 5.6. Wynik działania modelu HTC_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR Survey3 Red+Green+NIR.



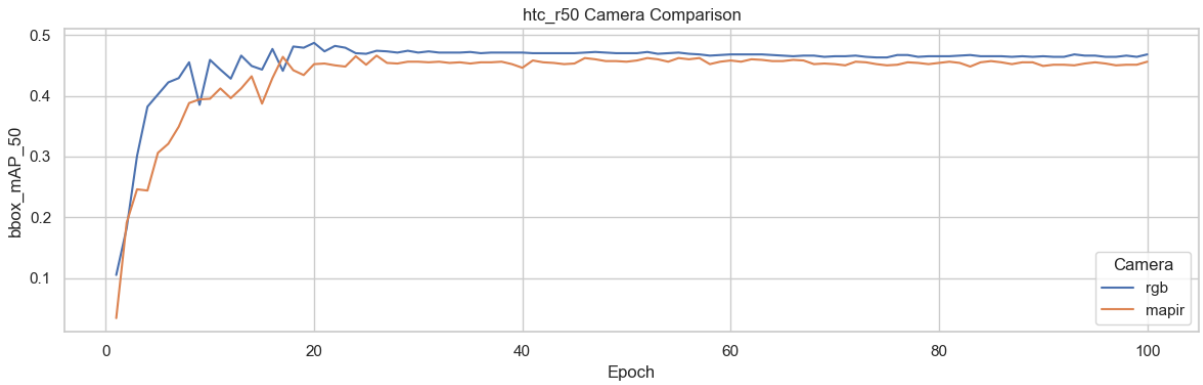
Rysunek 5.7. Wynik działania modelu HTC_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR Survey3 Red+Green+NIR.



Rysunek 5.8. Wynik działania modelu HTC_r50 wytrenowanego na obrazach wielospektralnych z kamery MapIR Survey3 Red+Green+NIR.

5.3. Krzywe uczenia dla danych wielospektralnych i RGB

W niniejszym rozdziale zaprezentowane zostały krzywe uczenia zarejestrowane podczas trenowania poszczególnych modeli dla dwóch typów zobrazowania: RGB i zobrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR.



Rysunek 5.9. Przebieg uczenia modelu opartego o algorytm HTC_r50 dla zobrazowania RGB i zobrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR.

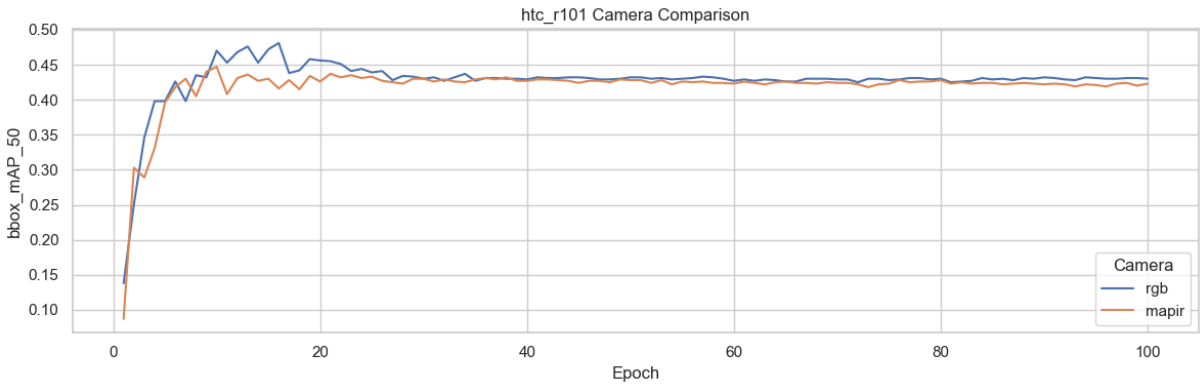
Rysunek 5.9. przedstawia porównanie wyników mAP (bbox_mAP_50) dla modelu HTC_r50 trenowanego na danych RGB oraz na danych wielospektralnych z kamery MapIR Survey3 Red+Green+NIR. Na obu krzywych można zaobserwować szybki wzrost wartości mAP w początkowych etapach treningu, co wskazuje na to, że model 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 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 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 Survey3 Red+Green+NIR, 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 analizowanego zadania detekcji obiektów. Obrazy z kamery MapIR Survey3 Red+Green+NIR, choć stabilne, wykazują mniejszą wydajność, co może wynikać z ograniczeń spektralnych tychże danych lub większej specyficzności informacji, jakie zawierają. Pomimo różnic, obie krzywe pozostają stosunkowo stabilne po osiągnięciu punktu zbieżności, co wskazuje na skuteczne uogólnianie przez model, ale bez oznak przeuczenia. Brak znaczących fluktuacji po osiągnięciu optymalnej wartości mAP sugeruje, że model generalizuje dobrze, a stabilność krzywych świadczy o odpowiednio przeprowadzonym procesie trenowania.

Podsumowując, model HTC_r50 wykazał się wysoką stabilnością i skutecznością w klasyfikacji obiektów na podstawie obu rodzajów danych wejściowych. Dane RGB pozwoliły osiągnąć nieco lepsze wyniki, jednak obrazy wielospektralne z kamery MapIR Survey3 Red+Green+NIR również zapewniły solidne wyniki bez oznak przeuczenia. Stabilność krzywych i brak spadku wydajności w kolejnych epokach potwierdzają dobrze przeprowadzony proces trenowania oraz zdolność modelu do generalizacji.

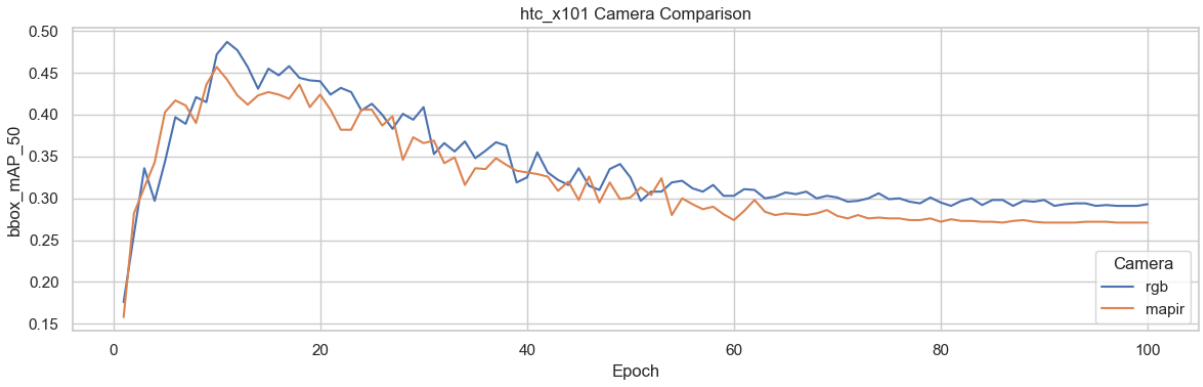


Rysunek 5.10. Przebieg uczenia modelu opartego o algorytm HTC_r101 dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR.

Rysunek 5.10. przedstawia porównanie wyników mAP (bbox_mAP_50) dla modelu HTC_r101 trenowanego na obrazach RGB oraz obrazach wielospektralnych z kamery MapIR Survey3 Red+Green+NIR. Oba zestawy danych prowadzą do podobnych wyników, jednak istnieją subtelne różnice w dynamice procesu uczenia.

Na początku treningu (pierwsze 10 - 15 epok) widać gwałtowny wzrost wartości mAP dla obu typów danych. Krzywa dla danych RGB w pierwszych epokach wzrasta szybciej i osiąga nieco wyższą wartość maksymalną (około 0,47) w porównaniu do danych z kamery MapIR (około 0,43). Szybka konwergencja w początkowych etapach sugeruje, że model efektywnie uczy się kluczowych cech z danych wejściowych. 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. Stabilność tychże wyników świadczy o osiągnięciu punktu zbieżności i braku dalszej poprawy wydajności modelu, co jest charakterystyczne dla dobrze przeprowadzonego procesu uczenia. 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 wyższej wartości mAP. 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 przeuczenia, w postaci spadków wydajności, świadczy o skutecznie dobranych hiperparametrach.

Podsumowując, model HTC_r101 wykazuje stabilne wyniki i wysoką skuteczność w detekcji obiektów na podstawie danych RGB i MapIR, przy czym obrazy RGB dostarczają nieco wyższej jakości informacji, co przekłada się na lepsze wyniki. Pomimo tych różnic, obie krzywe sugerują, że model skutecznie generalizuje, co świadczy o efektywności zastosowanego podejścia.



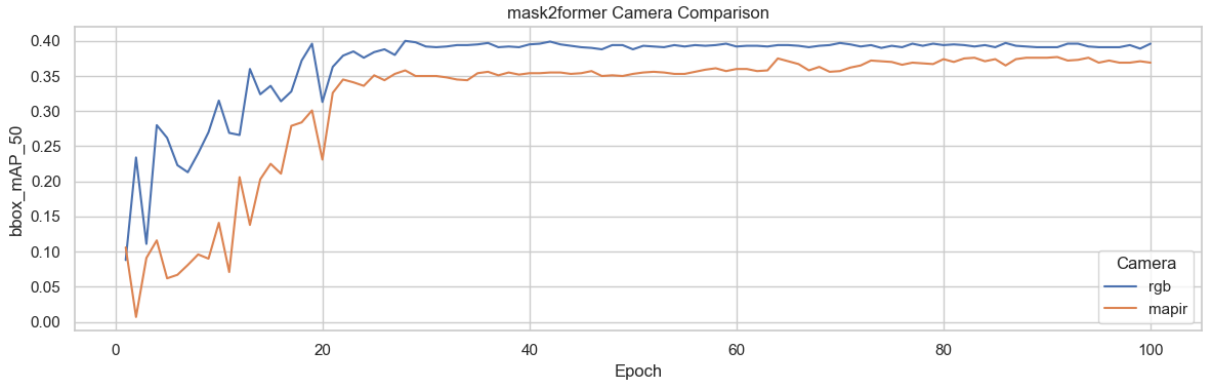
Rysunek 5.11. Przebieg uczenia modelu opartego o algorytm HTC_x101 dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR.

Rysunek 5.11. przedstawia porównanie wyników mAP (bbox_mAP_50) dla modelu HTC_x101 trenowanego na danych RGB oraz na danych z kamery MapIR Survey3

Red+Green+NIR. Analiza krzywych wskazuje na różnice w dynamice uczenia się modelu oraz trudności w utrzymaniu wysokiej wydajności w późniejszych fazach treningu.

Na początkowym etapie treningu (pierwsze kilkanaście epok) widoczny jest szybki wzrost wartości mAP dla obu rodzajów danych. Maksymalna wartość mAP osiąga około 0,45 dla danych RGB oraz około 0,40 dla MapIR, co następuje po około 20 epokach. Szybka konwergencja wskazuje, że model skutecznie przyswaja kluczowe cechy z danych, jednak już na tym etapie widoczna jest przewaga obrazów RGB. 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. Fluktuacje widoczne na krzywych mAP dla obu typów danych w środkowej fazie treningu mogą świadczyć o niestabilności modelu w zakresie rozpoznawania cech, ale stabilizacja pod koniec treningu, choć na niższym poziomie niż w początkowej fazie, sugeruje, że model zdołał częściowo uogólnić nauczone cechy.

Porównując oba zestawy danych, obrazy RGB pozwoliły osiągnąć lepsze wyniki niż dane MapIR. Może to wynikać z większej różnorodności i bogactwa informacji dostarczanych przez obrazy RGB, które są łatwiejsze do przetwarzania przez model w porównaniu do bardziej specyficznych danych z kamery MapIR.

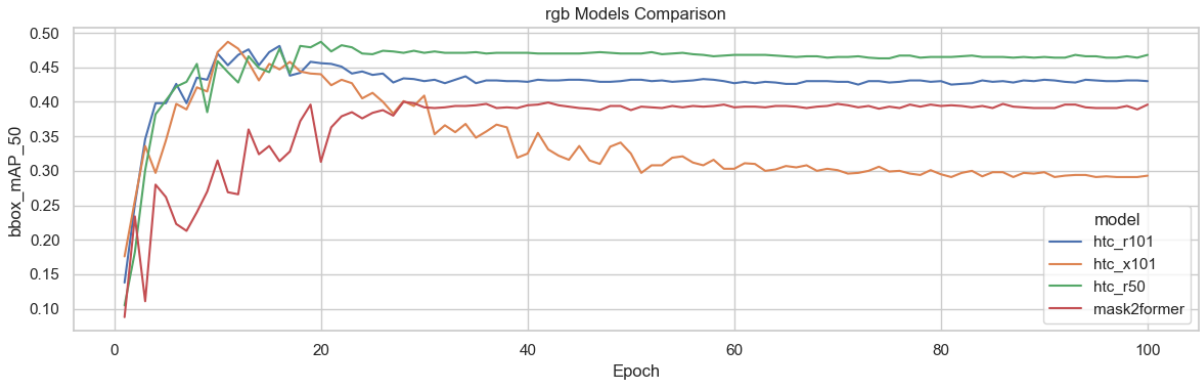


Rysunek 5.12. Przebieg uczenia modelu opartego o algorytm Mask2Former dla zobrazowania RGB oraz dla zobrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR.

Rysunek 5.12 przedstawia porównanie wyników mAP (bbox_mAP_50) dla modelu Mask2Former trenowanego na obrazach RGB i obrazach z kamery MapIR Survey3 Red+Green+NIR. 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. Ponadto, brak wyraźnych spadków mAP po osiągnięciu szczytowych wartości sugeruje, że model nie ulega przetrenowaniu. W przypadku obrazów z kamery MapIR Survey3 Red+Green+NIR widać 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 specyficznych cech z analizowanych danych.

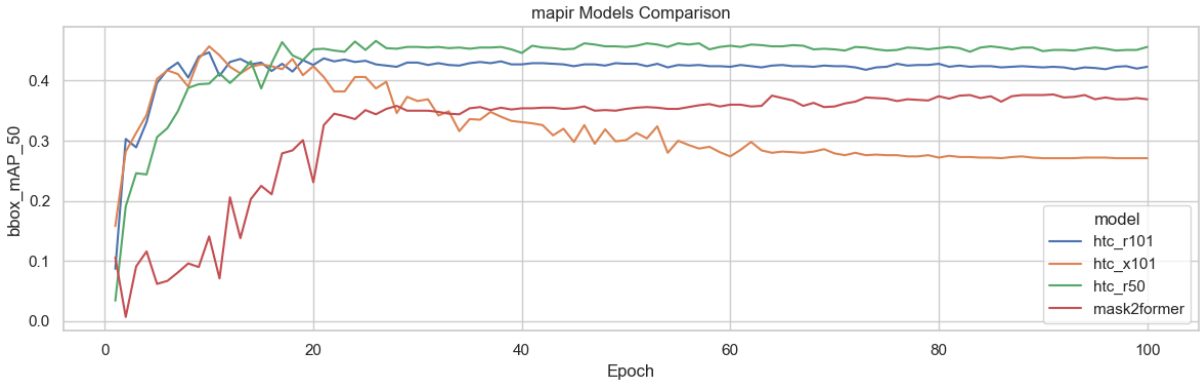
Porównując oba zestawy danych, model wyraźnie lepiej radzi sobie z obrazami RGB, osiągając wyższe wartości mAP oraz stabilniejszą krzywą treningu. Wyniki dla MapIR są niższe, co może wynikać z ograniczonej różnorodności informacji w obrazach MapIR w porównaniu do pełniejszego spektrum dostarczanego przez dane RGB. 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 w analizowanym zadaniu.



Rysunek 5.13. Przebieg uczenia modeli opartych o algorytmy HTC_r50, HTC_r101, HTC_x101 oraz Mask2Former dla zobrazowania RGB.

Rysunek 5.13 przedstawia z kolei porównanie wyników mAP (bbox_mAP_50) dla czterech modeli trenowanych na danych RGB: HTC_r50, HTC_r101, HTC_x101 oraz Mask2Former. Wyniki pozwalają na ocenę skuteczności poszczególnych algorytmów w analizie obrazów RGB oraz ich zdolności do generalizacji.

Model HTC_r50 osiąga najwyższe wartości mAP, stabilizując się na poziomie około 0,45. Jego krzywa charakteryzuje się szybkim wzrostem w początkowej fazie treningu oraz wysoką stabilnością w późniejszych epokach, co świadczy o efektywności w przyswajaniu kluczowych cech i dobrym uogólnianiu. HTC_r101 uzyskuje wyniki zbliżone do HTC_r50, jednak nieco niższe, stabilizując się na poziomie około 0,43 mAP. Wynik ten nadal wskazuje na wysoką skuteczność modelu, choć jest on nieco mniej wydajny w porównaniu do HTC_r50. Model HTC_x101 wypada wyraźnie gorzej, osiągając maksymalne wartości mAP na poziomie około 0,35, z zauważalnym spadkiem wydajności w późniejszych epokach. Krzywa HTC_x101 wskazuje na problemy z utrzymaniem stabilności, co może sugerować trudności w generalizacji. Najniższą wartość mAP uzyskuje model Mask2Former, który stabilizuje się na poziomie około 0,30. Pomimo szybkiego wzrostu w początkowej fazie treningu, jego wydajność pozostaje najniższa spośród wszystkich analizowanych modeli, a krzywa charakteryzuje się trudnościami w osiągnięciu pełnej stabilizacji.



Rysunek 5.14. Przebieg uczenia modeli opartych o algorytmy HTC_r50, HTC_r101, HTC_x101 oraz Mask2Former dla zobrazowania wielospektralnego z kamery MapIR Survey3 Red+Green+NIR.

Rysunek 5.14. przedstawia porównanie wyników mAP (bbox_mAP_50) dla modeli HTC_r50, HTC_r101, HTC_x101 oraz Mask2Former trenowanych na danych z kamery MapIR Survey3 Red+Green+NIR.

Model HTC_r50 ponownie osiąga najlepsze wyniki, stabilizując się na poziomie około 0,42 mAP, charakteryzuje się szybkim wzrostem w początkowych epokach oraz wysoką stabilnością w późniejszych fazach treningu, co wskazuje na efektywne przyswajanie cech z danych MapIR oraz dobrą zdolność do generalizacji. HTC_r101 również osiąga dobre wyniki, stabilizując się na poziomie około 0,40 mAP. Krzywa tego modelu pozostaje stabilna po początkowym gwałtownym wzroście, jednak jego skuteczność jest nieco niższa w porównaniu do HTC_r50. Model HTC_x101 wykazuje gorszą wydajność w stosunku do poprzednich modeli, osiągając maksymalne wartości mAP poniżej 0,35. Krzywa tego modelu wskazuje na trudności w utrzymaniu stabilności oraz możliwe problemy z generalizacją w trakcie treningu. Mask2Former osiąga najniższą wartość mAP, stabilizując się na poziomie około 0,30. Pomimo początkowego wzrostu, jego wydajność pozostaje najniższa spośród wszystkich analizowanych modeli, co sugeruje ograniczoną zdolność tego modelu do efektywnego wykorzystania danych wielospektralnych MapIR. Porównanie modeli pokazuje, że HTC_r50 jest najbardziej skuteczny na danych z kamery MapIR, osiągając najwyższą wartość mAP oraz stabilność w procesie uczenia. HTC_r101 również prezentuje dobrą wydajność, natomiast HTC_x101 i Mask2Former wypadają wyraźnie słabiej, co może wynikać z ograniczeń ich architektur w przetwarzaniu danych wielospektralnych analizowanych w ramach omawianego zadania.

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 RGB i obrazów z kamery wielospektralnej MapIR Survey3 Red+Green+NIR, można sformułować kilka kluczowych wniosków. Przedstawione wyniki jasno pokazują, że model HTC_r50 jest najbardziej efektywną architekturą zarówno w przypadku danych RGB, jak i danych wielospektralnych z kamery MapIR. Uzyskane wyniki mAP, F1-score oraz accuracy wyraźnie wskazują, że model ten najlepiej radzi sobie z zadaniem automatycznej oceny faz rozwoju kukurydzy.

Dane RGB okazały się bardziej informatywne i lepiej nadają się do trenowania modeli w kontekście klasyfikacji faz BBCH. Wszystkie testowane modele osiągają wyższe wyniki na obrazach RGB w porównaniu do obrazów z kamery MapIR Survey3 Red+Green+NIR. 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 spektrum informacji dostarczanych przez obrazy RGB. Dane z kamery 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 z kolei mniej efektywne w porównaniu do HTC_r50 i HTC_r101.

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 kukurydzy.

Wyniki eksperymentów jednoznacznie wskazują, że wybór odpowiedniego modelu ma kluczowe znaczenie w kontekście automatycznej klasyfikacji faz rozwoju kukurydzy w skali BBCH. Model HTC_r50 okazał się najbardziej efektywny w obu analizowanych przypadkach, jednak jego przewaga jest szczególnie widoczna na obrazach RGB. W analizowanym zadaniu model HTC_r50 w połączeniu z danymi RGB okazuje się zatem najlepszym rozwiązaniem.

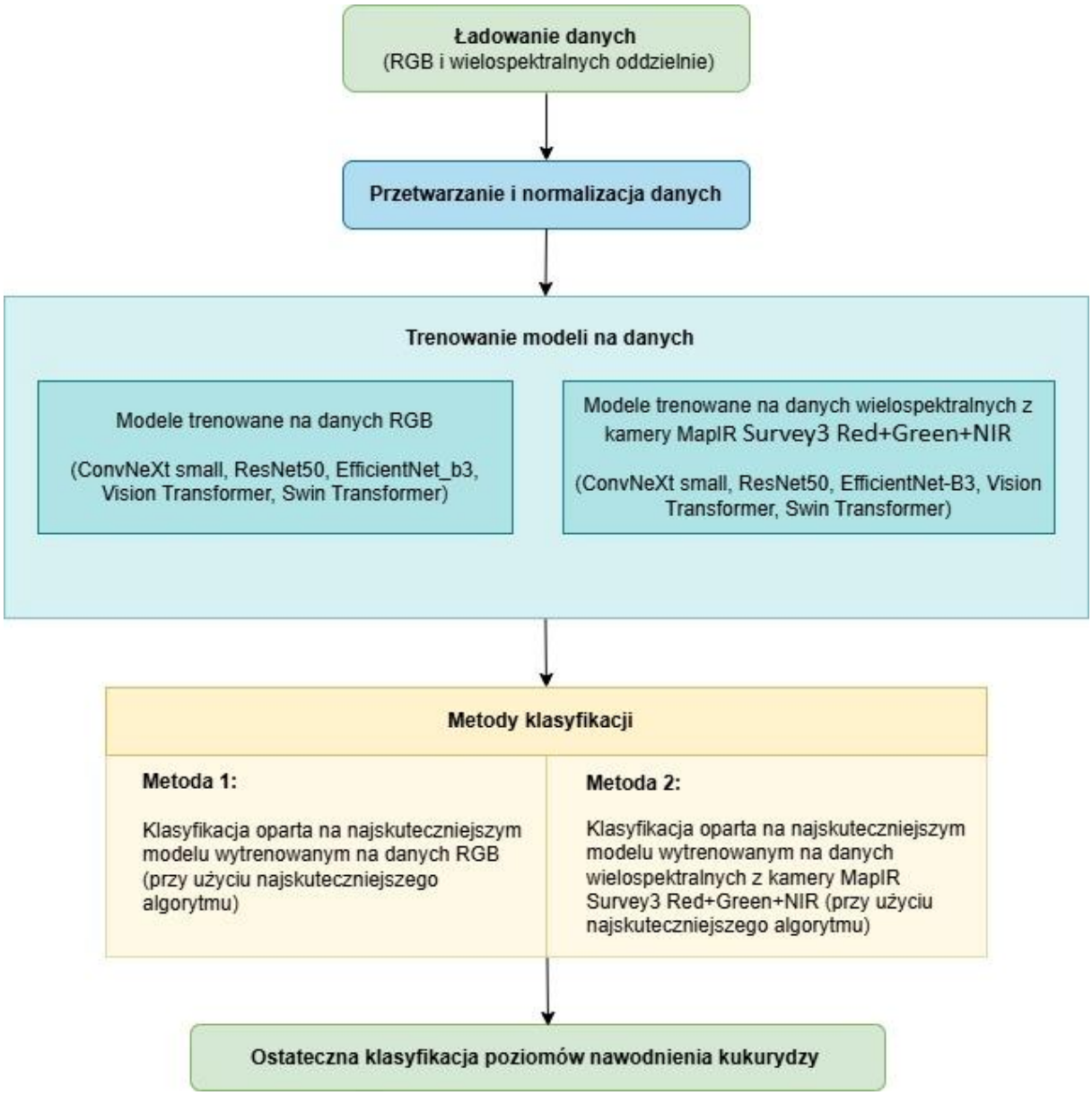
6. Detekcja i klasyfikacja poziomów nawodnienia kukurydzy

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: *mały, średni* i *duży poziom nawodnienia*, gdzie *mały poziom nawodnienia* oznaczał wystawienie roślin na warunki znacznie ograniczonej dostępności wody, wręcz w warunkach suszy fizjologicznej, *średni poziom nawodnienia* wiązał się z umiarkowanym dostępem roślin do wody, a *duży poziom nawodnienia* wiązał się z utrzymywaniem stałego optymalnego poziomu wilgotności. Badania prowadzono w warunkach glebowych charakterystycznych dla obszaru centralnej Polski, na terenie Województwa Mazowieckiego. 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 częściach tego rozdziału zaprezentowany zostanie schemat zaproponowanego rozwiązania, wyniki uzyskane dla danych wielospektralnych z kamery MapIR Survey3 Red+Green+NIR, wyniki uzyskane 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

Schemat widoczny na rysunku 6.1. przedstawia dwa główne podejścia zastosowane w opracowanym systemie: klasyfikacja z użyciem obrazów RGB oraz klasyfikacja z użyciem obrazów wielospektralnych z kamery MapIR Survey3 Red+Green+NIR. 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 architektur uczenia głębokiego. Prace te miały charakter badawczo-wdrożeniowy z potencjałem na wykorzystanie zdobytej w ten sposób wiedzy również w innych zadaniach.



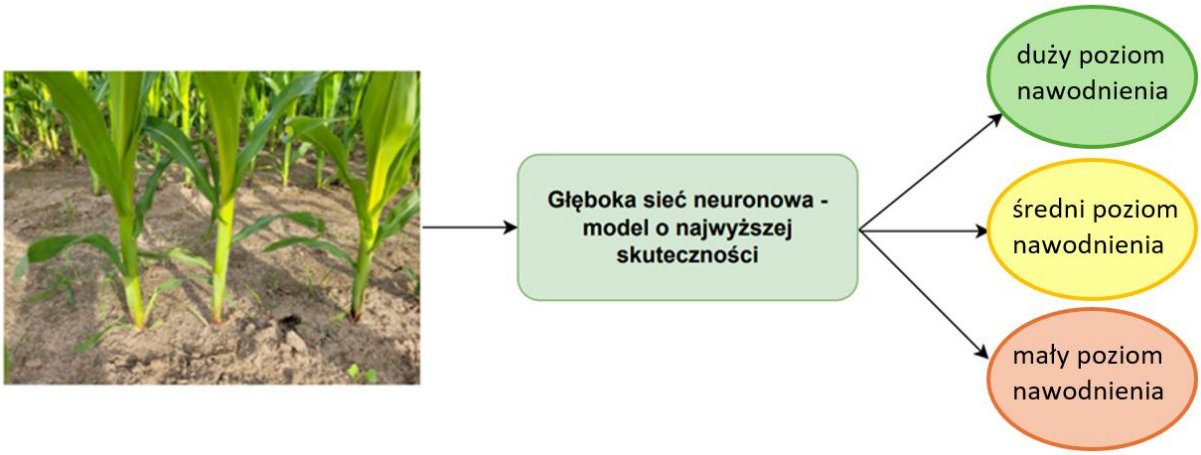
Rysunek 6.1. Schemat zaproponowanego rozwiązania w ramach realizacji zadania detekcji i klasyfikacji poziomów nawodnienia kukurydzy.

Zgodnie ze schematem, najpierw następuje proces ładowania danych (tutaj należy rozważyć konkretny typ danych: RGB lub wielospektralnych z kamery MapIR), po czym dane te są poddawane odpowiedniemu przetwarzaniu i normalizacji. W kolejnym kroku następuje etap trenowania głębokich sztucznych sieci neuronowych na bazie wczytanych obrazów z użyciem zaimplementowanych architektur, którymi są: ConvNeXt Small [Liu i in., 2022], ResNet50 [He i in., 2016], EfficientNet-B3 [Tan, Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] i Swin Transformer [Liu i in., 2021], 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 Survey3 Red+Green+NIR. 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 6.2. pokazuje 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 zobrazowań 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 z kamery MapIR Survey3 Red+Green+NIR

W przypadku analizy obrazów wielospektralnych z kamery MapIR Survey3 Red+Green+NIR 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 [Liu i in., 2022], ResNet50 [He i in., 2016], EfficientNet-B3 [Tan, Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] i Swin Transformer [Liu i in., 2021], 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 poszczególnych algorytmów. Do porównania zastosowano typowe metryki oceny: 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 Survey3 Red+Green+NIR przy zastosowaniu algorytmów ConvNeXt Small, ResNet50, EfficientNet-B3, Vision Transformer i Swin Transformer.

Jak widać na podstawie analizy zaprezentowanych wyników, najlepiej z detekcją i klasyfikacją poziomów nawodnienia kukurydzy na bazie obrazów z kamery wielospektralnej MapIR Survey3 Red+Green+NIR poradził sobie algorytm Swin Transformer, który uzyskał wartość parametru accuracy rzędu 0,8631 oraz wartość parametru F1-score rzędu 0,8506. Tuż za nim jest algorytm ConvNeXt Small z wartością accuracy rzędu 0,8548 oraz wartością F1-

score rzędu 0,8372. Dalej znajdują się kolejno algorytmy Vision Transformer, EfficientNet-B3 i ResNet50 z wartościami metryki accuracy równymi kolejno: 0,8257, 0,7718 i 0,7635 oraz z wartościami metryki F1-score równymi kolejno: 0,8105, 0,7426, 0,7301.

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 Survey3 Red+Green+NIR: ConvNeXt Small [Liu i in., 2022], ResNet50 [He i in., 2016], EfficientNet-B3 [Tan, Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] i Swin Transformer [Liu i in., 2021], 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 poszczególnych algorytmów. Do ich porównania zastosowano typowe metryki oceny: 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

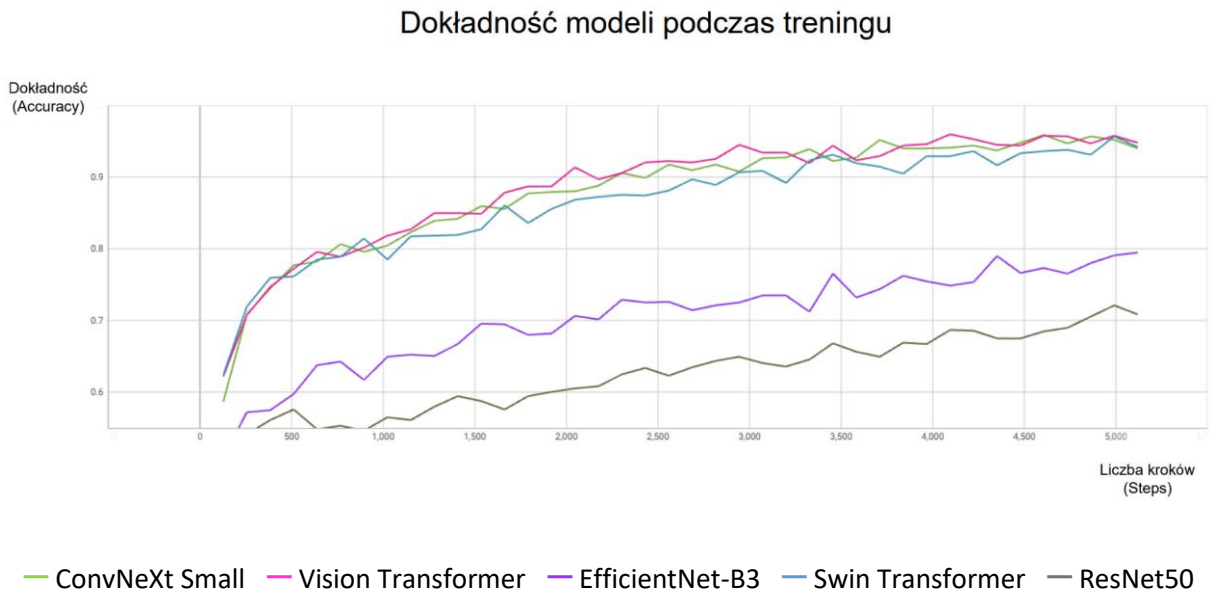
Tabela 6.2. Porównanie wyników uzyskanych dla danych RGB przy zastosowaniu algorytmów ConvNeXt Small, ResNet50, EfficientNet-B3, Vision Transformer i Swin Transformer.

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 oraz

wartość parametru F1-score rzędu 0,8747. Tuż za nim jest algorytm Vision Transformer z wartością accuracy rzędu 0,8506 oraz wartością F1-score rzędu 0,8298. Dalej znajdują się kolejno algorytmy Swin Transformer, EfficientNet-B3 i ResNet50 z wartościami metryki accuracy odpowiednio: 0,8382, 0,8091 i 0,7552 oraz wartościami metryki F1-score odpowiednio: 0,8149, 0,7911 i 0,7208.

6.4. Krzywe uczenia dla danych wielospektralnych i RGB

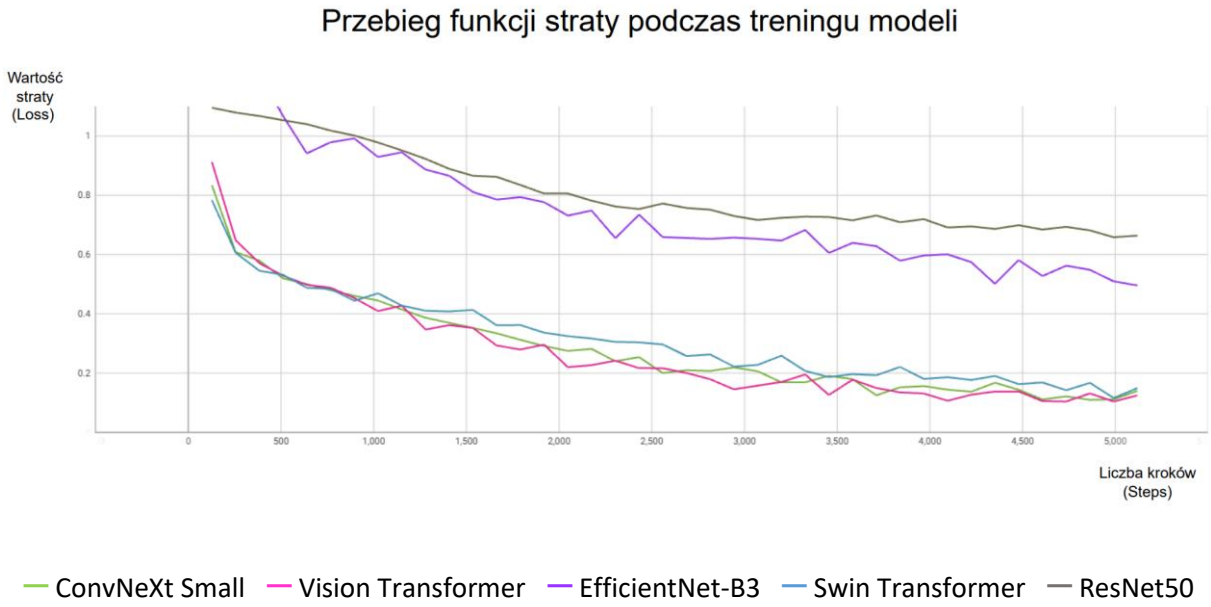
W tej części niniejszej rozprawy zaprezentowane zostaną krzywe uczenia zarejestrowane dla poszczególnych algorytmów służących do trenowania głębokich sztucznych sieci neuronowych w celu realizacji zadania polegającego na detekcji i klasyfikacji poziomów nawodnienia kukurydzy. Poniżej krzywych zamieszczone zostało podsumowanie z analizą porównawczą tychże krzywych.



Rysunek 6.3. Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas treningu na danych z kamery MapIR Survey3 Red+Green+NIR.

Rysunek 6.3. przedstawia zmianę dokładności modeli podczas treningu w zależności od liczby kroków treningu. Wykres obrazuje, jakie postępy osiągały poszczególne algorytmy 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

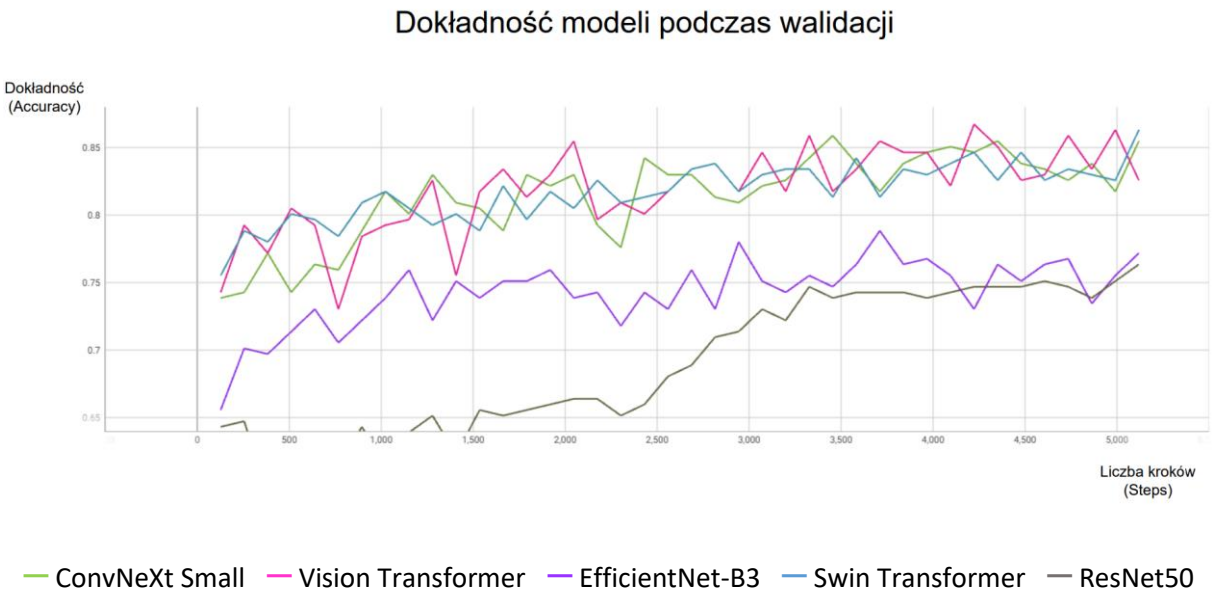
Small również uzyskał wysoką dokładność, zbliżając się do wyniku algorytmu Vision Transformer. Ostatecznie jego dokładność ustabilizowała się na poziomie nieco poniżej 0,9. Algorytm Swin Transformer osiągnął podobne wyniki do ConvNeXt Small, jednak z nieco większymi wahaniami w trakcie treningu, a ostatecznie 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 trening z wynikiem powyżej 0,8. Algorytm ResNet50 osiągnął zaś najniższą dokładność spośród wszystkich algorytmów. Z wykresu wynika, że algorytm Vision Transformer wykazał się największą stabilnością i szybkością w osiąganiu wysokiej dokładności, podczas gdy algorytm ResNet50 radził sobie najsłabiej.



Rysunek 6.4. Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu na danych z kamery MapIR Survey3 Red+Green+NIR.

Na Rysunku 6.4. przedstawiono przebieg funkcji straty w trakcie treningu modeli w zależności od liczby kroków treningu. 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 je na niskim poziomie do końca treningu. Algorytm ConvNeXt Small również wykazywał szybki spadek straty,

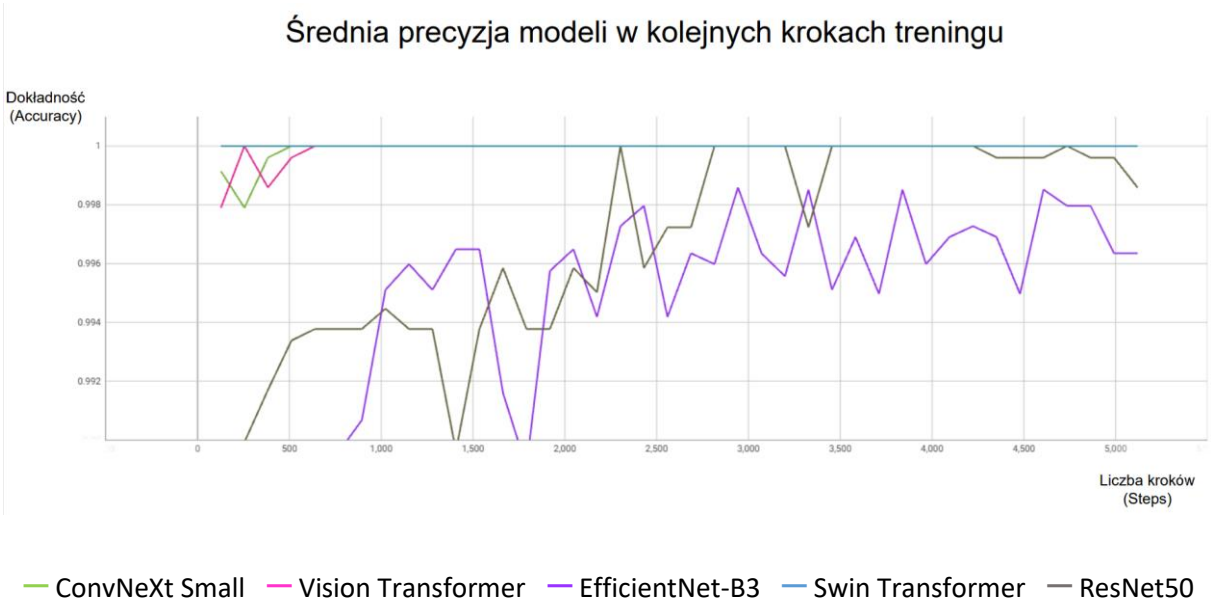
osiągając wartości podobne do algorytmu Vision Transformer, z końcową stratą oscylującą wokół 0,2. Algorytm Swin Transformer miał nieco większe wahania wartości straty, ale ostatecznie zbliżył się do poziomu algorytmów ConvNeXt Small i Vision Transformer, osiągając podobną końcową wartość straty. Algorytm 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 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 Survey3 Red+Green+NIR.

Rysunek 6.5 przedstawia dokładność modeli podczas walidacji w zależności od liczby kroków treningu. 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

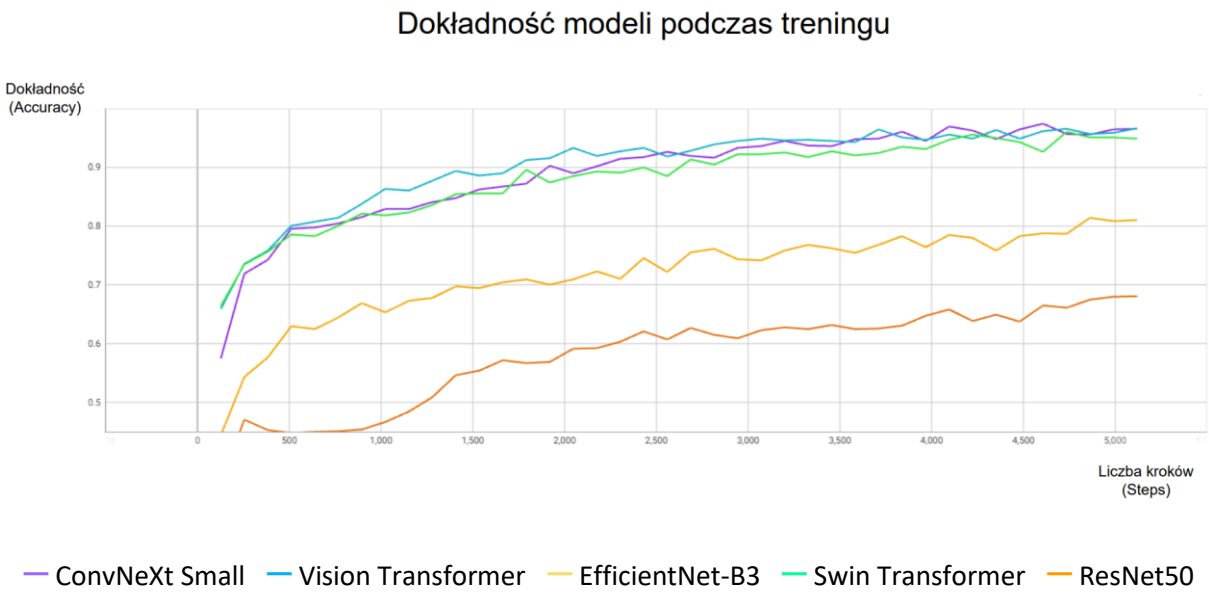
podobny przebieg walidacji, również wykazując pewne wahania, ale kończąc na stabilnym poziomie około 0,82 – 0,85. Algorytm EfficientNet-B3 miał niższą dokładność na początku walidacji, lecz w miarę postępu treningu osiągał stabilny wzrost, kończąc na poziomie około 0,75. Algorytm ResNet50 wykazywał najniższą dokładność spośród wszystkich algorytmów, utrzymując się poniżej wartości 0,7 przez większość czasu walidacji. Dopiero pod koniec walidacji jego dokładność zaczęła rosnąć, osiągając wartość około 0,7 na ostatnich krokach treningu. 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 Survey3 Red+Green+NIR.

Rysunek 6.6 przedstawia średnią precyzję modeli w kolejnych krokach treningu, mierzoną jako dokładność w zależności od liczby kroków treningu. Algorytm Swin Transformer wykazał stałą i wysoką precyzję przez cały proces treningu, utrzymując wartość bliską 1,0, co oznacza, że model ten był niezwykle dokładny od samego początku. Algorytmy Vision Transformer i ConvNeXt Small osiągnęły wysoką precyzję już na początku treningu, z wynikami bliskimi 1,0, i utrzymywały ten poziom przez pierwsze 500 kroków. Algorytmy

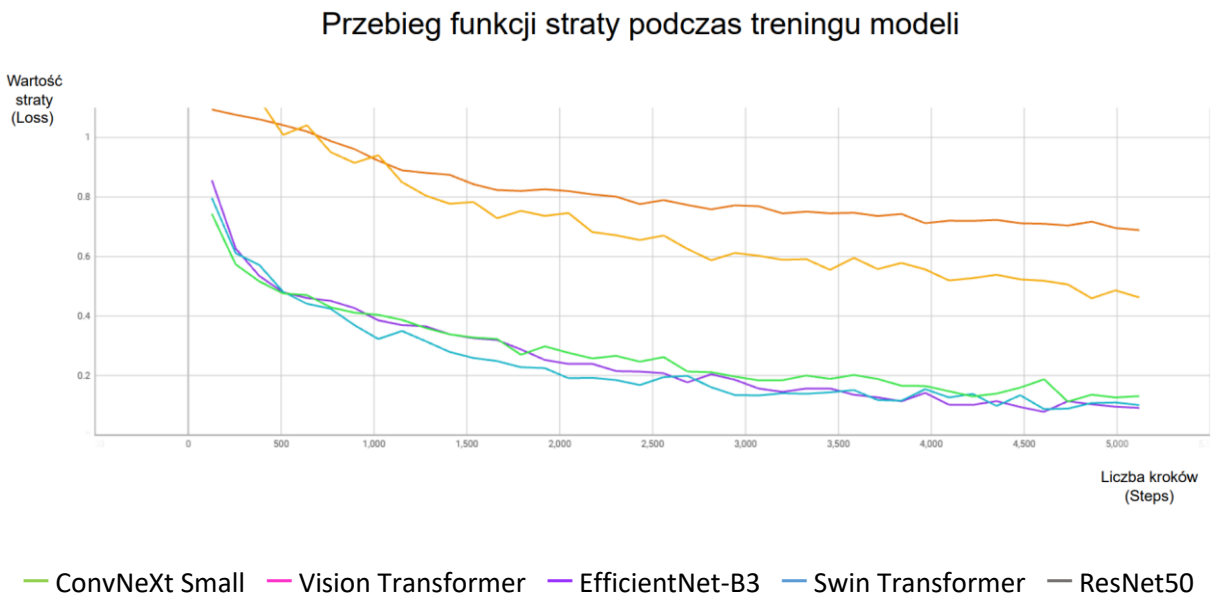
ResNet50 i EfficientNet-B3 wykazały znacznie większe wahania precyzji. ResNet50 początkowo uzyskał wysoką precyzję, jednak z biegiem treningu wartości te spadły, wykazując duże wahania. EfficientNet-B3 miał najniższą precyzję spośród wszystkich modeli, z wyraźnymi wahaniami i brakiem stabilności w wynikach utrzymujących się poniżej wartości 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.

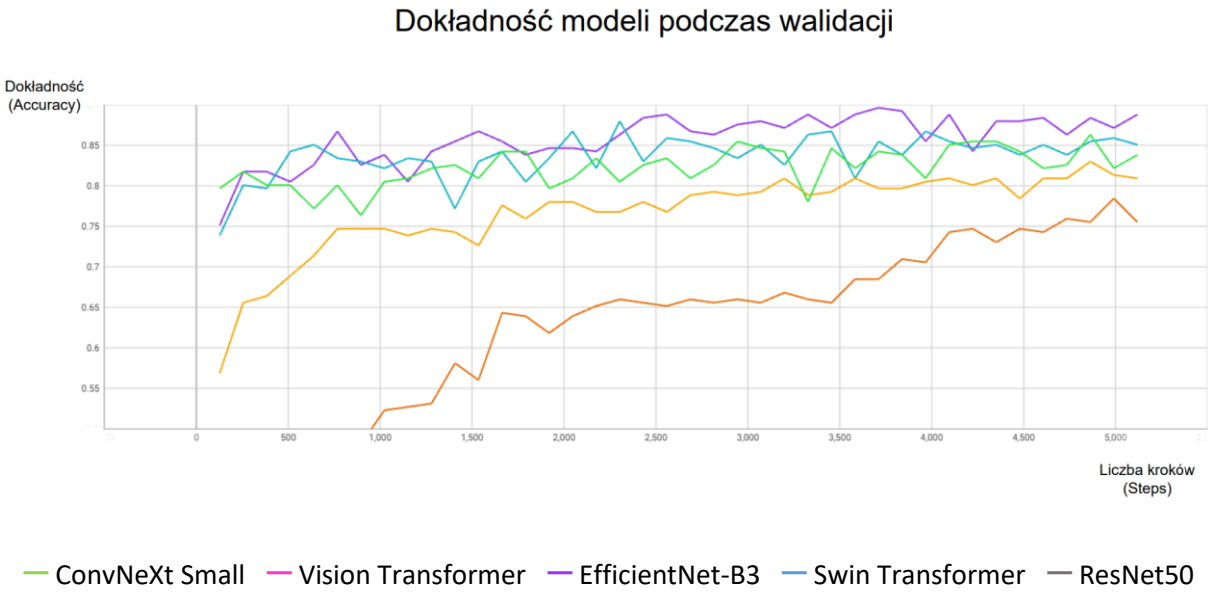
Na rysunku 6.7 przedstawiono dokładność modeli podczas treningu w zależności od liczby kroków treningu. Algorytm Vision Transformer osiągnął najwyższą dokładność, stabilizując się na poziomie bliskim 0,9 po około 2000 krokach treningu i utrzymując tę wartość dokładności do końca treningu. Algorytm Swin Transformer również osiągnął wysoką dokładność, 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 umiarkowana w porównaniu z wynikami pozostałych modeli. 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.



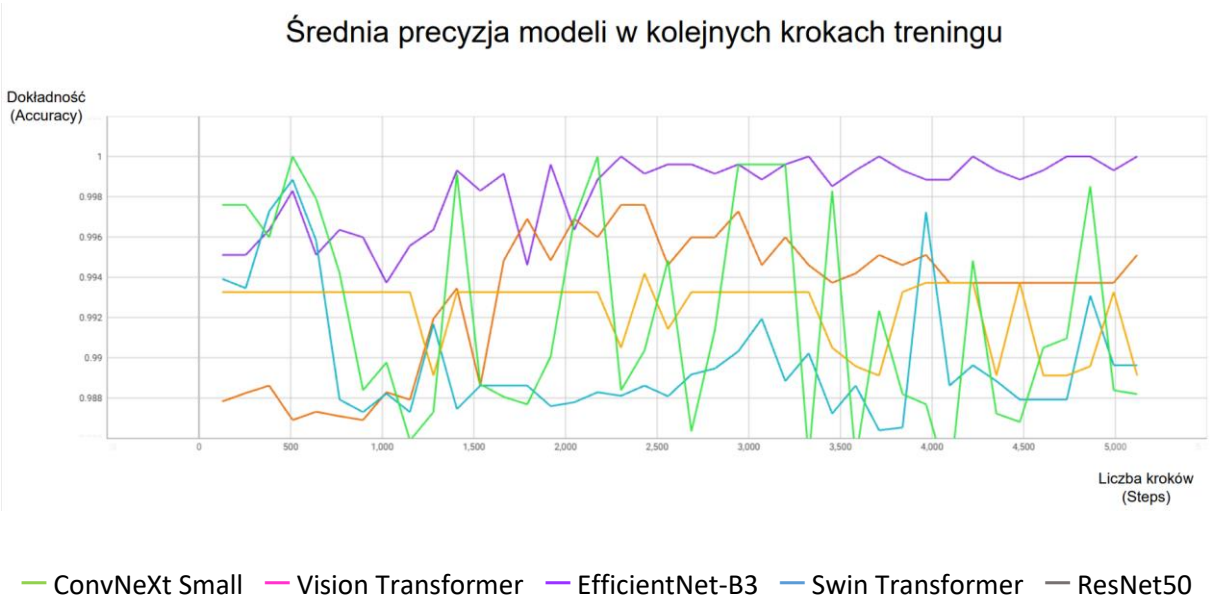
Rysunek 6.8. Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu na danych z kamery RGB.

Rysunek 6.8 przedstawia przebieg funkcji straty podczas treningu modeli w zależności od liczby kroków treningu. Algorytmy ConvNeXt Small, Vision Transformer i Swin Transformer wykazały najszybszy spadek wartości straty, osiągając wartości w przedziale 0,1 – 0,2 już po około 1000 krokach treningu 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 relatywnie 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 modeli podczas walidacji w zależności od liczby kroków treningu. Algorytm ConvNeXt Small osiągnął najwyższą dokładność podczas walidacji, stabilizując się w okolicach 0,85 – 0,87. Jest to najwyższa wartość spośród wszystkich analizowanych modeli w trakcie walidacji. Algorytmy Vision Transformer i Swin Transformer osiągnęły bardzo podobne wyniki, z dokładnością osiągającą stabilne wartości w zakresie 0,82 – 0,85. Oba modele wykazały się stabilnością i niewielkimi wahaniami. Algorytm EfficientNet-B3 miał średnią dokładność w walidacji, utrzymującą się stabilnie na poziomie około 0,75. Chociaż jego wyniki były gorsze od trzech poprzednich algorytmów, były stabilne przez większość treningu. 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 w zależności od liczby kroków treningu. Algorytm ConvNeXt Small wykazał się największą stabilnością, utrzymując wysoką precyzję przez cały czas trwania treningu. Jego dokładność była bliska wartości 1,0 i wykazywała minimalne wahania przez cały okres treningu, 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, przy czym zauważalne były sporadyczne spadki. Algorytm EfficientNet-B3 charakteryzował się niestabilną precyzją z wyraźnymi 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 znaczące 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, natomiast dla danych wielospektralnych z kamery MapIR Survey3 Red+Green+NIR najlepsze wyniki pozwolił osiągnąć algorytm Vision Transformer. Jeżeli chodzi o najlepsze wyniki globalne, to zostały one osiągnięte dla danych RGB i przy zastosowaniu algorytmu EfficientNet-B3.

7. Detekcja i klasyfikacja porażenia kukurydzy patogenami

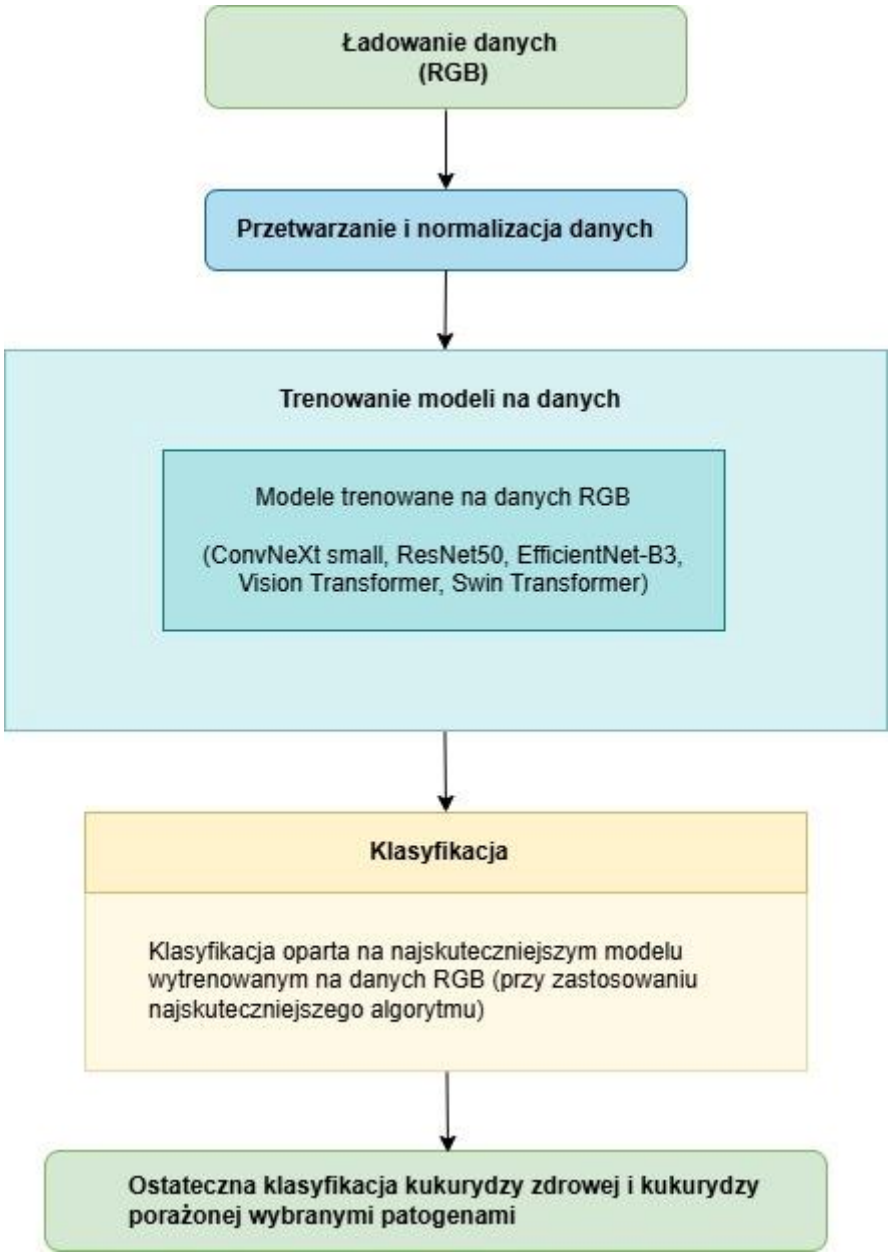
Zadanie związane z detekcją i klasyfikacją kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami miało na celu określenie, czy widoczna na wskazanym obrazie roślina jest zdrowa, czy porażona, a jeśli porażona to jakim patogenem spośród puli patogenów uwzględnianych w analizie. Do analizy porażenia 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*.

Celem nadrzędnym zadania było zaś zapewnienie dodatkowej funkcjonalności opracowanego systemu monitoringu kondycji kukurydzy polegającej na detekcji kukurydzy zdrowej oraz kukurydzy potencjalnie porażonej wybranymi patogenami oraz generowanie i zapisywanie wyników tejże detekcji w postaci wartości prawdopodobieństw wykrycia porażenia patogenami na poszczególnych częściach pola, z dokładnością do częstotliwości pobierania jednego obrazu.

W rozdziale tym zostanie zaprezentowany schemat zaproponowanego rozwiązania, wyniki uzyskane dla danych RGB pozyskanych z zewnętrznego zbioru danych [Kaggle], przebieg krzywych uczenia przy zastosowaniu badanych algorytmów uczenia głębokiego oraz porównanie wyników i wybór najlepszych rozwiązań.

7.1. Schemat zaproponowanego rozwiązania

Schemat zaprezentowany na rysunku 7.1. przedstawia główne podejście zastosowane w opracowanym systemie wykorzystujące obrazowanie 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.



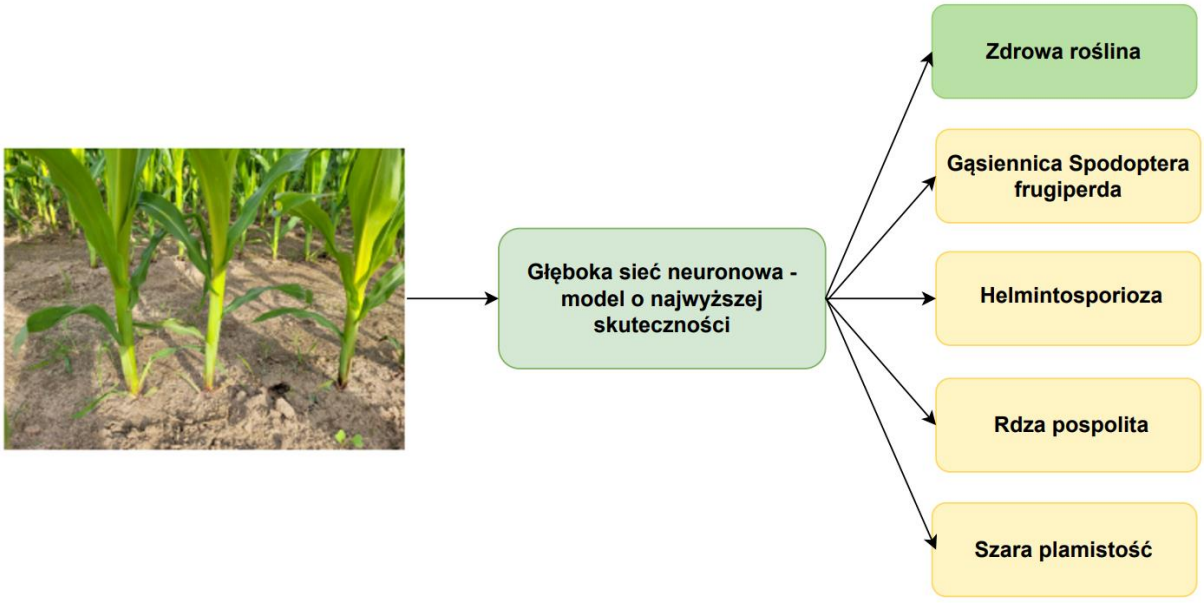
Rysunek 7.1. Schemat zaproponowanego rozwiązania w ramach realizacji zadania detekcji i klasyfikacji kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami.

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 [Liu i in., 2022], ResNet50 [He i in., 2016], EfficientNet-B3 [Tan, Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] i Swin Transformer [Liu i in., 2021], a każda z tych architektur prowadzi do stworzenia osobnego modelu. Po etapie trenowania i analizowaniu

poszczególnych modeli następuje wybór najlepszego algorytmu oraz klasyfikacja porażenia kukurydzy z użyciem modelu wytrenowanego przy wykorzystaniu tego właśnie algorytmu. 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 zbiór danych zawierający obrazy kukurydzy porażonej wybranymi patogenami [Kaggle].

Finalna klasyfikacja opiera się na wyborze najskuteczniejszej metody wykorzystującej zobrazowanie RGB oraz architekturę uczenia głębokiego, która osiągnęła najlepsze wyniki spośród wszystkich analizowanych architektur.

Na rysunku 7.2. pokazano 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 wyniku 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 modeli wykorzystywanych podczas procesu trenowania zostaną omówione w kolejnych podrozdziałach. Umożliwi to prześledzenie wyboru najlepszego rozwiązania.

7.2. Wyniki i ich analiza

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 [Liu i in., 2022], ResNet50 [He i in., 2016], EfficientNet-B3 [Tan, Le, 2019], Vision Transformer [Dosovitskiy i in., 2020] I Swin Transformer [Liu i in., 2021]. 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 poszczególnych algorytmów. Do porównania zastosowano następujące metryki oceny: accuracy, precision, recall i F1-score.

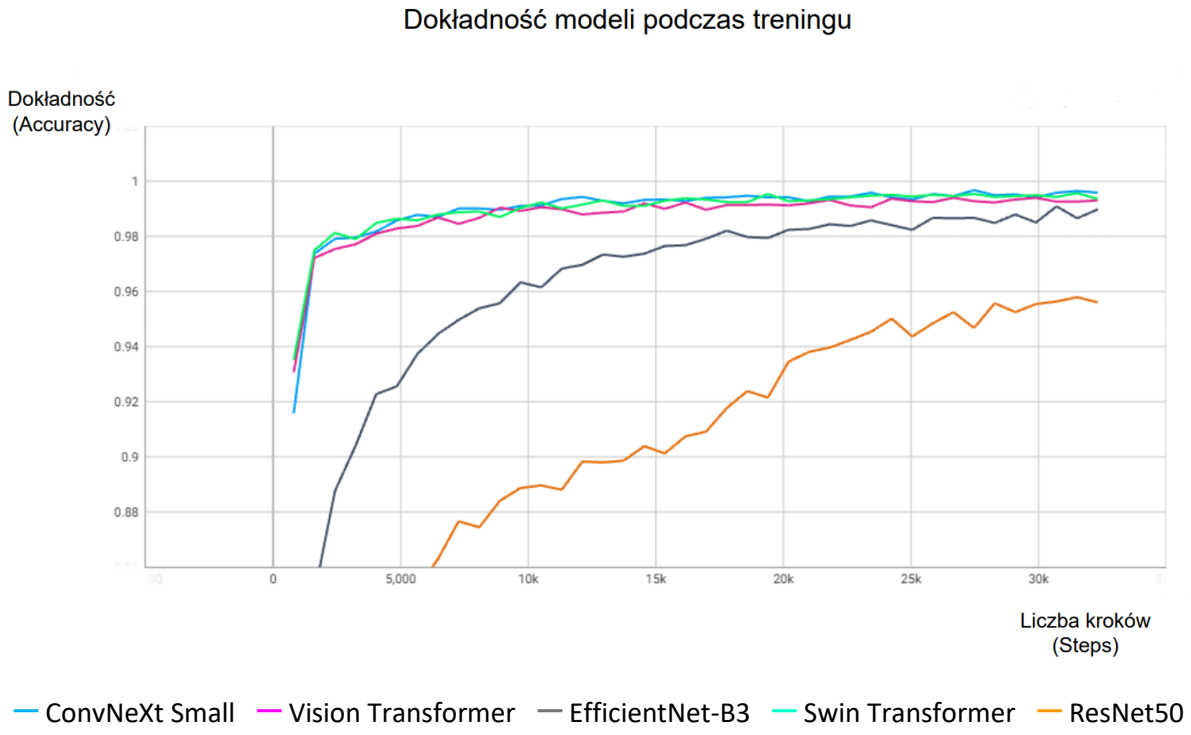
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
ConvNeXt Small	0.9969	0.9930	0.9921	0.9926
ResNet50	0.4416	0.4411	0.4171	0.3439
EfficientNet B3	0.9951	0.9902	0.9905	0.9904
Vision Transformer	0.9988	0.9962	0.9980	0.9971
Swin Transformer	0.9994	0.9981	0.9990	0.9985

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 oraz wartość parametru F1-score rzędu 0,9985. Tuż za nim jest algorytm Vision Transformer z wartością accuracy rzędu 0,9988 oraz wartością F1-score rzędu 0,9971. Dalej znajdują się kolejno algorytmy ConvNeXt Small, EfficientNet-B3 i ResNet50 z wartościami metryki accuracy odpowiednio: 0,9969, 0,9951 i 0,4416 oraz wartościami metryki F1-score odpowiednio: 0,9926, 0,9904 i 0,3439.

7.3. Krzywe uczenia

W tej części niniejszej rozprawy zostały zaprezentowane krzywe uczenia zarejestrowane dla poszczególnych modeli. W nawiązaniu do otrzymanych krzywych zamieszczone zostało podsumowanie z analizą porównawczą tychże krzywych.

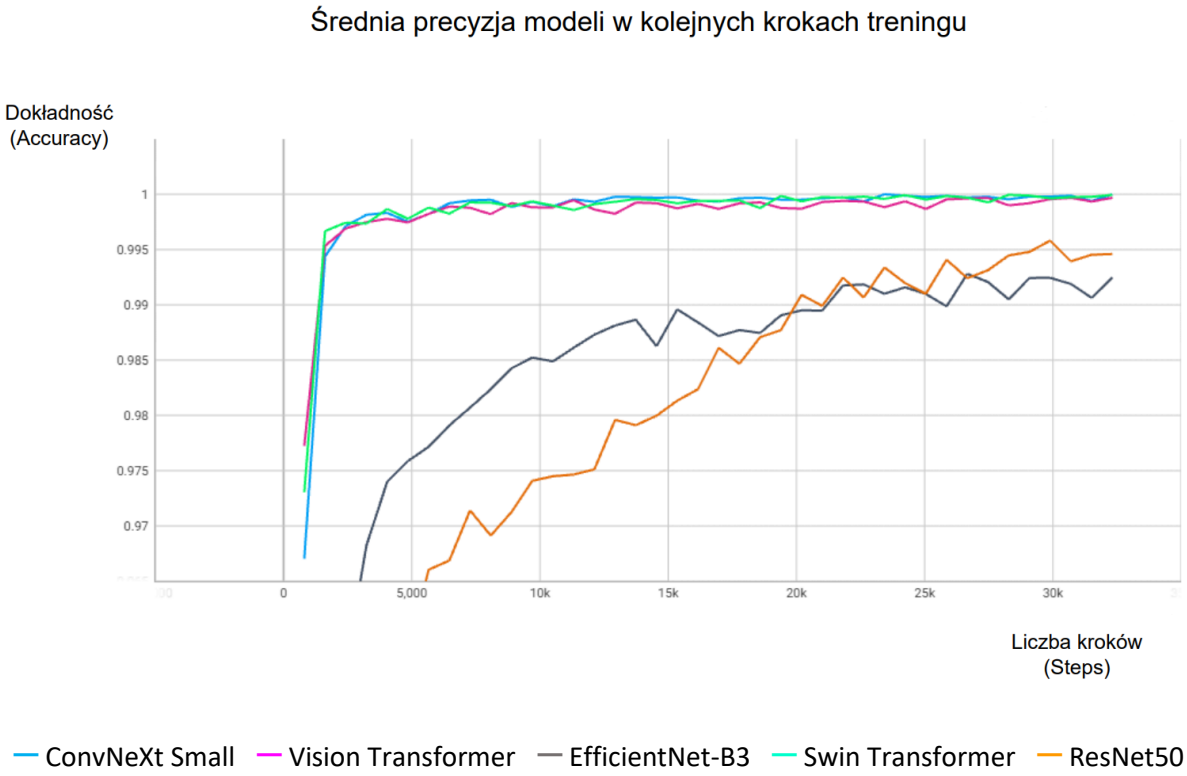


Rysunek 7.3. Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas treningu.

Rysunek 7.3 przedstawia dokładność modeli podczas treningu w zależności od liczby kroków treningu. Model 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. Modele Vision Transformer i Swin Transformer wykazały bardzo podobny przebieg treningu, osiągając wysoką dokładność bliską 1,0, i zachowując stabilność przez większość trwania treningu. Model EfficientNet-B3 początkowo uzyskiwał niższą dokładność, ale stopniowo poprawiał swoje wyniki, osiągając ostatecznie poziom około 0,98. W porównaniu do ConvNeXt Small, Vision Transformer i Swin Transformer, poprawa tego modelu była bardziej stopniowa. Z kolei model ResNet50

wykazywał najniższą dokładność spośród wszystkich analizowanych algorytmów. Jego dokładność wzrastała powoli, osiągając wartość maksymalnie około 0,93.

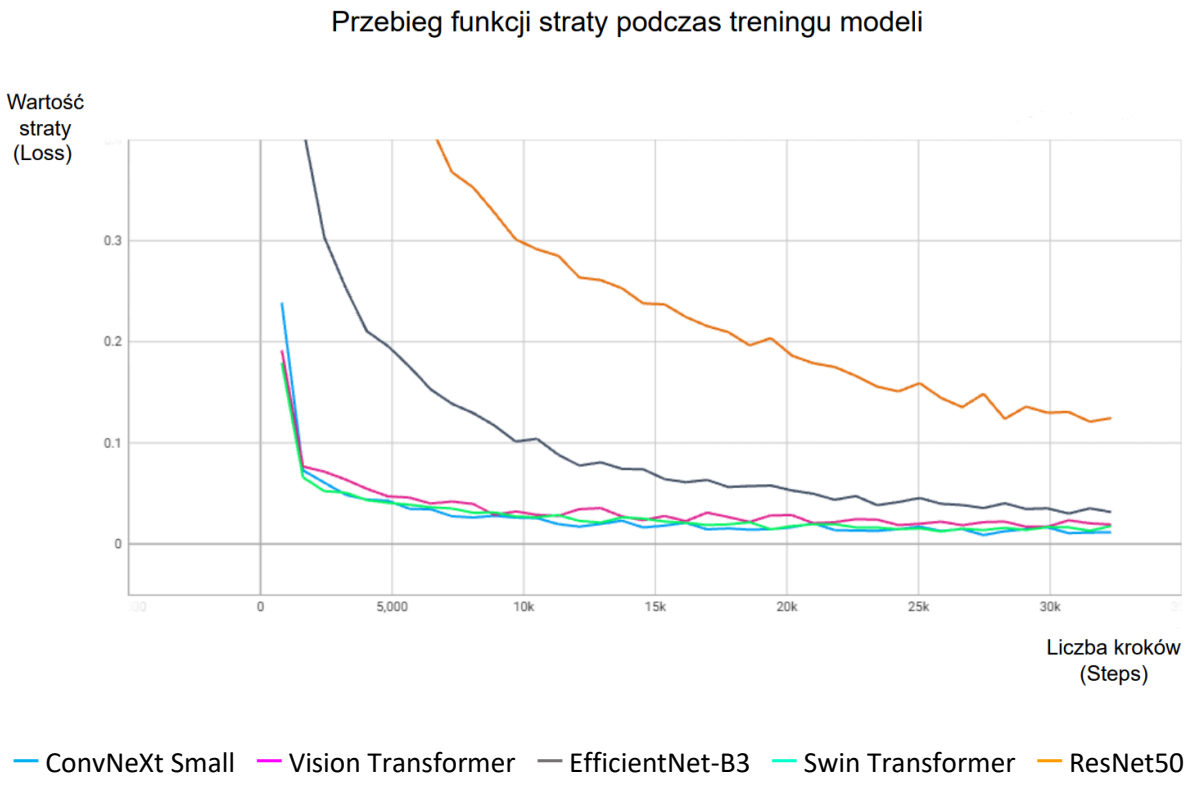
Podsumowując, ConvNeXt Small, Vision Transformer i Swin Transformer okazały się najbardziej skutecznymi i stabilnymi modelami podczas treningu.



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 w zależności od liczby kroków treningu. Modele ConvNeXt Small, Vision Transformer, i Swin Transformer osiągnęły precyzję bliską 1,0 już na wczesnym etapie treningu i utrzymały tę wartość przez cały czas jego trwania. Wykresy tychże modeli cechują się dużą stabilnością oraz minimalnymi wahaniami w kolejnych krokach. Model EfficientNet-B3 na początku osiągał nieco niższą precyzję, ale z każdym krokiem treningowym jego wartość rosła, osiągając około wartość 0,985 na końcu procesu. Model ResNet50 wykazywał stopniowy wzrost precyzji, osiągając wartość bliską 0,98 pod koniec treningu.

Podsumowując, wszystkie analizowane modele poprawiały swoją precyzję w trakcie treningu, przy czym ConvNeXt Small, Vision Transformer i Swin Transformer osiągnęły najwyższe wartości i wykazały się największą stabilnością.

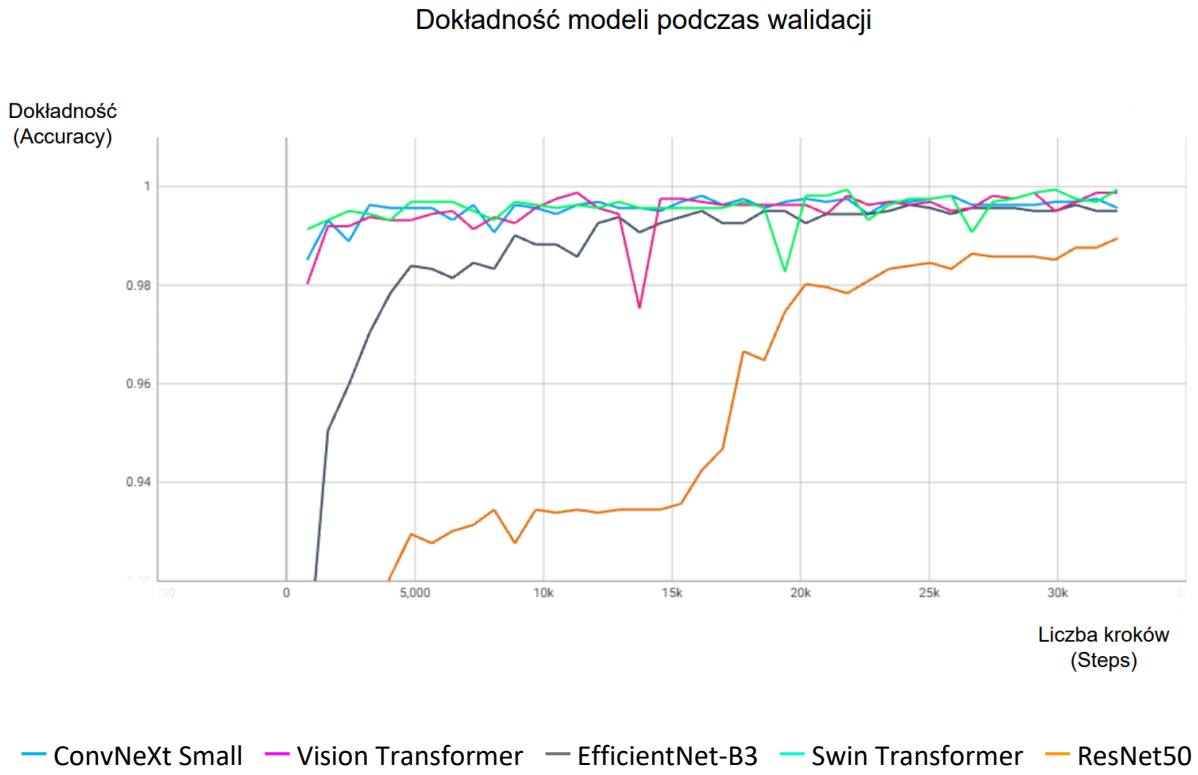


Rysunek 7.5. Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas treningu.

Na zaprezentowanym na rysunku 7.5. wykresie przedstawiony został przebieg funkcji straty podczas treningu modeli w zależności od liczby kroków treningu. Modele ConvNeXt Small, Vision Transformer, i Swin Transformer szybko osiągnęły niskie wartości funkcji straty, oscylujące w okolicach 0,05 lub niżej, i utrzymywały je na stabilnym poziomie przez cały czas trwania treningu. Model EfficientNet-B3 początkowo wykazywał wyższą wartość straty, ale sukcesywnie ją redukował, osiągając stabilny poziom około 0,1. Wynik ten jest wyższy w porównaniu do modeli ConvNeXt Small, Vision Transformer i Swin Transformer, ale nadal pokazuje skuteczny proces minimalizacji funkcji straty. Model ResNet50 miał najwyższą wartość straty spośród wszystkich analizowanych modeli, chociaż strata tego modelu

systematycznie malała w trakcie treningu, a na końcu procesu utrzymała się na poziomie powyżej 0,3.

Podsumowując, wszystkie modele zredukowały wartość funkcji straty w trakcie treningu, przy czym ConvNeXt Small, Vision Transformer i Swin Transformer osiągnęły najniższe wartości.

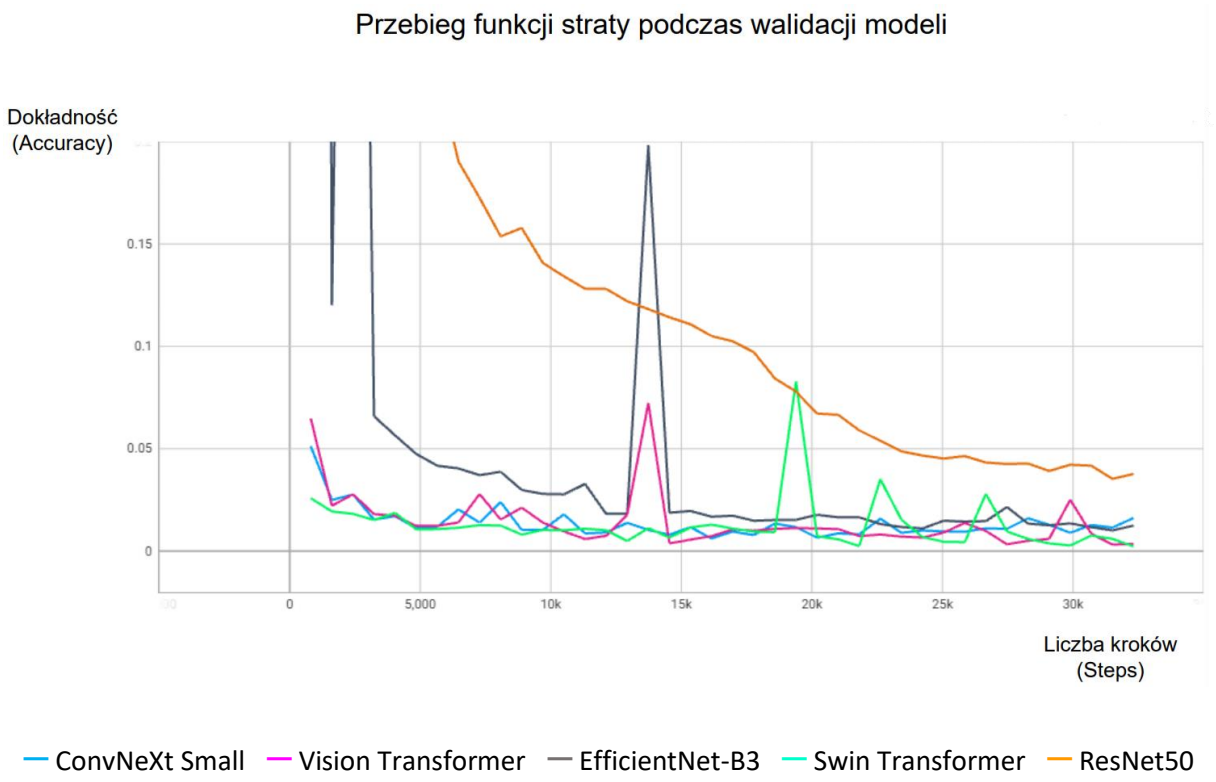


Rysunek 7.6. Krzywe dla poszczególnych modeli przedstawiające zmiany ich dokładności podczas procesu walidacji.

Rysunek 7.6. przedstawia wykres dokładności modeli podczas walidacji w zależności od liczby kroków treningu. Modele ConvNeXt Small, Vision Transformer, i Swin Transformer osiągnęły najwyższą dokładność, utrzymując wartości bliskie 1,0 przez większość procesu walidacji. Pomimo drobnych wahań, ich dokładność pozostała stabilna na przestrzeni całego treningu. Model EfficientNet-B3 początkowo miał niższą dokładność, ale szybko poprawił swoje wyniki, utrzymując je na poziomie około 0,98. Jego dokładność była stabilna, choć nieco niższa niż w przypadku modeli ConvNeXt Small, Vision Transformer i Swin Transformer. Model ResNet50 wykazywał najniższą dokładność spośród wszystkich analizowanych modeli,

a jego wyniki stopniowo się poprawiały, osiągając maksymalnie wartość bliską 0,96 na końcu walidacji.

Podsumowując, modele ConvNeXt Small, Vision Transformer i Swin Transformer wyróżniły się najwyższą dokładnością i stabilnością podczas walidacji. Model EfficientNet-B3 również osiągnął wysoką dokładność i stabilność. ResNet50 charakteryzował się natomiast niższą dokładnością w porównaniu do pozostałych modeli.



Rysunek 7.7. Krzywe dla poszczególnych modeli przedstawiające przebieg funkcji straty podczas walidacji.

Rysunek 7.7. przedstawia wykres przebiegu funkcji straty podczas walidacji modeli w zależności od liczby kroków treningu. Modele ConvNeXt Small, Swin Transformer i Vision Transformer uzyskały najniższe wartości funkcji straty, które oscylowały blisko zera przez większość czasu trwania walidacji. Sporadyczne wahania były minimalne, co wskazuje na stabilność tychże modeli. Model EfficientNet-B3 wykazywał większe wahania wartości funkcji straty, szczególnie widoczne w okolicach 10 000 i 15 000 kroków, gdzie odnotowano chwilowe wzrosty. Po tych skokach funkcja straty zmniejszyła się, osiągając końcowo relatywnie niskie wartości, co wskazuje na poprawę w dalszej części walidacji. Model ResNet50 charakteryzował

się najwyższymi wartościami funkcji straty spośród wszystkich analizowanych modeli, jego strata systematycznie malała, ale przez prawie cały przebieg treningu pozostawała wyższa w porównaniu do innych modeli, jedynymi wyjątkami były skokowe wzrosty wartości funkcji straty w przypadku modeli EfficientNet-B3 i Swin Transformer.

Podsumowując, modele ConvNeXt Small, Swin Transformer i Vision Transformer były najbardziej skuteczne w minimalizowaniu wartości funkcji straty podczas walidacji. Model EfficientNet-B3 wykazał większe wahania, a model ResNet50 miał najwyższą wartość funkcji 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 i EfficientNet-B3. Z kolei algorytm ResNet50 poradził sobie wyraźnie gorzej niż pozostałe algorytmy.

8. Prezentacja wyników

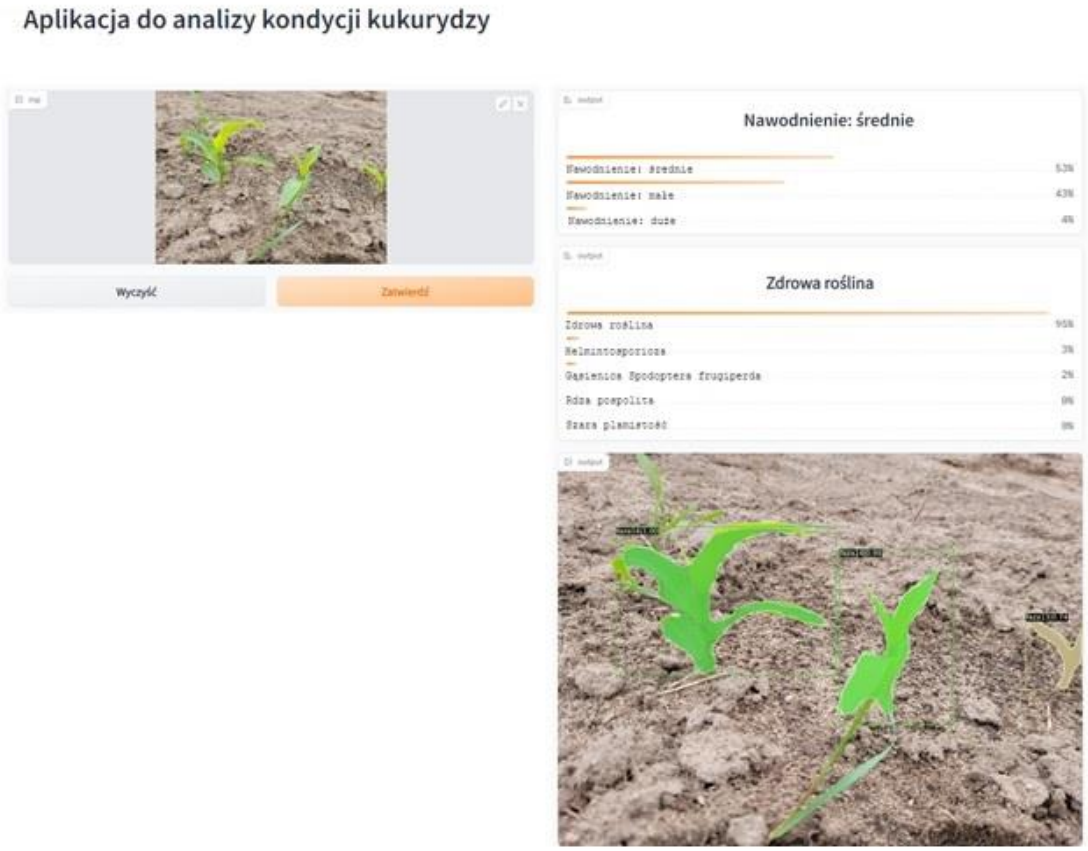
Wszystkie zadania projektowe będące przedmiotem niniejszej rozprawy, czyli detekcja kukurydzy i klasyfikacja jej faz rozwojowych, detekcja i klasyfikacja poziomów nawodnienia kukurydzy oraz detekcja i klasyfikacja kukurydzy zdrowej i kukurydzy porażonej wybranymi patogenami, zostały zrealizowane. Niezmiernie ważna stała się zatem odpowiednia prezentacja otrzymanych wyników. Została ona zrealizowana dwutorowo: poprzez wizualizację w aplikacji opracowanej dzięki bibliotece Gradio [Gradio, 2024] oraz poprzez opracowanie metody do generowania wyjściowego pliku w formacie CSV zawierającego wszystkie odpowiedzi zestawu wytrenowanych modeli wraz z niezbędnymi parametrami przetwarzania. W kolejnych podrozdziałach zaprezentowane zostaną oba podejścia oraz uzyskane w ten sposób przykładowe wyniki.

8.1. Aplikacja wykorzystująca bibliotekę Gradio

Wizualną prezentację wyników działania wytrenowanych modeli zrealizowano przy użyciu biblioteki Gradio [Gradio, 2024]. Jest to biblioteka typu open source języka Python, która służy do tworzenia wizualnych demonstracji uczenia maszynowego. Pozwala na tworzenie interfejsu użytkownika dla modeli uczenia maszynowego i umożliwia wypróbowanie działania tychże modeli poprzez przeglądarkę. Taki był też cel zastosowania tejże aplikacji, aby w charakterze kontrolnym można było zbadać działanie wytrenowanych modeli na wybranych obrazach. Analiza ta jest możliwa poza fizycznym prototypem robota, na komputerze o odpowiednio skonfigurowanym środowisku, aby bez konieczności prac w terenie móc testować jakość opracowanego rozwiązania i przetwarzać dowolnie wybrane obrazy kukurydzy.

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 i przeanalizowanie go przez poszczególne modele. Na wyjściu generowane są 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 wykrytych obiektów zaznaczonych na analizowanym obrazie w postaci kolorowych obszarów.

Na rysunku 8.1. przedstawiono wygląd aplikacji stworzonej przy wykorzystaniu biblioteki Gradio. Widoczne są na nim wyniki analizy poziomów nawodnienia kukurydzy oraz porażenia wybranymi patogenami, wraz z prawdopodobieństwami wykrycia odpowiednich klas, a także wyniki detekcji faz rozwojowych kukurydzy dla przykładowego obrazu RGB.



Rysunek 8.1. Wygląd interfejsu stworzonej aplikacji Gradio [Gradio, 2024] - przypadek analizy obrazu RGB.

8.2. Plik wyjściowy w formacie CSV

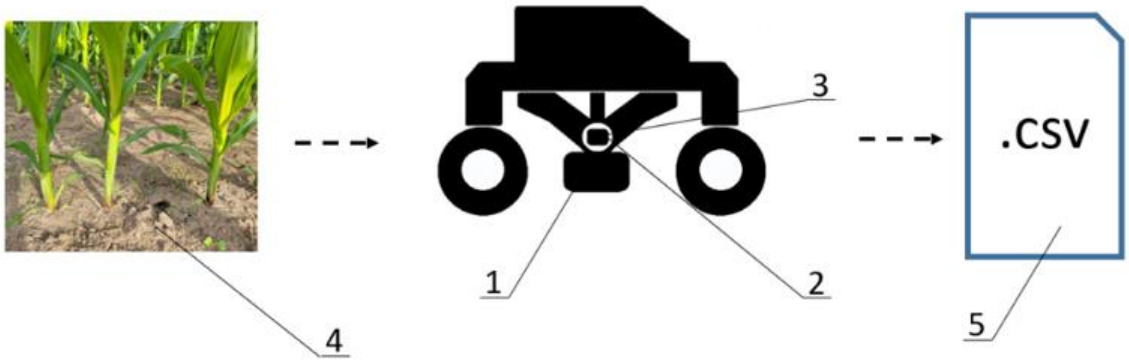
Oprócz strony wizualnej prezentującej wyniki działania zestawu wytrenowanych modeli, w ramach opracowywania zadań projektowych został napisany również kod, który jest uruchamiany na komputerze dostępnym na prototypie robota polowego, który podczas pracy robota generuje z odpowiedzi poszczególnych modeli wyjściowy plik w formacie CSV. Plik ten jest kluczowy dla gromadzenia wyników prowadzonych analiz terenowych i eksportowania ich następnie do odpowiedniej bazy danych w celu dalszego przetwarzania. W danym pliku

.csv możliwa jest agregacja odpowiedzi sieci generowanych podczas pracy robota dla wielu roślin, czyli dla wielu przeanalizowanych obrazów. Dodatkowo, znajdują się w nim również dodatkowe dane, jak np. nazwy obrazów wraz z ich lokalizacjami w postaci ścieżek do plików, wartości czasu przetwarzania oraz wartości parametru timestamp, który ułatwia ustalenie geolokalizacji obrazów w późniejszym etapie. Plik ten jest sukcesywnie uzupełniany o kolejne wiersze w trakcie procesu przetwarzania obrazów, które są poddawane analizie przez wytrenowane modele. W pliku CSV znajdują się następujące kolumny:

- *Filename* – nazwa obrazu
- *Nawodnienie małe* – wartość prawdopodobieństwa wykrycia nawodnienia przypisanego do klasy „małe”
- *Nawodnienie średnie* – wartość prawdopodobieństwa wykrycia nawodnienia przypisanego do klasy „średnie”
- *Nawodnienie duże* – wartość prawdopodobieństwa wykrycia nawodnienia przypisanego do klasy „duże”
- *Porażenie Gąsiennica Spodoptera frugiperda* – wartość prawdopodobieństwa wykrycia porażenia gąsienicą Spodoptera frugiperda
- *Porażenie Helmintosporioza* – wartość prawdopodobieństwa wykrycia porażenia grzybami z rodzaju Helmintosporioza
- *Porażenie Rdza pospolita* – wartość prawdopodobieństwa wykrycia porażenia rdzą pospolitą
- *Porażenie Szara plamistość* – wartość prawdopodobieństwa wykrycia porażenia szarą plamistością
- *Porażenie Zdrowa roślina* – wartość prawdopodobieństwa wykrycia zdrowej rośliny
- *faza10* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 10 w skali BBCH
- *faza11* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 11 w skali BBCH
- *faza12* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 12 w skali BBCH
- *faza13* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 13 w skali BBCH

- *faza14* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 14 w skali BBCH
- *faza15* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 15 w skali BBCH
- *faza16* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 16 w skali BBCH
- *faza17* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 17 w skali BBCH
- *faza18* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 18 w skali BBCH
- *faza19* – wartość prawdopodobieństwa wykrycia kukurydzy będącej w fazie 19 w skali BBCH
- *processing time* – wartość długości czasu przetwarzania obrazu
- *timestamp* – znacznik czasowy (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 opracowanie 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 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 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 zobrazowań i prowadzenia obliczeń oraz podsumowano uzyskane wyniki w ramach dokumentacji projektowej i publikacji naukowych.

Opracowany system wspiera automatyzację oceny kondycji upraw kukurydzy 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 może odegrać kluczową rolę w rozwoju zautomatyzowanych rozwiązań w dziedzinie rolnictwa precyzyjnego.

9.1.1. Główne osiągnięcia pracy i jej oryginalny wkład

Oryginalnym dorobkiem naukowym autorki, powstałym w ramach prac badawczych prowadzonych nad przygotowywaniem niniejszej rozprawy oraz wnoszącym wkład w rozwój dyscypliny naukowej Informatyka Techniczna i Telekomunikacja, jest:

a) Stworzenie autorskich, oznaczonych zbiorów danych z kamery RGB, kamery wielospektralnej MicaSense RedEdge MX Dual oraz kamery wielospektralnej MapIR Survey3 Red+Green+NIR 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 z kamery wielospektralnej MicaSense RedEdge MX Dual do zupełnie nowego zastosowania, jakim jest detekcja i klasyfikacja faz rozwojowych kukurydzy – (rozdziały 4, 5, 6) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]
c) Przeprowadzenie nowatorskich badań i wykorzystanie zobrazowań z kamery wielospektralnej MapIR Survey3 Red+Green+NIR 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]
d) Zastosowanie metody klasyfikatora głosującego, w którym modele wytrenowane na pojedynczych kanałach spektralnych zarejestrowanych za pomocą kamery MicaSense RedEdge MX Dual głosują nad sumarycznym wynikiem klasyfikacji faz rozwojowych kukurydzy – (rozdziały 3.5.2., 4.4.) [Stypułkowska, Rokita, 2024]
e) 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]
f) Wykorzystanie zaawansowanych architektur głębokich sztucznych sieci neuronowych do zupełnie nowego zastosowania, niż przykłady obecne w dostępnej literaturze naukowej – (rozdziały 3.3., 3.4., 3.5., 4, 5, 6, 7) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]
g) Automatyzacja procesów jednoczesnego określania takich parametrów kondycji kukurydzy jak: fazy rozwojowe, poziom nawodnienia i porażenie wybranymi

patogenami (rozdziały 4, 5, 6, 7, 8) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]

h) Zastosowanie opracowanego rozwiązania w rzeczywistym projekcie robota polowego, tytuł projektu: „Inteligentny robot spełniający wymogi rolnictwa precyzyjnego" – (rozdziały 4, 5, 6, 7, 8) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]

i) Wykorzystanie zdobytej wiedzy również w innym projekcie pn. „Zwiększenie konkurencyjności polskich towarów roślinnych na rynkach międzynarodowych poprzez podniesienie ich jakości i bezpieczeństwa fitosanitarnego FITOEXPORT" – (rozdziały 4, 8) [Stypułkowska, Rokita, 2024]

j) Wpływ opracowanego rozwiązania na automatyzację procesów charakterystycznych dla innej dziedziny wiedzy, jaką jest rolnictwo precyzyjne (rozdziały 4, 5, 6, 7, 8) [Stypułkowska, 2023a], [Stypułkowska, 2023b], [Stypułkowska, Rokita, 2024]

Ad. a) Stworzenie autorskich zbiorów danych w postaci:

- oznaczonego zbioru obrazów pochodzących z kamery MicaSense RedEdge MX Dual, odpowiadających poszczególnym 10 kanałom spektralnym, z obiektami w postaci kukurydzy oznaczonymi metodą poligonów i mającymi przypisaną klasę odpowiadającą poszczególnym fazom rozwoju zgodnie z międzynarodową skalą rozwoju roślin 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 analogiczne do obiektów powyższego zbioru pochodzącego z kamery MicaSense RedEdge MX Dual);
- oznaczonego zbioru obrazów pochodzących z kamery wielospektralnej MapIR Survey3 Red+Green+NIR, z obiektami w postaci kukurydzy oznaczonymi metodą poligonów 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 poligonów i mającymi przypisaną klasę odpowiadającą poszczególnym fazom rozwoju w skali BBCH (obiekty

w tym zbiorze są analogiczne do obiektów powyższego zbioru pochodzącego z kamery MapIR Survey3 Red+Green+NIR);

- oznaczonego zbioru obrazów pochodzących z kamery wielospektralnej MapIR Survey3 Red+Green+NIR, oznaczonych metodą *ground truth* i odpowiadających przyjętym zakresom nawodnienia kukurydzy;
- oznaczonego zbioru obrazów pochodzących z kamery RGB, oznaczonych metodą *ground truth* i odpowiadających przyjętym zakresom nawodnienia kukurydzy (obiekty w tym zbiorze są analogiczne do obiektów z powyższego zbioru pochodzącego z kamery MapIR Survey3 Red+Green+NIR).

Jak zauważono na podstawie analizowanej literatury, żaden analogiczny zbiór danych nie został jeszcze dotychczas opublikowany. Są to zatem zupełnie nowe zbiory oznaczonych danych obrazowych, które otwierają pole do prowadzenia analogicznych badań i niewątpliwie stanowią oryginalny dorobek naukowy. Stworzone zbiory powstały na poletku testowym uprawianym na potrzeby badań przez autorkę rozprawy.

Ad. b) Wykorzystanie zobrazowania z kamery wielospektralnej MicaSense RedEdge MX Dual do zupełnie nowego zastosowania, jakim jest detekcja i klasyfikacja faz rozwojowych kukurydzy. Analiza literaturowa pozwoliła wysnuć wniosek, iż do tej pory jeszcze nikt nie prowadził identycznych badań, co podkreśla innowacyjność przeprowadzonych prac.

Ad. c) Wykorzystanie zobrazowania z kamery wielospektralnej MapIR Survey3 Red+Green+NIR 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. d) Zastosowanie metody prostego 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 wskazanych 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. e) 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 realizację podejścia do tego tematu z bliższej perspektywy, niż znaczna część dostępnych opublikowanych badań naukowych, co niewątpliwie poszerza dorobek naukowy w tej kwestii.

Ad. f) 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. W ramach zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy są to architektury: HTC_r50, HTC_r101, HTC_x101, Mask2Former, zaś w przypadku zadań związanych 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. g) Automatyzacja procesów jednoczesnego określania parametrów kondycji kukurydzy w postaci:

- faz rozwojowych, poziomów nawodnienia i porażenia wybranymi patogenami – w przypadku obrazów RGB,
- faz rozwojowych - w przypadku obrazów wielospektralnych z kamery MicaSense RedEdge MX Dual,
- faz rozwojowych i poziomów nawodnienia - w przypadku obrazów wielospektralnych z kamery MapIR Survey3 Red+Green+NIR,

a także jednoczesny export wyników w postaci dogodnej do dalszego przetwarzania w formacie CSV.

Ad. h) Zastosowanie rozwiązania w rzeczywistym projekcie robota polowego (tytuł projektu: „Inteligentny robot spełniający wymogi rolnictwa precyzyjnego”), co pokazało jego praktyczne wykorzystanie w rzeczywistych warunkach uprawy kukurydzy.

Ad i) Wykorzystanie zdobytej wiedzy również w innym projekcie pn. „Zwiększenie konkurencyjności polskich towarów roślinnych na rynkach międzynarodowych poprzez

podniesienie ich jakości i bezpieczeństwa fitosanitarnego FITOEXPORT", co pokazało istnienie zapotrzebowania na tego typu wiedzę.

Ad. j) Wpływ opracowanego rozwiązania na automatyzację procesów charakterystycznych dla innej dziedziny wiedzy, jaką jest rolnictwo precyzyjne. Opracowane rozwiązanie przyczynia się do zwiększenia wydajności, a także oszczędności zasobów i optymalizacji monitorowania upraw, w porównaniu do ręcznego określania badanych parametrów z analogiczną dokładnością 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 zestawu oznaczonych obrazów RGB pozyskiwanych w początkowych sezonach wegetacyjnych, w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, najlepsze wyniki według metryki F1-score uzyskał model HTC_r101, za nim był model HTC_x101, następnie HTC_r50, a najgorsze wyniki spośród badanych modeli osiągnął Mask2Former (wyniki te dotyczą powiększonego zbioru treningowego, o czym mowa w rozdziale *4. Detekcja i klasyfikacja faz rozwoju kukurydzy w skali BBCH z wykorzystaniem obrazowania RGB i obrazowania z kamery wielospektralnej MicaSense RedEdge MX Dual*).
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów wielospketralnych z kamery MicaSense RedEdge MX Dual pozyskiwanych w początkowych sezonach wegetacyjnych, 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 MicaSense RedEdge MX Dual w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH oraz przy wykorzystaniu klasyfikatora głosującego najlepsze wyniki uzyskał model HTC_r50, za nim był model HTC_x101, następnie HTC_r101, a najgorsze wyniki uzyskał model Mask2Former.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB oraz wyniki tychże modeli trenowanych na podstawie obrazów z kamery MicaSense RedEdge MX Dual i zastosowaniu klasyfikatora głosującego, w celu realizacji zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH, model HTC_r101 uzyskał lepszą skuteczność dla danych wielospketralnych przy zastosowaniu klasyfikatora głosującego, model HTC_r50 uzyskał lepszą skuteczność dla danych wielospketralnych przy zastosowaniu klasyfikatora głosującego, model HTC_x101 również uzyskał lepszą skuteczność dla danych wielospketralnych przy zastosowaniu klasyfikatora głosującego, zaś model Mas2Former uzyskał lepszą skuteczność dla danych RGB.
- Porównując globalne wyniki dla zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy w skali BBCH, na podstawie obrazów RGB i obrazów wielospektralnych z kamery MicaSense RedEdge MX Dual, pozyskiwanych w tych samych sezonach wegetacyjnych, najlepsze wyniki osiągnięto przy zastosowaniu obrazowania wielospektralnego, użyciu metody klasyfikatora głosującego oraz modelu HTC_r50.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB pozyskanych w późniejszym sezonie wegetacyjnym, w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, najlepsze wyniki uzyskał model HTC_r50, za nim był model HTC_x101, następnie HTC_r101, a najgorsze wyniki spośród badanych modeli osiągnął Mask2Former.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów wielospektralnych z kamery MapIR Survey3 Red+Green+NIR, pozyskanych w późniejszym sezonie wegetacyjnym, w celu detekcji i klasyfikacji faz rozwojowych kukurydzy w skali BBCH, najlepsze wyniki uzyskał model HTC_r50, za nim znalazł się model HTC_r101, następnie HTC_x101, zaś najgorsze wyniki uzyskał model Mask2Former.

- Porównując wyniki uzyskane z wykorzystaniem modeli trenowanych na podstawie obrazów z kamery RGB oraz modeli trenowanych na podstawie obrazów z kamery wielospektralnej MapIR Survey3 Red+Green+NIR w celu realizacji zadania związanego z detekcją i klasyfikacją faz rozwojowych kukurydzy w skali BBCH, najlepsze wyniki uzyskano przy zastosowaniu obrazowania RGB oraz architektury HTC_r50.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB pozyskanych w celu detekcji i klasyfikacji poziomów nawodnienia kukurydzy, najlepsze wyniki uzyskał model ConvNeXt Small, za nim był model Vision Transformer, następnie Swin Transformer, EfficientNet-B3, a najgorsze wyniki spośród badanych modeli osiągnął ResNet50.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów z kamery wielospektralnej MapIR Survey3 Red+Green+NIR pozyskanych w celu detekcji i klasyfikacji poziomów nawodnienia kukurydzy, najlepsze wyniki uzyskał algorytm Swin Transformer, 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 najlepsze wyniki uzyskano dzięki zastosowaniu obrazowania RGB oraz architektury ConvNeXt Small.
- Porównując wyniki poszczególnych modeli trenowanych na podstawie obrazów RGB, w celu detekcji i klasyfikacji porażenia kukurydzy, najlepsze wyniki uzyskał model Swin Transformer, za nim był model Vision Transformer, następnie ConvNeXt Small, EfficientNet-B3, a najgorsze wyniki spośród badanych modeli osiągnął ResNet50.
- Dzięki przeprowadzonym badaniom oraz opracowanemu systemowi możliwa była automatyzacja procesów monitorowania parametrów kondycji kukurydzy. To pokazuje, że systemy AI mogą skutecznie pracować na robotach polowych, usprawniając systemy zarządzania w produkcji rolnej zgodnie z ideą rolnictwa precyzyjnego.

9.2. Rekomendacje dla przyszłych badań

Opracowane rozwiązanie wykorzystuje zbiory oznaczonych danych RGB, zbiór oznaczonych danych wielospketralnych z kamery MicaSense RedEdge MX Dual oraz zbiory oznaczonych danych wielospektralnych z kamery MapIR Survey3 Red + Green + NIR.

Niewątpliwym kierunkiem rozwoju w tym przypadku jest rozszerzenie tychże zbiorów o dodatkowe obrazy i oznaczenie ich według przyjętych zasad omówionych w niniejszej rozprawie. Zwiększenie liczności zbiorów treningowych pozwoli na uzyskanie jeszcze wyższej skuteczności omawianych modeli uczenia głębokiego, gdyż wpłynie to 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, zadaniu związanym z detekcją i klasyfikacją poziomów nawodnienia kukurydzy oraz zadaniu związanym z detekcją i klasyfikacją porażenia kukurydzy wybranymi patogenami.

W przypadku zbiorów danych cennym usprawnieniem będzie także kontrolowane porażenie roślin wybranymi patogenami na poletku testowym. Dotychczas analiza porażenia kukurydzy opierała się na zewnętrznym zbiorze danych [Kaggle] zawierającym obrazy RGB. Dzięki nowemu podejściu możliwe będzie stworzenie dwóch autorskich zbiorów danych: jednego zawierającego wyłącznie obrazowanie RGB oraz drugiego zawierającego wyłącznie obrazowanie wielospektralne. Obrazowanie wielospektralne może ujawniać więcej różnic między okazami roślin zdrowych i porażonych niż obrazowanie RGB, co pozwoli na bardziej precyzyjną analizę wpływu patogenów.

Obecne rozwiązanie opiera się na wykorzystaniu w ramach zadania związanego z detekcją i klasyfikacją faz rozwoju kukurydzy w skali BBCH, następujących algorytmów: HTC_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 dotychczas wykorzystywanych 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 nic 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 zobrazowań, ich oznaczaniem, trenowaniem zestawów 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 faz rozwoju i stanu zdrowia roślin uprawnych może być rozszerzona o nowe gatunki kluczowych roślin uprawnych, jakimi są np. żyto, pszenica, owies, proso itp.

Opracowane rozwiązanie dotyczące detekcji i klasyfikacji faz rozwoju kukurydzy oraz detekcji i klasyfikacji poziomów nawodnienia kukurydzy 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 zdolność adaptacji i wszechstronność w innych regionach świata.

Opracowane rozwiązanie mogłoby zostać zintegrowane z technologiami IoT, co umożliwiłoby jeszcze lepszy monitoring w czasie rzeczywistym oraz zbieranie i analizę danych z wielu pól jednocześnie. Systemy takie mogłyby także komunikować się bezpośrednio z urządzeniami na polu uprawnym, jeszcze bardziej automatyzując proces zarządzania uprawami. Nowoczesne technologie w rolnictwie, takie jak sztuczna inteligencja, mają potencjał, aby rewolucjonizować uprawy i wspierać rolników i przedsiębiorców w ich codziennych działaniach, co podkreśla znaczenie dalszych badań w tej dziedzinie i czyni je wartościowymi.

Spis rysunków

Spis tabel

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 *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (pp. 265-283).

[Anaconda, 2020] Anaconda. (2020). Anaconda Software Distribution. https://anaconda.com, dostęp: wrzesień 2024

[Albon, 2019] Albon, C. (2019). *Uczenie maszynowe w Pythonie. Receptury*. Helion.

[Bera, i in., 2024] Bera, A., Bhattacharjee, D., & Krejcar, O. (2024). PND-Net: Plant nutrition deficiency and disease classification using graph convolutional network. *Scientific Reports, 14*, 15537.

[Bishop, 2006] Bishop, C. (2006). *Pattern recognition and machine learning.* Springer.

[Bock i in., 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.

[Brown, 2015] Brown, C. (2015). Challenges in color image processing: Lighting conditions and noise. *IEEE Transactions on Image Processing, 24*(4), 1342-1353.

[Chen i in., 2019] Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, B., ... & Loy, C. C. (2019). Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (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. (2019). *Deep learning. Praca z językiem Python i biblioteką Keras.* Helion.

[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. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 886-893). IEEE.

[Deng, 2018] Deng, L. (2018). Cloud computing for deep learning: Opportunities and concerns. *IEEE Signal Processing Magazine, 35*(3), 121-134.

[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. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 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.

[Gradio, 2024] Strona internetowa Gradio, https://www.gradio.app/, dostęp: wrzesień 2024

[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 (CVPR)* (pp. 770-778).

[He i in., 2017] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (pp. 2961-2969).

[Helion, 2024] Skala BBCH dla kukurydzy, https://pdf.helion.pl/e_1wwu/e_1wwu.pdf, dostęp: wrzesień 2024

[Huang i in., 2019] Huang, Y., Liao, B., Cheng, S., & Wu, H. (2019). GPipe: Efficient training of giant neural networks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)* (pp. 103-112).

[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 University Press.

[Jouppi i in., 2017] Jouppi, N. P., Young, C., Patil, N., et al. (2017). In-datacenter performance analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*.

[Kaggle] Zbiór obrazów przedstawiających zdrową oraz porażoną kukurydzę, https://www.kaggle.com/datasets/, dostęp: listopad 2022.

[Kamilaris, Prenafeta-Boldú, 2018] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture, 147,* 70-90.

[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 *Proceedings of the 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] Krzanowski, W. J. (2000). *Principles of multivariate analysis: A user's perspective.* Oxford University Press.

[Label Studio, 2024] Strona internetowa Label Studio, https://labelstud.io, dostęp: wrzesień, 2024.

[Lancashire i in., 1991] Lancashire, P. D., Bleiholder, H., Van Den Boom, T., Langelüddeke, P., Stauss, R., Weber, E., & Witzenberger, 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 (CVPR)* (pp. 2117-2125).

[Liu i in., 2019] Liu, M., et al. (2019). Maize seedling detection using UAV RGB imagery and YOLO algorithm. *Remote Sensing, 11*(10), 1190.

[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, 240*, 111690.

[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 (ICCV)* (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 (CVPR)* (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, 54*, 689-709.

[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.

[MapIR, 2024] Survey3W Camera - Red+Green+NIR (RGN, NDVI), https://www.mapir.camera/en-gb/products/survey3w-camera-red-green-nir-rgn-ndvi?srsltid=AfmBOoryiA8OiIMy16-PXO8BDBtXzwJhSYgAs96hvoCA5eE8Mq33POib, dostęp: wrzesień 2024.

[Meier, 2001] Meier, U. (2001). *Growth Stages of Mono and Dicotyledonous Plants. BBCH Monograph.* Federal Biological Research Centre for Agriculture and Forestry, Bonn.

[MicaSense, 2024a] Specifications RedEdge-MX Dual Camera Imaging System by MicaSense – AEROMOTUS, https://www.aeromotus.com/product/rededge-mx-dual-camera-imaging-system-by-micasense/, dostęp: wrzesień 2024.

[MicaSense, 2024b] Przewodnik integracji RedEdge-MX Dual Camera System - Integration Guide – MicaSense Knowledge Base, https://support.micasense.com/hc/en-us/articles/360037369993-RedEdge-MX-Dual-Camera-System-Integration-Guide, dostęp: wrzesień 2024.

[MMDetection, 2020] Contributors to MMDetection. (2020). MMDetection: OpenMMLab Detection Toolbox and Benchmark, https://github.com/open-mmlab/mmdetection, dostęp: wrzesień 2022

[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, karty graficzne z rodziny GeForce RTX 3080, https://www.nvidia.com/pl-pl/geforce/graphics-cards/30-series/rtx-3080-3080ti/, dostęp: wrzesień 2024.

[Osinga, 2019] Osinga, D. (2019). *Deep Learning. Receptury.* Helion.

[paperswithcode.com, 2024] Instance segmentation on COCO test-dev, https://paperswithcode.com/sota/instance-segmentation-on-coco, dostęp: wrzesień 2024.

[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, 32*, 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] Raschka, S. (2018). *Python. Uczenie maszynowe.* Helion.

[Rosenblatt, 1957] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automation.* Cornell Aeronautical Laboratory.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*(6), 386-408.

[Rumelhart i in., 1986] Rumelhart, D. E., Hinton, G. E., & 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]. *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, 10*(10), 1422.

[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.* Springer.

[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, J. (2023). The use of AI to determine the condition of corn in a field robot that meets the requirements of precision farming. In M. Ganzha et al. (Eds.), *Communication papers of the 18th Conference on Computer Science and Intelligence Systems, Annals of Computer Science and Information Systems, 37,* 313-321. Warszawa: Polskie Towarzystwo Informatyczne. DOI: 10.15439/2023f3649.

[Stypułkowska, 2023b] Stypułkowska, J. (2023). Zgłoszenie patentowe nr P.446025: *Sposób oraz urządzenie do kontroli parametrów jakościowych upraw, w szczególności upraw*

kukurydzy. Złożone do Urzędu Patentowego Rzeczypospolitej Polskiej. Zgłaszający: Sieć Badawcza Łukasiewicz – Instytut Lotnictwa.

[Stypułkowska, Rokita, 2025] Stypułkowska, J., & Rokita, P. (2025). Classification of Maize Growth Stages Using Deep Neural Networks with Voting Classifier. *Machine Graphics and Vision* (in press).

[Taigman i in., 2014] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1701–1708.

[Tan, Le, 2019] Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)* (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 i in., 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.

[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] Weidman, S. (2019). *Deep learning from scratch: Building with Python from first principles.* Helion. (Tłum. T. Walczak, *Uczenie głębokie od zera. Podstawy implementacji w Pythonie*, 2020).

[White, 2019] White, D. (2019). Advances in image noise reduction techniques. *Journal of Visual Computing, 9*(2), 85-101.

[Widrow i in., 1960] Widrow, B., et al. (1960). An adaptive "Adaline" neuron using chemical "memistors". *Number Technical Report* 1553-2, Stanford Electronics Laboratories, Stanford, California, October 1960.

[Wightman, 2019] Wightman, R. (2019). *PyTorch Image Models*, https://github.com/rwightman/pytorch-image-models, dostęp: wrzesień, 2024

[Wolfert i in., 2017] Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M.-J. (2017). Big Data in Smart Farming – A Review. *Agricultural Systems, 153*, 69-80.

[Xie i in., 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)*, 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, 176*, 105660.

[Yao i in., 2024] Yao, Y., Yue, J., Liu, Y., Yang, H., Feng, H., Shen, J., Hu, J., & Liu, Q. (2024). Classification of maize growth stages based on phenotypic traits and UAV remote sensing. *Agriculture, 14*, 1175.

[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.

[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.

[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*. *16*(1), 1-15.

[Zocca i in., 2018] Zocca V., Spacagna G., Slater D., Roelants P. (2018), *Deep Learning. Uczenie głębokie z językiem Python. Sztuczna inteligencja i sieci neuronowe*, Helion.

Załączniki

Załączniki dostępne są na dysku:

https://drive.google.com/drive/folders/1abKi-9uiJs9IUeloVgeU7UsS68LWw7uG?usp=drive_link

Kod umieszczony na tymże dysku (załącznik B) oraz zbiory danych (załącznik C) są udostępnione tylko i wyłącznie na potrzeby recenzji, za zgodą Pana mgr inż. Mariusza Kacprzaka – Kierownika Działu Teledetekcji Sieci Badawczej Łukasiewicz – Instytutu Lotnictwa. Po zakończeniu procesu recenzji kod oraz zbiory danych zostaną usunięte z dysku.

A. Publikacje, dyplom wyróżnienia, informacja o złożonym wniosku patentowym

A1 – Artykuł naukowy:

Stypułkowska, J. (2023). The use of AI to determine the condition of corn in a field robot that meets the requirements of precision farming. In M. Ganzha, et al. (Eds.), *Communication Papers of the 18th Conference on Computer Science and Intelligence Systems, Annals of Computer Science and Information Systems, 37,* 313-321. Warszawa: Polskie Towarzystwo Informatyczne. https://doi.org/10.15439/2023f3649.

A2 – Artykuł naukowy:

Stypułkowska, J., & Rokita, P. (2025). Classification of Maize Growth Stages Using Deep Neural Networks with Voting Classifier. *Machine Graphics and Vision*. – artykuł został zaakceptowany do publikacji i zostanie opublikowany w 2025 roku.

A3 – Dyplom wyróżnienia:

Dyplom uzyskany na konferencji naukowej 18th Conerence 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 – Informacja o złożeniu wniosku patentowego:

W niniejszym załączniku przedstawiono informacje o złożeniu wniosku patentowego w Urzędzie Patentowym Rzeczypospolitej Polskiej. Załącznik zawiera numer zgłoszenia, datę oraz nazwę zgłaszającego, ale nie zawiera pełnej treści zgłoszenia patentowego ze względu na poufność danych.

A5 – Artykuł popularnonaukowy:

Artykuł opublikowany na portalu Farming Future Food, w którym opisano wyniki moich badań.

Tytuł artykułu: *Artificial intelligence can provide accurate overview of corn development and health*,

https://farmingfuturefood.com/artificial-intelligence-can-provide-accurate-overview-of-corn-development-and-health/

B. Kod opracowany i wykorzystywany w ramach realizacji badań

W załączniku tym udostępniono kod opracowany i wykorzystany w ramach realizacji badań.

C. Zbiory danych wykorzystane w ramach realizacji badań

W załączniku tym udostępniono zbiory danych wykorzystane w ramach realizacji badań.