POLITECHNIKA GDAŃSKA

Imię i nazwisko autora rozprawy: **Arkadiusz Kwasigroch**
Dyscyplina naukowa: Automatyka, Elektronika, Elektrotechnika i Technologie Kosmiczne

ROZPRAWA DOKTORSKA

Tytuł rozprawy w języku polskim: **Poprawa jakości klasyfikacji głębokich sieci neuronowych poprzez optymalizację ich struktury i dwuetapowy proces uczenia**

Tytuł rozprawy w języku angielskim: **Improving the quality of deep neural network classification through optimization of their structure and a two-stage learning process**

Promotor podpis	~~Drugi promotor~~ ~~podpis~~
dr hab. inż. Michał Grochowski, prof. PG	
~~Promotor pomocniczy~~ ~~podpis~~	~~Kopromotor~~ ~~podpis~~

Gdańsk, rok 2023

POLITECHNIKA GDAŃSKA

OŚWIADCZENIE

Autor rozprawy doktorskiej: mgr inż. Arkadiusz Kwasigroch

Ja, niżej podpisany, oświadczam, iż jestem świadomy, że zgodnie z przepisem art. 27 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 poz. 1062), uczelnia może korzystać z mojej rozprawy doktorskiej zatytułowanej: „Poprawa jakości klasyfikacji głębokich sieci neuronowych poprzez optymalizację ich struktury i dwuetapowy proces uczenia" do prowadzenia badań naukowych lub w celach dydaktycznych.

Świadomy odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych i konsekwencji dyscyplinarnych określonych w ustawie Prawo o szkolnictwie wyższym i nauce (Dz.U.2021.478 t.j.), a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana rozprawa doktorska została napisana przeze mnie samodzielnie.

Oświadczam, że treść rozprawy opracowana została na podstawie wyników badań prowadzonych pod kierunkiem i w ścisłej współpracy z promotorem dr. hab. inż. Michałem Grochowskim, prof. PG.

Niniejsza rozprawa doktorska nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem stopnia doktora.

Wszystkie informacje umieszczone w ww. rozprawie uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami, zgodnie z przepisem art. 34 ustawy o prawie autorskim i prawach pokrewnych.

Potwierdzam zgodność niniejszej wersji pracy doktorskiej z załączoną wersją elektroniczną.

Gdańsk, dnia .... podpis doktoranta

Ja, niżej podpisany, wyrażam zgodę na umieszczenie ww. rozprawy doktorskiej w wersji elektronicznej w otwartym, cyfrowym repozytorium instytucjonalnym Politechniki Gdańskiej.

Gdańsk, dnia .... podpis doktoranta

[1] Art. 27. 1. Instytucje oświatowe oraz podmioty, o których mowa w art. 7 ust. 1 pkt 1, 2 i 4–8 ustawy z dnia 20 lipca 2018 r. – Prawo o szkolnictwie wyższym i nauce, mogą na potrzeby zilustrowania treści przekazywanych w celach dydaktycznych lub w celu prowadzenia działalności naukowej korzystać z rozpowszechnionych utworów w oryginale i w tłumaczeniu oraz zwielokrotniać w tym celu rozpowszechnione drobne utwory lub fragmenty większych utworów.

[2] W przypadku publicznego udostępniania utworów w taki sposób, aby każdy mógł mieć do nich dostęp w miejscu i czasie przez siebie wybranym korzystanie, o którym mowa w ust. 1, jest dozwolone wyłącznie dla ograniczonego kręgu osób uczących się, nauczających lub prowadzących badania naukowe, zidentyfikowanych przez podmioty wymienione w ust. 1.

POLITECHNIKA GDAŃSKA

OPIS ROZPRAWY DOKTORSKIEJ

Autor rozprawy doktorskiej: mgr inż. Arkadiusz Kwasigroch

Tytuł rozprawy doktorskiej w języku polskim: Poprawa jakości klasyfikacji głębokich sieci neuronowych poprzez optymalizację ich struktury i dwuetapowy proces uczenia

Tytuł rozprawy w języku angielskim: Improving the quality of deep neural network classification through optimization of their structure and a two-stage learning process

Język rozprawy doktorskiej: polski

Promotor rozprawy doktorskiej: dr hab. inż. Michał Grochowski, prof. PG

Data obrony:

Słowa kluczowe rozprawy doktorskiej w języku polskim: uczenie maszynowe, sztuczna inteligencja, głębokie uczenie

Słowa kluczowe rozprawy doktorskiej w języku angielskim: machine learning, artificial intelligence, deep learning

Streszczenie rozprawy w języku polskim: W pracy doktorskiej podjęto problem realizacji algorytmów głębokiego uczenia w warunkach deficytu danych uczących. Głównym celem było opracowanie podejścia optymalizującego strukturę sieci neuronowej oraz zastosowanie uczeniu dwuetapowym, w celu uzyskania mniejszych struktur, zachowując przy tym dokładności. Proponowane rozwiązania poddano testom na zadaniu klasyfikacji znamion skórnych na znamiona złośliwe i łagodne. W pierwszym etapie badań dokonano analizy wpływu elementów architektury oraz metod uczenia na wyniki. Następnie, w oparciu o uzyskane wyniki, zaproponowano system automatycznego doboru struktury sieci neuronowej oparty o algorytmy ewolucyjne i modyfikacje sieci zachowujące funkcję. Zastosowanie algorytmu umożliwiło redukcję liczby parametrów o 98%, w porównaniu do popularnych sieci VGG. Kolejnym etapem badań była analiza metodami wstępnego uczenia. Przeprowadzono analizę zastosowania wstępnego uczenia w sposób samonadzorowany oraz w sposób nadzorowany. W badaniach wykazano, że połączenie tych metod przynosi lepsze wyniki niż zastosowanie tylko wstępnego uczenia w sposób nadzorowany. Ostatnim etapem badań była integracja metod doboru struktury z metodami wstępnego uczenia. Przeprowadzone badania potwierdziły, że odpowiedni dobór struktury i metody uczenia mają istotny wpływ na dokładność i efektywność modeli.

Streszczenie rozprawy w języku angielskim: The doctoral thesis addressed the problem of implementing deep learning algorithms in conditions of limited training data. The main goal was to develop an approach that optimizes the structure of a neural network and applies two-stage learning to achieve smaller structures while maintaining accuracy. The proposed solutions were tested on the task of classifying skin lesions as malignant or benign. In the first stage of the research, an analysis was conducted on the impact of architectural elements and learning methods on the results. Based on the obtained results, an automatic neural architecture search system was proposed, which relied on evolutionary algorithms and function preserving transformations. The application of the algorithm enabled a reduction of parameters by 98% compared to popular VGG networks. The next stage of the research involved an analysis of pre-training methods. An examination was carried out on the application of self-supervised supervised pre-training. The study demonstrated that combining these methods yields better results than using only supervised pre-training. The final stage of the research involved integrating the neural architecture search with pre-training methods. The conducted research confirmed that the appropriate selection of structure and learning methods significantly impact the accuracy and efficiency of models.

Podziękowania

Powstanie tej pracy nie byłoby możliwe bez wsparcia wielu osób.

Przede wszystkim chciałbym podziękować mojemu promotorowi prof. Michałowi Grochowskiemu za liczne wskazówki, cierpliwość oraz całą pomoc, jaką mi udzielił przez cały okres trwania doktoratu.

Dziękuję również wszystkim koleżankom i kolegom z Katedry Inteligentnych Systemów Sterowania i Wspomagania Decyzji, a w szczególności Agnieszce, Marysi, Zuzannie i Tomkowi, z którymi dzieliłem pokój.

Pragnę także wyrazić podziękowanie wszystkim pracownikom Wydziału Elektrotechniki i Automatyki oraz Politechniki Gdańskiej, dzięki którym realizacja mojej pracy doktorskiej była możliwa.

Dziękuję Ministerstwu Edukacji i Nauki za udzielone wsparcie finansowe w ramach projektu „Opracowanie nowych struktur głębokich sieci neuronowych do analizy obrazów 2d, dla celów wspomagania decyzji", co umożliwiło mi prowadzenie badań.

Chciałbym również podziękować prof. Christophowi Lippertowi oraz wszystkim koleżankom i kolegom z Instytutu Hasso Plattnera w Poczdamie. Staż w tym miejscu był dla mnie niezwykle wartościowym doświadczeniem.

Wyrazy podziękowania kieruję do społeczności MLGdańsk, a w szczególności do organizatorów: Marcina, Roberta i Adama. Dziękuję za słowa motywacji podczas pisania pracy, liczne spotkania i interesujące dyskusje.

Dziękuję także koleżankom i kolegom z firmy Brainscan, gdzie miałem możliwość rozwoju moich umiejętności przy praktycznych zastosowaniach algorytmów sztucznej inteligencji.

Dziękuję wszystkim znajomym a w szczególności Karolowi, Kamili, Bartkowi, Marcinowi, Adamowi, oraz Przemkowi. Dziękuję za wspólnie spędzony czas i słowa motywacji.

Na końcu chciałbym wyrazić głęboką wdzięczność moim rodzicom, którzy wspierali mnie w trakcie doktoratu i na których zawsze mogłem liczyć. Wam dedykuję tę pracę.

Spis treści

1. Wprowadzenie

1.1. Motywacja

Rok 2023 jest przełomowy dla wzrostu zainteresowania sztuczną inteligencją, głównie za sprawą udostępnienia rozwiązania ChatGPT przez firmę OpenAI [1]. Rozwiązanie przyciągnęło znaczną uwagę i wywołało szeroką debatę na temat sztucznej inteligencji, płynących korzyści i zagrożeń. Niektórzy są zdania, że sztuczna inteligencja może przynieść zmiany, porównywalne do tych, które wywołało pojawienie się komputera, internetu czy nawet elektryczności. Warto jednak zauważyć, że dynamiczny rozwój sztucznej inteligencji jest obserwowany przez środowisko naukowe i akademickie od ponad 10 lat.

Szczególnie szybko rozwijane są algorytmy głębokiego uczenia, będące częścią szerokiej gamy narzędzi tzw. uczenia maszynowego, a szerzej – sztucznej inteligencji. Głębokie sieci neuronowe są algorytmami, które ewoluowały z klasycznych sieci neuronowych. Jako przełomowy moment w rozwoju tych algorytmów uznaje się zwycięstwo rozwiązania wykorzystującego te modele w jednym z najważniejszych konkursów w dziedzinie rozpoznawania obrazów – ImageNet, w 2012 roku. Jednym z celów konkursu ImageNet było stworzenie dużego, otwartego zbioru danych obrazowych z towarzyszącymi etykietami, który miał służyć naukowcom do projektowania i oceny algorytmów rozpoznawania obrazów.

Jednym z zadań konkursowych było przypisanie obiektom na obrazie jednej z tysiąca klas, reprezentujących m.in. obiekty z życia codziennego takie jak zwierzęta, rośliny, środki transportu, budynki, czy przedmioty domowe. Rozwiązanie oparte o głębokie, konwolucyjne sieci neuronowe uzyskało dokładność wyższą o około 10 punktów procentowych od kolejnego rozwiązania, w którym zastosowano klasyczne algorytmy (zbiór ekstraktorów cech wraz z algorytmem maszyn wektorów nośnych

(ang. Support Vector Machines – SVM)) [2]. Od tego momentu nastąpił wzrost zainteresowania algorytmami głębokiego uczenia, a zwycięskie rozwiązania konkursowe w kolejnych latach w większości oparte były na głębokich sieciach neuronowych. Niektóre z rozwiązań opracowanych na potrzeby konkursu zostały szeroko zaadaptowane w różnych zastosowaniach przetwarzania i analizy obrazu, a zbiór danych wykorzystywany w konkursie stał się, i nadal jest, ważnym punktem odniesienia (ang. *benchmark*) dla oceny nowych rozwiązań takich jak architektury czy metody uczenia.

Poza obszarem przetwarzania obrazów, w którym głębokie sieci neuronowe osiągnęły dominującą pozycję, algorytmy te uzyskały również imponujące wyniki w innych dziedzinach, takich jak analiza dźwięku, mowy i tekstu. Algorytmy te znalazły zastosowanie w wielu obszarach takich jak medycyna [3], robotyka [4], pojazdy autonomiczne [5], chemia [6], [7], rolnictwo [8], a nawet sztuka [9]. Przykładami rozwiązań, które pokazują znaczący potencjał, skuteczność, a nawet kreatywność, są wspomniane na wstępie modele językowe takie jak ChatGPT oparte o architekturę transformer [10]. Modele te wykazują imponującą zdolność rozumienia języka, generowania odpowiedzi, interakcji z ludźmi oraz podejmowania decyzji. Innym przykładem rozwiązania opartego o głębokie sieci neuronowe, które przyciągnęło uwagę mediów, był system DALL-E2 [11], zdolny do generowania realistycznych i abstrakcyjnych obrazów na podstawie tekstowego opisu.

Opracowanie i implementacja tak skutecznych systemów jest możliwe dzięki wieloletnim i systematycznym pracom wielu badaczy, zespołów naukowych i firm. Rozwój architektur sieci neuronowych [12]–[14], metody uczenia [15], [16], oraz innych technik wspomagających działania sieci [17]–[19] spowodował zwiększenie dokładności i wydajności modeli neuronowych, a także zmniejszył zapotrzebowanie na dane uczące. Przykładem postępu osiągniętego w rozwiązaniach algorytmicznych, może być rozwój algorytmów doboru struktury sieci neuronowej, gdzie czas doboru struktury został skrócony z 2000 dni do zaledwie czterech dni, przy równoczesnym zachowaniu dokładności [20]. Innym przykładem może być ograniczenie zapotrzebowania modeli na dane oznaczone poprzez zastosowanie danych nieoznaczonych w uczeniu samonadzorowanym [21].

Mimo wielu imponujących osiągnięć głębokiego uczenia, realizacja systemów opartych o te rozwiązania, skuteczne uczenie i kompleksowa ewaluacja, niesie ze sobą szereg wyzwań. Wśród najważniejszych wyzwań i problemów można wyróżnić:

trudności w doborze odpowiedniego rodzaju i typu sieci neuronowej, dobór architektury oraz hiperparametrów, ograniczona dostępność odpowiednio oznaczonych danych, skomplikowana i wieloznaczna interpretacja działań podejmowanych przez sieci neuronowe, czy potencjalne błędy wynikające z nieprawidłowej interpretacji danych wejściowych.

Intencją badań prowadzonych w ramach niniejszej pracy oraz wspierającego badania Diamentowego grantu „Opracowanie nowych struktur głębokich sieci neuronowych do analizy obrazów 2d, dla celów wspomagania decyzji", była próba rozwiązania lub zminimalizowania części z wymienionych problemów związanych z rozwojem złożonych algorytmów, jakimi są głębokie sieci neuronowe. W szczególności, w pracy podjęto badania nad rozwojem algorytmów automatycznego doboru struktury głębokich sieci neuronowych oraz metod uczenia dwuetapowego opartego o nadzorowane i samonadzorowane metody wstępnego uczenia, celem poprawy jakości i skuteczności klasyfikacji. Praktycznym zagadnieniem analizowanym w pracy jest ważne społecznie zagadnienie diagnostyki znamion skórnych pod kątem wykrywania znamion złośliwych.

1.2. Cele i zakres pracy

Głębokie uczenia zmieniło podejście do projektowania rozwiązań uczenia maszynowego. Klasyczne podejścia bazowały na tzw. inżynierii cech, czyli przetwarzaniu informacji na potrzeby algorytmów uczenia maszynowego. Próba uczenia takich algorytmów z wykorzystaniem nieprzetworzonych danych, zwłaszcza wielowymiarowych, takich jak obrazy, prowadziła zazwyczaj do słabych wyników. Dopiero odpowiednie przetworzenie i znaczna redukcja rozmiaru danych wejściowych pozwalały na poprawę skuteczności. Skuteczna inżynieria cech wymaga szerokiej wiedzy i doświadczenia w zakresie rodzaju danych (np. obraz czy dźwięk), jak również w konkretnym obszarze zastosowań. Na przykład, opracowanie systemu analizy badań rentgenowskich, wymaga z jednej strony znajomości technik i algorytmów przetwarzania obrazu, a z drugiej strony znajomości cech chorób, które algorytm ma diagnozować. Rozwiązania oparte na ręcznej inżynierii cech i prostych algorytmach uczenia maszynowego działały tylko w określonych warunkach, a ich możliwości generalizacji były ograniczone. Na przykład, zmiana warunków oświetlenia mogła negatywnie wpływać na dokładność systemu. Zaletą klasycznych rozwiązań

jest jednak mniejsze zapotrzebowanie na dane i moc obliczeniową oraz większa transparentność i możliwość interpretacji osiągniętych wyników. Wadą natomiast jest gorsza dokładność oraz pracochłonny i wymagający specjalistycznej wiedzy proces przygotowania metod ekstrakcji cech. Rozwój algorytmów głębokiego uczenia nie spowodował jednak, że opisywane rozwiązania odeszły w zapomnienie. Klasyczne metody uczenia maszynowego nadal z powodzeniem stosowane są w wielu zadaniach, które nie wymagają stosowania aż tak złożonych i wymagających obliczeniowo modeli.

Głębokie sieci neuronowe, w przeciwieństwie do rozwiązań klasycznych, są w stanie działać na wielowymiarowych, słabo przetworzonych danych bez konieczności stosowania algorytmów ekstrakcji cech. Głęboka sieć neuronowa w efekcie uczenia na dużych zbiorach danych może wydobyć z nich odpowiednie cechy, poprzez kolejne warstwy, tworząc reprezentacje danych o coraz większej złożoności. Pierwsze warstwy mogą wykrywać bezpośrednio na obrazie proste cechy takie jak podstawowe kształty czy kolory. Kolejna warstwa na podstawie tej reprezentacji może wykrywać bardziej złożone elementy. Natomiast ostatnie warstwy, na podstawie reprezentacji generowanych przez warstwy poprzednie, mogą wykrywać cechy o najwyższym poziomie takie jak obecność obiektów czy ich części np. koła na zdjęciu samochodu, czy oczu na zdjęciu twarzy. Realizacji ekstrakcji cech przez sieć neuronową warunkowana jest przez jej parametry, które są dobierane automatycznie w trakcie uczenia. Stanowi to zasadniczą różnicę w porównaniu z klasycznymi rozwiązaniami, gdzie algorytmy ekstrakcji cech i ich ustawienia zazwyczaj dobierane są ręcznie. Dzięki realizacji ekstrakcji cech przez model, głębokie sieci neuronowe mogą analizować słabo przetworzone dane bez konieczności projektowania algorytmów ekstrakcji cech. Wadą tego podejścia jest jednak to, że efektywne uczenie reprezentacji danych wymaga zastosowania dużego zbioru uczącego.

Mimo wysokiej skuteczności w wielu obszarach, tworzenie i stosowanie systemów opartych o głębokie uczenie nadal wiąże się z licznymi ograniczeniami. Konkurs ImageNet przyczynił się do rozwoju głębokich sieci neuronowych, ale zwycięskie rozwiązania były przystosowane do konkretnych zadań konkursowych, które cechowały się dużym zbiorem treningowym i równomiernym rozkładem klas. W praktycznych zastosowaniach dostępne zbiory są znacznie mniejsze i słabiej zbilansowane. Konieczne jest zatem przystosowanie modeli i algorytmów głębokiego uczenia do zadań małej skali, gdzie zbiór uczący oraz liczba rozpatrywanych klas jest

mniejsza, a realizowane zadanie jest trudne i złożone. Niniejsza rozprawa doktorska stanowi próbę znalezienia odpowiedzi na te problemy.

Jednym z głównych tematów, rozpatrywanych w ramach niniejszej pracy doktorskiej, jest dobór architektury sieci neuronowej oraz hiperparametrów. Zazwyczaj, dobór struktury i hiperparametrów dokonywany jest ręcznie w sposób eksperymentalny. Dobór eksperymentalny był łatwiejszy w rozwiązaniach klasycznych, gdzie czas treningu był znacznie krótszy, a przestrzeń poszukiwań dużo mniejsza. W głębokich sieciach neuronowych, gdzie czas uczenia może wynosić od kilku godzin do nawet kilku dni, możliwość testowania wielu rozwiązań jest ograniczona. Ponadto głębokie sieci neuronowe są dużo większe, co wiąże się z doborem większej liczby hiperparametrów. Wraz z rozwojem głębokiego uczenia powstało wiele rozwiązań, co dodatkowo zwiększa przestrzeń poszukiwań i utrudnia wybór odpowiednich metod. Jednym z rozwiązań jest stosowanie znanych architektur takich ja te opracowane na potrzeby konkursu ImageNet. Są to jednak rozwiązania, które zostały przystosowane do zadań dużej skali. Użycie sieci zbyt złożonej do danego zadania prowadzi do wzrostu wymagań obliczeniowych, wydłużenia czasu obliczeń, oraz wzrostu zapotrzebowania na pamięć co ogranicza stosowanie rozwiązań na mniej wydajnym sprzęcie. Dodatkowo zbyt duża struktura sieci neuronowej prowadzi do nadmiernego dopasowania danych i pogarsza możliwość generalizacji. Odpowiedzią na wymienione problemy są metody automatycznego doboru struktury, które umożliwiają dobór struktury optymalnej dla danego zadania.

Kolejnym zagadnieniem podjętym w rozprawie jest zmniejszenie zapotrzebowania na oznaczone dane uczące. Głębokie sieci neuronowe potrzebują dużej ilości danych w celu skutecznego uczenia, co stanowi istotne ograniczenie w realizacji takich rozwiązań. Gdy dostępny zbiór danych jest niewielki, trening sieci neuronowej zwykle kończy się niepowodzeniem. Jednym z rozwiązań problemu deficytu danych jest zastosowanie uczenia dwuetapowego. Pierwszy etap polega na wstępnym uczeniu sieci na innym zadaniu, gdzie dostępny jest znacznie większy zbiór danych. Podczas treningu, sieć nabywa zdolność ekstrakcji cech, które są zazwyczaj na tyle uniwersalne, że mogą być przydatne w innych zadaniach. Drugim etapem jest trening sieci na zadaniu docelowym. Technika polegająca na wykorzystaniu parametrów sieci otrzymanych w wyniku treningu na innym zadaniu nosi nazwę transfer learning. Najczęściej stosowaną praktyką jest stosowanie parametrów sieci neuronowych poddanych wstępnemu uczeniu na zadaniu klasyfikacji ImageNet. Skuteczność tego rozwiązania jest zależna od podobieństwa zbioru ImageNet do zbioru stosowanego

w zadaniu docelowym. Innym sposobem rozwiązania problemu deficytu danych oznaczonych jest stosowanie metod, w których wstępne uczenie odbywa się bez użycia oznaczeń. Takie podejście może okazać się szczególnie przydatne w obszarach, gdzie występują duże zasoby nieoznaczonych danych, a koszt oznaczania jest wysoki. W ostatnich latach pojawiło się wiele rozwiązań w tym obszarze, między innymi metody PIRL [22], BYOL [23], czy SimCLR [21]. Jednak podobnie jak w metodach doboru struktury dominującym podejściem jest adaptowanie i testowania rozwiązań na dużych zbiorach danych. Znacznie mniejsza uwaga skupiona jest na rozwoju zastosowań w obszarach, gdzie stosowane są niewielkie rozmiary danych.

Uwzględniając przedstawione wcześniej zagadnienia oraz problemy, sformułowano następującą tezę rozprawy:

Zastosowanie podejścia optymalizacyjnego do projektowania architektury sieci oraz uczenia dwuetapowego w warunkach deficytu danych prowadzi do uzyskania mniejszych struktur dopasowanych do danego problemu, przy jednoczesnym zachowaniu dokładności klasyfikacji

W celu potwierdzenia tej tezy określono następujące cele badawcze:

1. Przeprowadzenie analizy wpływu poszczególnych elementów struktury sieci neuronowej oraz technik uczenia na uzyskiwaną skuteczność,
2. Opracowanie skutecznej metody doboru struktury sieci neuronowej przy ograniczonej ilości danych,
3. Analiza metod uczenia dwuetapowego w warunkach deficytu danych.

Rozprawa została podzielona na siedem rozdziałów. W rozdziale drugim przedstawiono podstawowe informacje związane z uczeniem maszynowym, w szczególności z klasycznymi i głębokimi sieciami neuronowymi. Następnie, w rozdziale trzecim szerzej opisano techniki stosowane w głębokich sieciach neuronowych i w analizie obrazu. W rozdziale czwartym scharakteryzowano bazę danych stosowaną w badaniach oraz zdefiniowano główne miary skuteczności stosowane w ewaluacji rozpatrywanych rozwiązań. Opisano także eksperymenty mające na celu określić wpływ struktury sieci neuronowej, hiperparametrów oraz stosowanych technik na uzyskiwaną skuteczność. W rozdziale piątym przedstawiono badania prowadzone w kierunku automatycznego doboru struktury sieci neuronowej. W rozdziale szóstym opisano wpływ zaproponowanego podejścia wykorzystującego

uczenie dwuetapowe na osiągane wyniki. Opisano również wyniki połączenia metod doboru struktury z uczeniem dwuetapowym. Rozdział siódmy stanowi podsumowanie rozprawy.

Badania realizowane w trakcie doktoratu finansowane były ze środków budżetowych na naukę w latach 2017-2021, jako projekt badawczy „Opracowanie nowych struktur głębokich sieci neuronowych do analizy obrazów 2d, dla celów wspomagania decyzji" nr 0207/DIA/2017/46 w ramach programu Diamentowy Grant organizowanego przez Ministerstwo Edukacji i Nauki.

Wynikiem badań prowadzonych w ramach doktoratu jest zbiór publikacji naukowych:

1. A. Kwasigroch, M. Grochowski i A. Mikołajczyk, „Self-Supervised Learning to Increase the Performance of Skin Lesion Classification," *Electronics*, t. 9, nr. 11, s. 1930, 2020, ISSN: 2079-9292. DOI: 10.3390/electronics9111930

2. A. Kwasigroch, M. Grochowski i A. Mikołajczyk, „Neural Architecture Search for Skin Lesion Classification," *IEEE Access*, t. 8, s. 9061–9071, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2964424

3. A. Kwasigroch, M. Grochowski i M. Mikolajczyk, „Deep Neural Network Architecture Search Using Network Morphism," w *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2019, s. 30–35. DOI: 10.1109/MMAR.2019.8864624

4. M. Grochowski, A. Kwasigroch i A. Mikołajczyk, „Selected Technical Issues of Deep Neural Networks for Image Classification Purposes," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, t. 67, nr. No. 2, s. 363–376, 2019. DOI: 10.24425/bpas.2019.128485

2. Wprowadzenie do głębokich sieci neuronowych w analizie obrazu

2.1. Uczenie maszynowe

Uczenie maszynowe jest rodziną metod z zakresu sztucznej inteligencji, która stosowana jest w przypadkach, gdy trudno jest określić formalną procedurę realizacji danego problemu. Przykładem może być rozpoznawanie cyfr, które jest zadaniem stosunkowo prostym dla człowieka, natomiast skomplikowanym dla komputerów. System rozpoznawania cyfr oparty o instrukcje, reguły i heurystykę działał będzie tylko w ściśle określonych warunkach (np. tylko dla określonego oświetlenia czy tła), a jego dokładność nie będzie zbyt wysoka. Bardziej skutecznym podejściem jest zastosowanie algorytmu uczenia maszynowego, co nie wymaga określenia instrukcji realizacji zadania przez twórcę algorytmu. Algorytm samodzielnie nauczy się rozpoznawania cyfr poprzez analizę obrazów cyfr wraz z odpowiadającymi etykietami określającymi, jaka cyfra znajduje się na obrazie.

Zadanie w uczeniu maszynowym realizowane jest przez model, który może być reprezentowany przez pewną funkcję matematyczną oraz parametry, które wpływają na sposób działania modelu. Umiejętności i wiedza modelu zakodowane są w postaci struktury modelu oraz jego parametrów. Model może być wyrażony nawet w postaci prostej funkcji liniowej $y = ax + b$ opisanej tylko dwoma parametrami a i b. Umiejętność realizacji zadania nabywana jest w trakcie uczenia, podczas którego dostrajane są wartości parametrów modelu. Uczenie polega na optymalizacji zadanej funkcji celu, która odzwierciedla skuteczność realizacji zadania.

2.1.1 Model uczenia maszynowego

Jak zostało wcześniej wspomniane, model uczenia maszynowego może być reprezentowany przez funkcję:

$$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{\theta}) \tag{2.1}$$

gdzie:

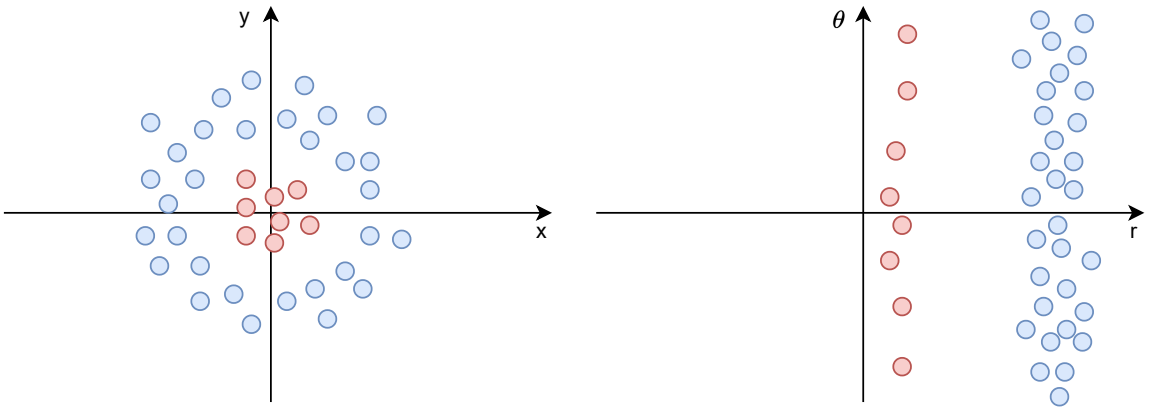
$\boldsymbol{x}$ – wejście modelu,
$\boldsymbol{\theta}$ – parametry modelu,
$\boldsymbol{y}$ – wyjście modelu,
f – funkcja realizowana przez model.

Przykład wejściowy może być przedstawiony w postaci wektora $x \in R^n$, który nazywany jest również reprezentacją przykładu. Reprezentacja składa się z pojedynczych liczb nazywanych cechami, które opisują właściwości danego przykładu uczącego. Reprezentacja może, ale nie musi mieć interpretacji fizycznej. Wektor opisujący wyniki pacjenta, wartość pikseli obrazu, czy kolejne próbki fali dźwiękowej to przykłady reprezentacji posiadających fizyczną interpretację. Reprezentacja bez interpretacji fizycznej najczęściej generowana jest przez inny algorytm uczenia maszynowego np. algorytm redukcji wymiarowości.

Przetwarzanie danych wejściowych w inną formę nazywane jest inżynierią cech. Celem inżynierii cech jest przekształcenie reprezentacji przykładów uczących w postać ułatwiającą uczenie oraz realizację zadania przez model. Ponadto zmiana reprezentacji może również zmniejszyć wymagania obliczeniowe poprzez zmniejszenie rozmiaru przykładów wejściowych. Przykładem inżynierii cech może być przekształcenie sygnału z dziedziny czasu na dziedzinę częstotliwości. W obszarze analizy obrazu opracowano wiele metod inżynierii cech: od prostych, takich jak zmiana przestrzeni kolorów, filtrowanie, zmiana obrazu na obraz czarno-biały, czy progowanie (ang. *thresholding*), po złożone algorytmy, takie jak SIFT [28], Haar [29] czy metoda HOG [30].

Znaczenie reprezentacji danych zostało przedstawione na rys. 2.1, na którym pokazano zbiór danych składający się z dwóch klas przedstawionych jako punkty czerwone i niebieskie. Zmiana reprezentacji danych ze współrzędnych kartezjańskich na biegunowe znacznie upraszcza zadanie klasyfikacji. W układzie biegunowym zadanie klasyfikacji może być realizowane przez prosty model:

$$f(r,\theta) = \begin{cases} 1, & \text{dla } r > \theta \\ 0, & \text{dla } r \leqslant \theta \end{cases} \tag{2.2}$$



Rysunek 2.1: Zmiana reprezentacji danych z układu kartezjańskiego (lewa strona) na układ biegunowy (prawa strona)

Kolejnym pojęciem związanym z uczeniem maszynowym jest wyjście modelu. Wyjście y generowane przez model może przybierać różną postać np. liczby całkowitej, liczby rzeczywistej, wektora lub macierzy. Przykładem wyjścia może być obraz oczyszczony z szumów, etykieta, czy współrzędne obiektów na obrazie.

Model stanowi kluczowy element systemów uczenia maszynowego, który może przyjmować różne formy od prostej funkcji liniowej, po skomplikowane funkcje opisane miliardami parametrów. Z postacią modelu związane jest zagadnienie przestrzeni hipotez, czyli zbioru funkcji, które mogą być reprezentowane przez model. Przykładowo, do przestrzeni hipotez modelu regresji liniowej należą wszystkie funkcje liniowe. Zmiana przestrzeni hipotez wpływa na pojemność modelu, czyli zdolność modelu do odwzorowania szerokiego zakresu relacji. Modele o niskiej pojemności, takie jak modele liniowe, nie są w stanie odwzorować złożonych relacji wymaganych do realizacji zadań takich jak klasyfikacja obrazów czy tłumaczenie tekstu.

Działanie modelu warunkowane jest przez zbiór parametrów θ, które określają relację pomiędzy wejściem a wyjściem modelu. Parametry podlegają dostrajaniu w trakcie uczenia, w celu osiągnięcia zamierzonego zachowania modelu. Modele uczenia maszynowego mogą zawierać od kilku do nawet kilku miliardów parametrów. Oprócz modeli zawierających parametry istnieje odrębna klasa modeli nieparametrycznych, jednak to zagadnienie nie jest poruszane w tej rozprawie.

2.1.2 Podział algorytmów uczenia maszynowego

Większość stosowanych algorytmów uczenia maszynowego można przypisać do jednej z dwóch kategorii: uczenia nadzorowanego (ang. *supervised learning*) lub uczenia nienadzorowanego (ang. *unsupervised learning*). Oprócz tych kategorii istnieją takie, które łączą cechy uczenia nadzorowanego i nienadzorowanego m.in. uczenie samonadzorowane (ang. *self-supervised learning*), uczenie częściowo nadzorowane (ang. *semi-supervised learning*) oraz uczenie słabo nadzorowane (ang. *weakly supervised learning*). Inną dziedziną uczenia maszynowego, znacznie różniącą się od wymienionych kategorii jest uczenie przez wzmocnienie (ang. *reinforcement learning*).

Uczenie w sposób nadzorowany jest obecnie jednym z najczęściej stosowanych podejść w uczeniu maszynowym. Uczenie polega na znalezieniu zależności f łączącej wejścia x z wyjściami y, przy użyciu zbioru par uczących $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$. Podczas uczenia, minimalizowana jest wybrana funkcja błędu, która wyraża zgodność pomiędzy wartościami oczekiwanymi y_i ze zbioru $\mathcal{D}$ a wartościami $\hat{y}_i$ generowanymi przez model dla wejścia x_i. Przykładowo, opracowanie systemu diagnostyki zmian patologicznych widocznych na obrazie tomografii komputerowej wiąże się z zastosowaniem zbioru danych, gdzie x_i jest obrazem otrzymanym w badaniu, a y_i stanowi wynik diagnozy. Duży zbiór danych oraz odpowiedni model umożliwia uzyskanie zadowalających wyników przy użyciu uczenia nadzorowanego. Jednak, zastosowanie uczenia nadzorowanego wymaga posiadania oczekiwanych odpowiedzi y, których uzyskanie jest zazwyczaj kosztowne i czasochłonne.

Kolejną dużą grupą metod uczenia maszynowego są metody uczenia nienadzorowanego. Algorytmy samodzielnie uczą się właściwości zbioru danych $\mathcal{D} = \{x_i\}_{i=1}^{N}$, odkrywają w nim wzorce, regularności i powiązania. Takie podejście określane jest również odkrywaniem wiedzy (ang. *knowledge discovery*). Zastosowanie uczenia nienadzorowanego nie wiąże się z koniecznością posiadania wartości oczekiwanych y_i, co odróżnia tę rodzinę metod od metod uczenia nadzorowanego. Grupowanie przykładów uczących (klasteryzacja), estymacja rozkładu prawdopodobieństwa, usuwanie szumu z danych czy redukcja wymiarowości to typowe zadania, w których stosuje się uczenie nienadzorowane.

Uczenie częściowo nadzorowane łączy w sobie cechy uczenia nadzorowanego oraz nienadzorowanego. Zastosowanie tej techniki umożliwia wykorzystanie danych oznaczonych $\mathcal{D}_{ozn} = \{(x_i, y_i)\}_{i=1}^{N}$ oraz nieoznaczonych $\mathcal{D}_{nozn} = \{x_i\}_{i=1}^{N}$. Uczenie częściowo nadzorowane znajduje zastosowanie w sytuacjach, gdy tylko część zbioru

jest oznaczona. Przykłady uczące bez oznaczeń mogą posłużyć uczeniu modelu oraz generowania odpowiednich reprezentacji, co powinno przełożyć się na większą skuteczność modelu. Uczenie częściowo nadzorowane może być realizowane na dwa sposoby: w sposób sekwencyjny lub równoległy. Sposób sekwencyjny polega na treningu modelu w sposób nienadzorowany, a następnie w sposób nadzorowany. Uczenie równoległe polega na równoczesnym wykorzystaniu podczas treningu przykładów oznaczonych oraz nieoznaczonych.

Uczenie samonadzorowane polega na uczeniu modelu zadania pomocniczego, gdzie etykiety dla zadania generowane są automatycznie z danych wejściowych. Celem stosowania uczenia samonadzorowanego jest nabycie przez model umiejętności generowania reprezentacji danych, co może zostać wykorzystane w zadaniu docelowym. Po treningu w sposób samonadzorowany, model poddaje się uczeniu w sposób nadzorowany na zadaniu docelowym. Opracowano wiele zadań dodatkowych, w których wykorzystuje się cechy zbioru danych do automatycznego generowania oznaczeń. Na przykład, obrazy mogą być poddane losowym modyfikacjom, a zadaniem sieci jest predykcja stopnia intensywności modyfikacji (np. kontrastu czy jasności).

Uczenie słabo nadzorowane jest szeroką rodziną metod, w których w trakcie uczenia stosowana jest niekompletna informacja. Do tej rodziny można zaliczyć algorytmy, uczące się na podstawie niepoprawnych lub niekompletnych oznaczeń. Przykładem zbioru danych z szumem w oznaczeniach jest zbiór obrazów Instagram-1B [31], gdzie rolę oznaczeń pełniły tagi (znacznik rozpoczęty znakiem #, np. #deep_learning) przypisywane zdjęciom przez użytkowników portalu Instagram. Innym przykładem uczenia słabo nadzorowanego jest uczenie na wielu instancjach (ang. *multiple instance learning*). Zadanie polega na wykryciu wystąpienie obiektu w zbiorze obrazów, gdzie obiekt niekoniecznie musi wstępować na wszystkich obrazach. Przykładem takiego problemu jest wykrywanie zmian w mózgu na badaniach tomografii komputerowej, która składa się z wielu przekrojów (około 100), a zmiana zazwyczaj występuje tylko na kilku przekrojach.

2.1.3 Zadania realizowane przez algorytmy uczenia maszynowego

Uczenie maszynowe znalazło liczne zastosowania w różnych dziedzinach. Typowymi zadaniami realizowanymi przez algorytmy uczenia maszynowego są m.in. klasyfikacja, regresja, klasteryzacja czy redukcja wymiarowości. Dodatkowo istnieją

zadania specyficzne dla danego obszaru takie jak segmentacja obrazu, tłumaczenie tekstu, synteza mowy czy generowanie struktur związków chemicznych.

Klasyfikacja należy do jednych z najczęściej realizowanych zadań przez modele uczenia maszynowego. Celem algorytmu klasyfikacji jest znalezienie odwzorowania wejść x na wyjścia y określające klasę, gdzie $y \in \{1, ..., C\}$, a C stanowi liczbą klas. Zbliżonym zadaniem do klasyfikacji jest regresja, z tą różnicą, że zamiast klasy zwracana jest wartość rzeczywista $y \in \mathbb{R}$ lub wartość ze zdefiniowanego zakresu. Modele realizujące zadania regresji oraz klasyfikacji najczęściej uczone są w sposób nadzorowany.

Klasteryzacja jest kolejnym przykładem zadania realizowanego przez algorytmy uczenia maszynowego. Klasteryzacja polega na przypisaniu grupy $y \in \{1, ..., K\}$ przykładowi x, gdzie K stanowi liczbę grup. Do uczenia stosowany jest zbiór danych bez oznaczeń $\mathcal{D}_{nozn} = \{x_i\}_{i=1}^{N}$. Metody klasteryzacji grupują podobne przykłady w K grupy, na podstawie zdefiniowanej miary podobieństwa. Klasteryzacja podobna jest do klasyfikacji, ponieważ każdemu przykładowi model przypisuje grupę ze zbioru y, do której przynależy. Grupy nie są jednak z góry zdefiniowane przez użytkownika algorytmu jak w klasyfikacji. W zależności od użytego algorytmu klasteryzacji podział na grupy może znacznie się różnić. Uczenie modelu klasteryzacji odbywa się w sposób nienadzorowany.

Inną klasą algorytmów uczenia maszynowego są algorytmy redukcji wymiarowości. Celem tych algorytmów jest znalezienie odwzorowania wejść $x \in \mathbb{R}^n$ na $y \in \mathbb{R}^m$, gdzie $m < n$. Pożądaną cechą algorytmów jest generowania wyjść y zawierających istotne informacje o przykładzie, przy jednoczesnym pomijaniu informacji nieistotnych, takich jak szum. Algorytmy często stosowane są na etapie przetwarzania danych na potrzeby innych algorytmów uczenia maszynowego. Zmniejszenie liczby wymiarów może przyczynić się do poprawy działania algorytmów poprzez przyspieszenie uczenia czy zwiększenie ich skuteczności. Redukcja wymiarowości jest również popularnym narzędziem stosowanym w wizualizacji, gdzie dane wielowymiarowe sprowadza się do dwóch lub trzech wymiarów (t-SNE [32], PCA). Redukcja wymiarowości stosowana jest również w metodach kompresji danych [33].

Detekcja anomalii jest techniką stosowaną do identyfikacji przypadków, znacząco odbiegających od reszty. Jednym z powszechnie stosowanych podejść jest modelowanie rozkładu danych za pomocą funkcji gęstości prawdopodobieństwa $p(x)$ i identyfikacja obserwacji o niskim prawdopodobieństwie. Innym podejściem jest

identyfikacja nietypowych obserwacji na podstawie analizy odległości przykładu od innych przykładów (podobnie jak w algorytmach klasteryzacji). Algorytmy detekcji anomalii znajdują szerokie zastosowanie w wielu obszarach, stosowane są w systemach bankowych do wykrywania nieautoryzowanego użycia kart kredytowych, identyfikacji nietypowego ruchu w sieciach komputerowych, czy filtrowania spamu w systemach pocztowych. Wykrywanie anomalii ma również istotne zastosowania przemysłowe, takie jak monitorowanie poprawności działania urządzeń i procesów.

2.1.4 Proces uczenia

Aby uzyskać wysoką skuteczność realizacji zadania, należy dobrać parametry modelu w trakcie uczenia. Uczenie odbywa się poprzez dostrajanie parametrów modelu w celu optymalizacji wybranej miary reprezentującej skuteczność realizacji zadania. Stosowana miara jest specyficzna dla rodzaju wykonywanego zadania, np. zadania klasyfikacji wymagają innych miar niż zadania regresji. Ponadto miara stosowana jako funkcja celu w procesie uczenia powinna mieć odpowiednie właściwości. Jedną z takich właściwości jest często różniczkowalność, która umożliwia stosowanie gradientowych metod optymalizacji.

Przed rozpoczęciem treningu należy dobrać szereg ustawień modelu, przetwarzania danych i treningu, które nazywane są hiperparametrami. Przykładami hiperparametrów mogą być współczynnik uczenia, liczba i rodzaj warstw sieci neuronowej, czy rozmiar paczki danych (ang. *batch size*). Hiperparametry dobierane są zazwyczaj ręcznie, a ich dobór może być czasochłonny, gdyż po każdej zmianie hiperparametrów trening musi być przeprowadzony od nowa. Efektywny dobór hiperparametrów wymaga zarówno doświadczenia, jak i intuicji.

2.1.5 Przykład regresji liniowej

Regresja liniowa stanowi jeden z najprostszych algorytmów uczenia maszynowego. Wejściem modelu jest wektor x, na podstawie którego, model generuje wyjściową wartość rzeczywistą y. Przykładowo, algorytm można zastosować do oszacowania ceny mieszkania y na podstawie cech mieszkania x. Wektor x będzie zawierał wartości opisujące parametry mieszkania, takie jak liczba pokoi, powierzchnia, czy liczba sypialni, natomiast wartość y będzie stanowiła cenę mieszkania. Zadaniem modelu jest wygenerowanie ceny mieszkania na podstawie dostarczonych cech mieszkania.

Model regresji liniowej opisany jest równaniem:

$$\hat{y} = \boldsymbol{\theta}^\top \boldsymbol{x} \tag{2.3}$$

W trakcie uczenia modelu dobierane są wartości parametrów $\boldsymbol{\theta}$, aby minimalizować założoną miarę. W zadaniach regresji najczęściej stosowaną miarą jest błąd średniokwadratowy (ang. *Mean Sqare Error* – MSE), który wyraża zdolność modelu do generowania odpowiedzi jak najbardziej zbliżonych odpowiedziom oczekiwanym. Jeśli odpowiedzi generowane przez model i odpowiedzi oczekiwane ze zbioru danych są takie same, to błąd średniokwadratowy wynosi zero. Miara definiowana jest równaniem:

$$\mathrm{MSE} = \frac{1}{M} \sum_{i=1}^{M} (y_i - \hat{y}_i)^2 \tag{2.4}$$

gdzie:

M – liczba przykładów,
$\hat{y}_i$ – wyjście modelu,
y_i – oczekiwana odpowiedź modelu.

Model wyrażony w postaci funkcji liniowej oraz błąd średniokwadratowy są różniczkowalne, zatem możliwe jest zastosowanie gradientowej metody optymalizacji w celu minimalizacji błędu. Metoda regresji liniowej jest na tyle prosta, że rozwiązanie zadania optymalizacji można otrzymać w sposób analityczny, zgodnie z poniższym wzorem:

$$\boldsymbol{\theta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{2.5}$$

gdzie:

$\boldsymbol{\theta}$ – parametry modelu,
$\boldsymbol{X}$ – macierz przykładów uczących,
$\boldsymbol{y}$ – wektor oczekiwanych odpowiedzi.

W praktyce rozwiązanie analityczne stosuje się tylko w przypadku optymalizacji modeli o niskim stopniu złożoności np. regresji liniowej czy SVM.

W bardziej złożonych modelach, w których nie jest możliwe analityczne wyznaczenie rozwiązania, stosowane są zazwyczaj iteracyjne, numeryczne metody

optymalizacji. Przykładem takiej metody jest algorytm stochastycznego spadku wzdłuż gradientu (ang. *Stochastic Gradient Descent* – SGD), który znajduje zastosowanie zarówno w uczeniu prostych, jak i złożonych modeli. Algorytm SGD jest algorytmem iteracyjnym, który aktualizuje wartości parametrów θ podczas każdej iteracji, zgodnie z poniższym równaniem:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \eta \nabla_{\theta_n} \mathcal{L}(\boldsymbol{\theta}_n) \tag{2.6}$$

gdzie:

$\boldsymbol{\theta}_{n+1},\ \boldsymbol{\theta}_n$ – wektory parametrów w iteracjach $n+1$ oraz n,
η – współczynnik uczenia,
$\nabla \mathcal{L}$ – gradient funkcji celu.

Algorytm SGD w tej formie stosowany jest również w uczeniu głębokich sieci neuronowych. Metody optymalizacji zostaną opisane szerzej w rozdziale (3).

2.1.6 Generalizacja i metody regularyzacji

Pożądaną właściwością modeli uczenia maszynowego jest zdolność generowania poprawnych odpowiedzi dla przykładów, które nie były użyte w trakcie uczenia. Zdolność tę nazywa się generalizacją lub uogólnianiem. Wysoka skuteczność mierzona na danych wykorzystanych podczas treningu, nie gwarantuje, że model będzie generował poprawne odpowiedzi dla nowych danych.

Z uogólnianiem związane są pojęcia nadmiernego i zbyt słabego dopasowania modelu. Zbyt słabe dopasowanie objawia się poprzez niską skuteczność modelu mierzoną na zbiorze treningowym. Wynika to zazwyczaj z zastosowania modelu o zbyt małej pojemności w stosunku do zadania i dostępnego zbioru danych. Model o zbyt małej pojemności nie jest w stanie odwzorować złożonych funkcji niezbędnych do prawidłowej realizacji zadania.

Nadmierne dopasowanie modelu objawia się wysoką skutecznością mierzoną na zbiorze treningowym, ale niską na zbiorze testowym. Zdolność generalizacji (uogólniania) modelu jest w tym przypadku ograniczona. Modele o dużej pojemności są podatne na nadmierne dopasowanie ze względu na możliwość dokładnego dopasowania modelu do zbioru uczącego. Jest to działanie niepożądane, ponieważ model zazwyczaj uczy się uwzględniania nieistotnych informacji w procesie decyzyjnym, takich jak szum pomiarowy. Pojemność modelu powinna być

dostosowana do trudności zadania oraz do charakterystyki dostępnego zbioru danych. Odpowiedni dobór pojemności modelu pozwala uniknąć problemu nadmiernego i niedostatecznego dopasowania modelu.

W praktyce, nadmierne dopasowanie jest problemem dużo trudniejszym do rozwiązania. Aby łagodzić ten problem powstały metody regularyzacji, czyli modyfikacje wprowadzane do algorytmu uczenia, które mają na celu zmniejszenie błędu generalizacji modelu [34]. Do często stosowanych technik regularyzacji należą: zmiana pojemności modelu, regularyzacja L1 i L2, dropout [17], wczesne zatrzymanie treningu (ang. *early stopping*) czy metody przetwarzania danych wzbogacające zbiór uczący (ang. *data augmentation*). Część z tych metod zostanie szerzej opisana w rozdziale 3.

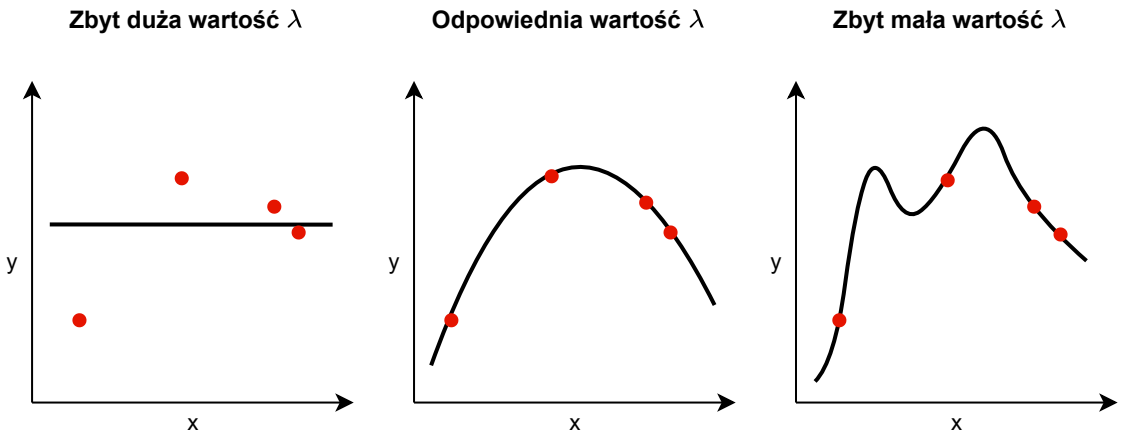
Przykładem szeroko stosowanej metody regularyzacji jest ograniczenie normy L_2 wektora parametrów modelu. Zastosowanie tej techniki ogranicza wzrost wartości parametrów modelu, utrudniając dopasowanie modelu do danych. Metoda ta znajduje zastosowanie zarówno w małych modelach, takich jak regresja liniowa, jak i w złożonych modelach z miliardami parametrów. Regularyzacja L_2 jest realizowana poprzez modyfikację funkcji celu, a jej intensywność jest zależna od dobieralnego współczynnika λ.

$$\mathcal{L}(\boldsymbol{\theta}) = \mathrm{MSE} + \lambda \boldsymbol{\theta}^\top \boldsymbol{\theta} \tag{2.7}$$

Wpływ wartości λ na dopasowanie modelu do danych treningowych przedstawiono na rys. 2.2

2.2. Sieci neuronowe

Sieci neuronowe są modelami uczenia maszynowego, które składają się z neuronów, realizujących proste obliczenia matematyczne. Neurony przesyłają sygnały między sobą, a ich organizacja połączeń przypomina sieć, stąd nazwa. Chociaż pojedyncze neurony realizują tylko proste operacje, to organizacja wielu neuronów w sieć umożliwia rozwiązywanie złożonych problemów. W kolejnych fragmentach zostanie opisany sposób działania jednokierunkowych sieci neuronowych, złożonych z warstw w pełni połączonych.



Rysunek 2.2: Wpływ współczynnika regularyzacji λ na dopasowanie modelu do zbioru uczącego

Działanie neuronu można opisać za pomocą równania:

$$y = f\left(\sum_{i=0}^{N} w_i x_i + b\right) \tag{2.8}$$

gdzie:

- y – wyjście neuronu,
- $\boldsymbol{x}$ – wejście neuronu,
- N – liczba wejść,
- f – funkcja aktywacji,
- $\boldsymbol{w}$ – wektor wag,
- b – próg

Wektor wag $\boldsymbol{w}$ oraz próg b są parametrami neuronu, które określają sposób jego działania. Wartości parametrów dobierane są w trakcie uczenia, tak aby sieć była w stanie realizować założone zadanie. Równanie neuronu można również przedstawić w postaci wektorowej:

$$y = f(\boldsymbol{w}^\top \boldsymbol{x}) \tag{2.9}$$

gdzie $\boldsymbol{w}$ jest wektorem wag (rozszerzonym o wartość b), natomiast $\boldsymbol{x}$ wektorem wejść (rozszerzonym o wartość jeden odpowiadającą progowi b).

Oprócz parametrów wpływ na działanie neuronu ma funkcja aktywacji, która dobierana jest na etapie projektowania struktury sieci neuronowej (jest to jeden z hipeparametrów). W praktyce stosuje się funkcje nieliniowe ze względu na

możliwość zwiększenia pojemności modelu i realizacji bardziej złożonych zadań niż w przypadku funkcji liniowych. Do najczęściej stosowanych funkcji aktywacji należą:

- funkcja liniowa $f(x) = x$,
- funkcja sigmoidalna $f(x) = \frac{e^x}{e^x+1}$,
- funkcja tangensoidalna $f(x) = \frac{e^x-e^{-x}}{e^x+e^{-x}}$,
- funkcja ReLU $f(x) = \max(0, x)$.

Typowym sposobem organizacji neuronów jest organizacja warstwowa. Wejściem każdego neuronu w warstwie są wyjścia neuronów z warstwy poprzedniej. Każdy z neuronów w warstwie realizuje inną funkcję, a ich parametry mogą zostać zapisane jako kolejne kolumny macierzy $\boldsymbol{W}$. Równanie opisujące funkcję realizowaną przez warstwę można zatem zapisać w następujący sposób:

$$\boldsymbol{y} = f(\boldsymbol{W}\boldsymbol{x}) \tag{2.10}$$

Warstwy sieci neuronowej zorganizowane są w taki sposób, że wyjście warstwy stanowi wejście warstwy kolejnej. Równanie wielowarstwowej sieci neuronowej można zatem przedstawić w postaci funkcji zagnieżdżonej. Na przykład, równanie sieci neuronowej złożonej z trzech warstw można zapisać następująco:

$$\boldsymbol{y} = f_3(f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2)\boldsymbol{\theta}_3) \tag{2.11}$$

gdzie:

$\boldsymbol{x}$ – wejście sieci neuronowej,
$\boldsymbol{y}$ – wyjście sieci neuronowej,
f_1, f_2, f_3 – funkcje aktywacji realizowane przez poszczególne warstwy,
$\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3$ – parametry warstw.

Dobór struktury sieci neuronowej jest złożonym problemem. Wybór liczby warstw, liczby neuronów w warstwie czy funkcji aktywacji, ma istotny wpływ na skuteczność osiąganą przez sieć neuronową. Bardziej złożona struktura daje możliwość realizacji złożonych funkcji matematycznych, jednak zwiększa podatność modelu na nadmierne dopasowanie i utrudnia uczenie. Dodatkowo bardziej złożony model zwiększa wymagania obliczeniowe, co prowadzi do dłuższego uczenia oraz

wydłużenia czasu generowania odpowiedzi. Dobór struktury przeprowadzany jest ręcznie lub w sposób automatyczny przy pomocy algorytmów doboru struktury (ang. *Neural Architecture Search*).

Uczenie sieci neuronowej przebiega w podobny sposób do uczenia mniej złożonych modeli np. opisanej wcześniej regresji liniowej. Zazwyczaj stosowane funkcje celu oraz struktury sieci neuronowych są różniczkowalne, dzięki czemu możliwe jest zastosowanie gradientowych metod optymalizacji. Gradient względem parametrów wyznaczany jest metodą wstecznej propagacji błędu, która opiera się na łańcuchowej metodzie wyznaczania pochodnych. Otrzymany gradient wykorzystywany jest w gradientowych metodach optymalizacji. Najczęściej stosowaną metodą uczenia sieci neuronowych jest metoda SGD (ang. *Stochastic Gradient Descent*).

2.3. Głębokie sieci neuronowe

Jednokierunkowe, w pełni połączone sieci neuronowe znalazły wiele zastosowań. Poprawne działanie takich sieci wymagało zastosowania wstępnego przetwarzania danych i ekstrakcji cech. Zastosowanie nieprzetworzonych danych w uczeniu (np. obrazów w formacie RGB) prowadziło do słabych wyników. W obszarach przetwarzania obrazu czy mowy opracowano liczne algorytmy, transformujące obraz lub nagranie w wektor cech.

Podejmowano próby zwiększenia skuteczności działania sieci poprzez zwiększenie liczby przykładów uczących oraz zwiększenie rozmiaru sieci neuronowej. Zauważono, że zwiększenie rozmiaru zbioru uczącego przynosiło korzyści tylko do pewnego poziomu, po którego osiągnięciu dalsze powiększanie zbioru nie przynosiło wzrostu skuteczności. Natomiast, zwiększanie rozmiaru sieci prowadziło do pogorszenia wyników, a nawet uniemożliwiało uczenie. Jedną z przyczyn był problem zanikającego gradientu, który uniemożliwiał przeprowadzenie aktualizacji parametrów sieci neuronowej.

Zastosowanie głębokich sieci neuronowych, złożonych z wielu warstw stało się możliwe dzięki wprowadzeniu szeregu rozwiązań. Wprowadzenie funkcji aktywacji ReLU, zmniejszyła problem zanikających gradientów, przyspieszyło trening i uodporniło sieć na niewłaściwą inicjalizację parametrów. Obecnie funkcja ReLU lub jej warianty stosowane są w większości struktur głębokich sieci neuronowych. W obszarze analizy obrazu istotny wpływ miało wprowadzenie warstw

konwolucyjnych, w których zredukowano liczbę połączeń, zmniejszono liczbę parametrów, oraz wprowadzono współdzielenie parametrów.

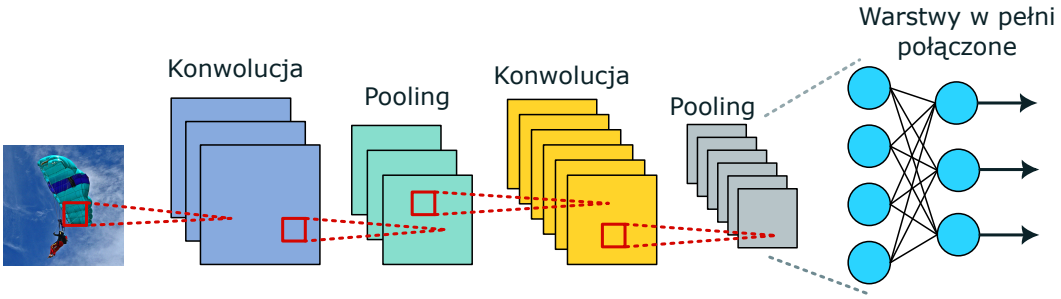
Wzrost skuteczności głębokich sieci neuronowych był możliwi dzięki rozwojowi w wielu różnych kierunkach. Między innymi nowe funkcje aktywacji (ReLU, Swish, GeLU), metody optymalizacji (Adam, AdamW, RMSProp), warstwy normalizujące (batch normalization, layer normalization, group normalization), rozwiązania architektoniczne (moduły Inception, połączenia skracające (ang. *shortcut connection*), warianty warstw konwolucyjnych (separable, depth), techniki transfer learning i wiele innych prowadzą do coraz lepszych wyników. Oprócz tego wzrost wydajności jednostek obliczeniowych oraz większe zbiory danych pozwalają na zwiększanie skuteczności realizowanych rozwiązań. Niektóre z metod stosowanych w rozwiązaniach opartych o głębokie sieci neuronowe zostaną opisane w kolejnym rozdziale.

Szerokim obszarem zastosowań uczenia maszynowego jest analiza obrazu. Z uwagi na duży rozmiar obrazów, osiągnięcie wysokiej skuteczności stanowi wyzwanie. Zastosowanie nieprzetworzonych obrazów w uczeniu prostych algorytmów takich jak płytkie sieci neuronowe czy SVM skutkuje niską skutecznością. Dopiero zastosowanie inżynierii cech i uzyskanie reprezentacji o mniejszej liczbie cech, zawierającej istotne informacje umożliwiało poprawę wyników. Metody ekstrakcji cech stanowią istotny element systemów przetwarzania obrazu.

Odmianą głębokich sieci neuronowych stosowaną w analizie obrazu są konwolucyjne sieci neuronowe. Konwolucyjne sieci neuronowe dokonują hierarchicznej ekstrakcji cech poprzez wielokrotne filtrowanie obrazu. Dzięki temu, dane wejściowe nie muszą być poddane złożonej ekstrakcji cech, jak w przypadku stosowania prostych algorytmów. Schemat prostej sieci konwolucyjnej przedstawiony jest na rys. 2.3. Sieć składa się z warstw konwolucyjnych, po których stosowane są nieliniowe funkcje aktywacji (nieprzedstawione na schemacie). Redukcja wymiarów reprezentacji obrazu (map cech) dokonywana jest przez warstwy grupujące. Warstwy konwolucyjne wraz z warstwami grupującymi działają jako ekstraktor cech, którego parametry dobierane są w procesie uczenia.

Realizowane zadanie determinuje rodzaj wyjścia i rodzaj ostatniej warstwy sieci neuronowej. Końcowe warstwy realizują zadanie na podstawie cech otrzymanych z poprzedzających warstw konwolucyjnych. Na przykład, w zadaniu segmentacji wyjściem sieci jest macierz o rozmiarze odpowiadającym obrazowi wejściowemu,

natomiast w zadaniach klasyfikacji wyjściem jest wektor o rozmiarze odpowiadającym liczbie rozpatrywanych klas.



Rysunek 2.3: Schemat konwolucyjnej sieci neuronowej. Dla zwiększenia czytelności, na schemacie nie umieszczono warstw aktywacji

Warstwy konwolucyjne znajdują zastosowanie w zadaniach, w których istotne są lokalne relacje przestrzenne w analizowanych przykładach wejściowych. Przykładem tego typu danych jest obraz, gdzie istotne są zarówno wartości pikseli (reprezentujące kolor), jak i ich wzajemne położenie. Na przykład, aby wykryć tekst na obrazie, należy wykryć krawędzie, co wymaga analizy intensywności sąsiadujących pikseli. Najpopularniejszymi wariantami warstw konwolucyjnych są klasyczna warstwa konwolucyjna, warstwa konwolucyjna z separacją głębokości (ang. *depth separable convolution*) oraz warstwa konwolucyjna z separacją przestrzeną (ang. *spatially separable convolution*).

Warstwy konwolucyjne są znacznie skuteczniejsze w analizie obrazu niż warstwy w pełni połączone. Połączenia neuronów w warstwach w pełni połączonych zorganizowane są na zasadzie „każdy z każdym". Na przykład, jeśli wejściem warstwy zawierającej 50 neuronów, jest stosunkowo niewielki obraz o rozmiarze $100 \times 100 \times 3$ pikseli to liczba połączeń w tej warstwie wynosi aż 1,5 mln ($100 \cdot 100 \cdot 3 \cdot 50$). Każde połączenie odpowiada jednemu parametrowi, co oznacza, że liczba parametrów opisującą tę warstwę również wynosi 1,5 mln.

Warstwa konwolucyjna korzysta z odmiennego sposobu organizacji połączeń pomiędzy sygnałem wejściowym a neuronami w porównaniu z warstwami w pełni połączonymi. Każdy neuron w warstwie konwolucyjnej połączony jest tylko z kilkoma sąsiadującymi neuronami z poprzedniej warstwy. Neurony w warstwie konwolucyjnej analizują tylko małą części obrazu, co pozwala na specjalizację w wykrywaniu określonych cech w obrazie. Jest to uzasadnione, ponieważ wykrycie cech, takich jak krawędzie wymaga analizy niewielkiej części obrazu, a nie całego obrazu. W praktyce

każdy neuron w warstwie konwolucyjnej połączony jest z niewielkim obszarem obrazu, najczęściej o rozmiarze 3×3, 5×5 lub 7×7. W porównaniu z neuronem w warstwie w pełni połączonej, w której neuron połączony jest ze wszystkimi pikselami obrazu wejściowego, sposób połączeń w warstwie konwolucyjnej charakteryzuje się znaczną redukcją liczby połączeń i parametrów. Przykładowo, dla 50 neuronów i wejścia o rozmiarze $100 \times 100 \times 3$ liczba parametrów w warstwie w pełni połączonej wynosi 1,5 mln, podczas gdy dla warstwy konwolucyjnej z 50 filtrami o rozmiarze 5×5 liczba ta wynosi zaledwie 3750 ($50 \cdot 5 \cdot 5 \cdot 3$). Dodatkowo właściwości warstw konwolucyjnych umożliwiają realizację obliczeń równoległych na jednostkach graficznych, co stanowi ich istotną zaletę.

Oprócz lokalności połączeń kolejną cechą neuronów w warstwie konwolucyjnej jest współdzielenie parametrów. Niektóre cechy, mogą występować w różnym miejscach na obrazie, dlatego lokalnie połączone neurony współdzielą ze sobą parametry. Dzięki temu możliwe jest wykrywanie określonych cech w różnych miejscach na obrazie, co zapewnia odporność sieci na zmianę położenia obiektów. Obraz generowany przez warstwę konwolucyjną nazywany jest mapą cech. Współdzielenie parametrów, podobnie jak lokalność połączeń, prowadzi do ograniczenia liczby parametrów i ilości obliczeń.

Kolejnym typem warstwy często stosowanej w sieciach konwolucyjnych w analizie obrazu jest warstwa grupująca (ang. *pooling*), której zadaniem jest redukcja rozmiaru map cech. Jedną z częściej stosowanych warstw grupujących jest warstwa MaxPooling. Działanie warstwy polega na podziale obrazu w siatkę, na niewielkie, nienakładające się kwadratowe obszary (zazwyczaj zawierające od kilku do kilkunastu pikseli), a następnie wybraniu z każdego obszaru największej wartości. Uzyskane wartości tworzą nowy, mniejszy obraz. Na przykład, zastosowanie obszarów o rozmiarze 2×2 piksele prowadzi do czterokrotnego zmniejszenia liczby pikseli w obrazie.

Operacja konwolucji jest operacją liniową, zatem w celu zwiększenia możliwości reprezentowania złożonych funkcji przez sieć neuronową po każdej warstwie konwolucyjnej stosowana jest nieliniowa warstwa aktywacji. Funkcją aktywacji, która znalazła najszersze zastosowanie w sieciach konwolucyjnych jest funkcja Rectified Linear Unit (ReLU), opisana wzorem:

$$y = \max(0, x) \tag{2.12}$$

Funkcja ReLU ma cechy ułatwiające trening sieci neuronowych. Dla argumentów $x \in (0, \infty)$ funkcja zachowuje się jak funkcja liniowa. Gradient w tym zakresie jest stały, co ułatwia jego przepływ i poprawia proces uczenia. Natomiast dla argumentów $x \in (-\infty, 0)$, zarówno funkcja, jak i gradient przyjmują wartość zero. Taka postać funkcji łagodzi problem zanikającego gradientu, który jest powszechny w wielowarstwowych sieciach, w których stosuje się funkcje nasycające się takie jak funkcja sigmoidalna.

3. Rozwiązania stosowane w konwolucyjnych sieciach neuronowych

Dynamiczny rozwój głębokiego uczenia skutkował powstaniem licznych rozwiązań, takich jak: warstwy, architektury, metody uczenia, metody przetwarzania danych i wiele innych. Niniejszy rozdział przedstawia metody, które zostały wykorzystane w ramach doktoratu, ale również inne powszechnie stosowane techniki.

Opis rozpoczyna się od przedstawiania warstw, które są podstawowym elementem budującym struktury głębokich sieci neuronowych stosowanych w analizie obrazu. Przedstawione zostaną warstwy konwolucyjne, funkcje aktywacji, metody inicjalizacji wag, warstwy grupujące oraz metody normalizacji. Następnie, opisane zostaną popularne architektury stosowane w zadaniach przetwarzania obrazu.

Kolejnym tematem omawianym w rozdziale będą metody uczenia m.in. optymalizatory oraz harmonogramy współczynnika uczenia. Następnie opisane zostaną inne techniki często stosowane w rozwiązaniach uczenia maszynowego, takie jak rozszerzanie danych, transfer learning czy grupowanie modeli.

Dalsza część rozdziału poświęcona zostanie metrykom stosowanym w ocenie działania modelu. Przedstawione zostaną różne podejścia do oceny skuteczności modelu, w szczególności metody ilościowe takie jak dokładność, czułość, swoistość czy wartość ROC AUC. Na końcu opisane zostaną procedury testowania modeli, które stosowane są w celu uzyskania wiarygodnych wyników.

3.1. Warstwy stosowane w konwolucyjnych sieciach neuronowych

Warstwy konwolucyjne

Warstwy konwolucyjne stanowią podstawowy element architektur stosowanych w przetwarzaniu obrazu. Warstwy te stosuje się w przetwarzaniu danych o strukturze siatki, jaką stanowi obraz, który składa się z uporządkowanych pikseli. Główną rolą warstw konwolucyjnych jest nauka reprezentacji, która posłuży realizacji zadania docelowego np. klasyfikacji czy segmentacji. W kolejnych sekcjach omówione zostaną szczegóły działania popularnych wariantów warstw konwolucyjnych.

Klasyczna warstwa konwolucyjna

Klasyczna warstwa konwolucyjna realizuje operację splotu, znaną również pod nazwą konwolucji. Operacja splotu dyskretnego opisana jest następującym wzorem:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \tag{3.1}$$

gdzie:

S – wyjście
K – filtr (jądro)
I – wejście

Pomimo że, w literaturze stosowana jest nazwa „warstwa konwolucyjna", to w implementacjach bibliotek programistycznych zamiast operacji splotu zazwyczaj realizowana jest operacja korelacji wzajemnej (ang. *cross-corelation*). Podobnie w materiałach dydaktycznych, zazwyczaj przedstawiana jest operacja korelacji wzajemnej, ponieważ forma tej operacji ułatwia objaśnienie operacji filtrowania, które może być wyrażone w postaci iloczynu Hadamarda. Operacja korelacji wzajemnej opisana jest wzorem:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{3.2}$$

Operację korelacji wzajemnej od operacji konwolucji różni jedynie odwrócenie filtru, co jest nieistotne w przypadku algorytmów uczenia maszynowego, ponieważ w wyniku optymalizacji parametrów model jest w stanie dostosować się do wybranej operacji.

Wejściem klasycznej warstwy konwolucyjnej, stosowanej w zadaniach przetwarzania obrazu, jest wielowymiarowy tensor o wymiarze $h \times w \times c$, gdzie h i w odpowiadają wysokości i szerokości wejścia, a c reprezentuje liczbę kanałów (map cech). Liczba map cech wejścia determinuje rozmiar pojedynczego filtru w warstwie, który wynosi $k_h \times k_w \times k_c$, gdzie: k_h i k_w reprezentują wysokość i szerokość filtru, a k_c liczbę kanałów filtru, która odpowiada liczbie kanałów wejścia c. Wartości parametrów filtru dobierane są w trakcie uczenia w procesie optymalizacji. Pojedynczy filtr konwolucyjny generuje jedną mapę cech na wyjściu. Zastosowanie wielu filtrów w warstwie umożliwia generację wyjścia składającego się z wielu map cech, co umożliwia wykrywanie różnorodnych cech.

Pojedyncza mapa cech generowana jest w następujący sposób. Każdemu kanałowi w obrazie wejściowym odpowiada kanał w filtrze. Filtr przesuwa się po obrazie wejściowym o stałą wartość kroku i dokonuje operacji iloczynu Hadamarda, zwanego również iloczynem skalarnym. Iloczyn Hadamarda dwóch macierzy opisany jest wzorem:

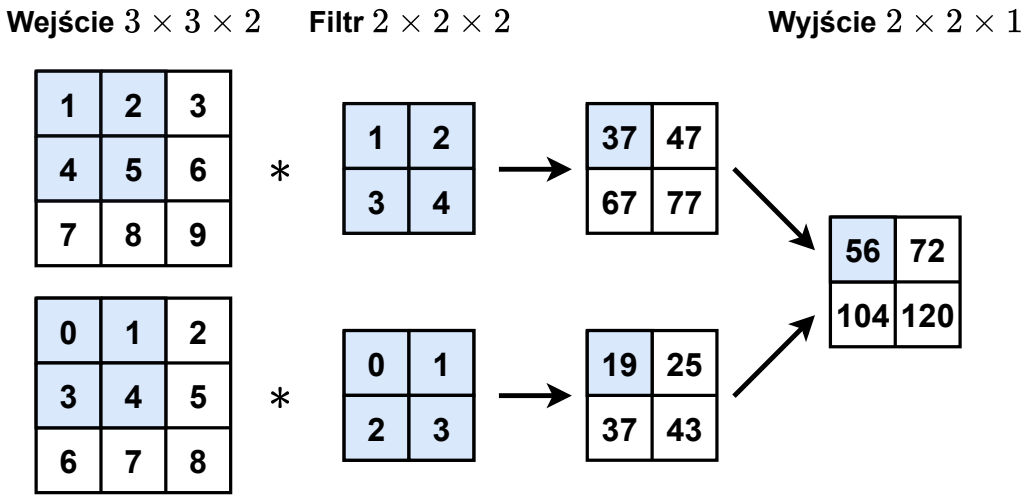
$$(\boldsymbol{A} \odot \boldsymbol{B})_{i,j} = a_{i,j} b_{i,j} \tag{3.3}$$

Na przykład:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \odot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix} \tag{3.4}$$

Następnie wygenerowane mapy cech odpowiadające każdemu kanałowi wejściowemu są sumowane, czego wynikiem jest pojedyncza mapa cech. Sposób działania pojedynczego filtru konwolucyjnego zaprezentowany jest na rys. 3.1. Warstwy konwolucyjne składają się zazwyczaj z wielu filtrów, więc filtrowanie powtarzane jest wielokrotnie, generując wiele map cech.

Wyjście generowane przez warstwy konwolucyjne ma mniejszą wysokość i szerokość niż wejściowy obraz, co jest przedstawione na rys. 3.1. Rozmiar wyjścia wynosi $(h - k_h + 1) \times (w - k_w + 1)$. Aby uzyskać mapę cech o takiej samej wysokości i szerokości co obraz wejściowy, do obrazu wejściowego dodaje się dodatkowe kolumny i wiersze, zazwyczaj wypełniane zerami. Rozmiar wyjściowej mapy cech przy zastosowaniu tej modyfikacji wynosi $(h - k_h + p_h + 1) \times (w - k_w + p_w + 1)$, gdzie p_h oraz p_w to liczba



Rysunek 3.1: Przykład działania filtru konwolucyjnego

dodanych wierszy i kolumn na krawędziach mapy cech. Wartości p_h, p_w oraz sposób wypełnienia dodatkowych kolumn i wierszy są dobieralnymi hiperparametrami.

Kolejnym dobieralnym hiperparametrem, który wpływa na działanie warstwy konwolucyjnej, jest kroku filtru (ang. *stride*). Zwiększenie kroku filtru powoduje zmniejszenie wysokości i szerokości wyjściowej mapy cech. Na przykład krok o wartości dwa zmniejsza wysokość i szerokość wyjściowej mapy cech prawie o połowę. Wyjściowy rozmiar mapy cech zależny od kroku można wyznaczyć za pomocą wzoru: $(h - k_h + p_h + s_h)/s_h \times (w - k_w + p_w + s_w)/s_w$, gdzie s_h oraz s_w reprezentują krok filtru w pionie i poziomie.

Warstwa przestrzennie separowalnej konwolucji

Warstwa przestrzennie separowalnej konwolucji (ang. *spatial separable convolution*), wykazuje się efektywnością, dlatego jest jest stosowana w rozwiązaniach o ograniczonych zasobach obliczeniowych m.in. w aplikacjach mobilnych. Warstwa opisana jest mniejszą liczbą parametrów niż klasyczna warstwa konwolucyjna, a jej działanie wymaga mniejszej ilości obliczeń. Działanie warstwy opiera się na właściwości niektórych macierzy kwadratowych, które można przedstawić jako iloczyn dwóch wektorów. Dzięki temu filtr $k \times k$ można przedstawić jako iloczyn dwóch filtrów $k \times 1$ i $1 \times k$ (rów. 3.5).

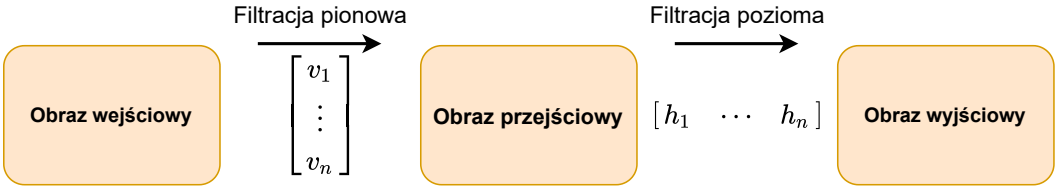
$$\boldsymbol{A} = \boldsymbol{v}^\top \boldsymbol{h} \tag{3.5}$$

Na przykład:

$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \tag{3.6}$$

Reprezentacja filtru w postaci iloczynu dwóch wektorów pozwala na zmniejszenie liczby parametrów i operacji potrzebnych do przetworzenia wejścia. Na przykład, dekompozycja przedstawiona w równaniu 3.6 pozwala na zmniejszenie liczby realizowanych operacji i liczby parametrów z dziewięciu do sześciu.

W praktyce, działanie tej warstwy polega na dwukrotnym przefiltrowaniu sygnału wejściowego: najpierw z użyciem filtru pionowego, a następnie poziomego (rys. 3.2). Ograniczeniem tej warstwy jest fakt, że nie każdy filtr można przedstawić w postaci iloczynu dwóch wektorów. Ogranicza to możliwość odwzorowania dużego zbioru funkcji przez sieć neuronową.



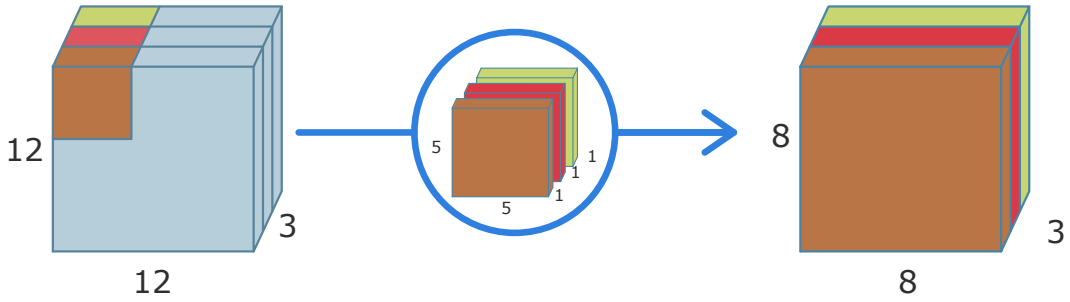
Rysunek 3.2: Działanie warstwy spatial separable convolution

Warstwa depth separable convolution

Warstwa depth separable convolution podobnie jak warstwa spatial separable convolution stosowana jest w celu zmniejszenia liczby parametrów i realizowanych operacji. Warstwa nie jest jednak ograniczona postacią filtru konwolucyjnego, dlatego też jest częściej stosowana. Działanie warstwy składa się z dwóch etapów: konwolucji typu depthwise oraz konwolucji punktowej.

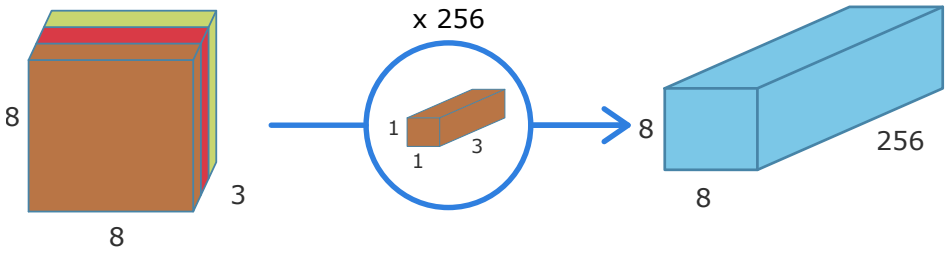
Konwolucja typu depthwise polega na przeprowadzeniu filtrowania każdego kanału wejściowego osobno. Każdemu kanałowi wejściowemu odpowiada filtr o rozmiarze $k_h \times k_w \times 1$. Wynikiem działania tej operacji jest wyjście o liczbie kanałów równej liczbie kanałów wejścia. Na przykład, jeśli wejście ma rozmiar $224 \times 224 \times 32$, należy zastosować

warstwę zawierająca 32 filtry o rozmiarze $w \times h \times 1$, gdzie w i h to szerokość i wysokość filtru. Wynikiem działania warstwy będzie wyjście o rozmiarze $224 \times 224 \times 32$. W tym kroku nie występuje wymiana informacji między kanałami.



Rysunek 3.3: Pierwszy etap realizowany przez warstwę depth separable convolution – konwolucja depthwise

Drugim krokiem jest operacja konwolucji punktowej (eng. *pointwise convolution*). Ten etap służy do wygenerowania większej liczby kanałów obrazu wyjściowego i pozwala uwzględnić relację między kanałami wejściowymi. Konwolucja punktowa polega na zastosowaniu filtru o rozmiarze $1 \times 1 \times c_{in}$, gdzie c_{in} odpowiada liczbie kanałów obrazu wejściowego. Wybór liczby filtrów determinuje liczbę kanałów obrazu wyjściowego. Schemat działania tego kroku przedstawiony jest na 3.4.



Rysunek 3.4: Drugi etap realizowany przez warstwę depth separable convolution – konwolucja punktowa

Funkcje aktywacji

Funkcja aktywacji zazwyczaj stosowana jest po warstwach konwolucyjnych lub po warstwach w pełni połączonych. W klasycznych sieciach neuronowych złożonych z kilku warstw dominującym zastosowaniem była funkcja sigmoidalna lub tangensoidalna, które są funkcjami ograniczonymi z dołu oraz z góry. Ich zastosowanie wymaga odpowiedniej inicjalizacji parametrów, w taki sposób, aby wyjścia nie znalazły się w obszarze nasycenia, co spowalnia, a nawet może zatrzymać

proces uczenia. Obecnie, w głębokich sieciach neuronowych domyślnym wyborem jest funkcja ReLU i jej warianty.

Funkcja ReLU opisana jest wzorem:

$$y = \begin{cases} x, & \text{dla } x > 0 \\ 0, & \text{dla } x \leqslant 0 \end{cases} \tag{3.7}$$

Funkcja nie jest ograniczona z góry, więc nie nasyca się w obszarze dla wejść $x > 0$. Właściwość ta przeciwdziała problemowi zanikającego gradientu oraz przyczynia się do usprawnienia procesu uczenia, dzięki czemu z powodzeniem stosowana jest w głębokich sieciach neuronowych. Autorzy [2] zaobserwowali, że czas uczenia sieci, w której zastosowano funkcję ReLU, zmniejsza się czterokrotnie w stosunku do sieci, w których zastosowano funkcję tangensoidalną. Brak ograniczenia funkcji z góry jest tak istotną cechą, że prawie wszystkie funkcje aktywacji proponowane po pojawieniu się funkcji ReLU mają tę własność.

Pomimo usprawnienia treningu i zmniejszenia problemu zanikającego gradientu, funkcja ReLU nadal jest podatna na niewłaściwą inicjalizację parametrów, co może prowadzić do pojawienia się tzw. martwych neuronów. Martwy neuron to neuron, którego sygnał wyjściowy zawsze wynosi zero, niezależnie od przykładu wejściowego sieci neuronowej. Konsekwencją tego jest gradient równy zero, co uniemożliwia aktualizację parametrów neuronu.

Problem martwych neuronów złagodzono w funkcji Leaky ReLU, będącą modyfikacją funkcji ReLU [35]. Funkcja Leaky ReLU dla argumentów $x \leqslant 0$ przybiera postać funkcji liniowej o niewielkim nachyleniu, które jest określane przez hiperparametr α. Dzięki temu funkcja posiada niezerowy gradient dla całej dziedziny $x \in (-\infty, \infty)$.

$$y = \begin{cases} x, & \text{dla } x > 0 \\ \alpha x, & \text{dla } x \leqslant 0 \end{cases} \tag{3.8}$$

Funkcję Leaky ReLU poddano dodatkowej zmianie, której wynikiem była funkcja Parametric ReLU [36]. Postać funkcji jest taka sama jak funkcji Leaky ReLU, różni się jednak tym, że wartość współczynnika α jest parametrem dobieralnym w trakcie uczenia, a nie ręcznie dobieranym hiperparametrem.

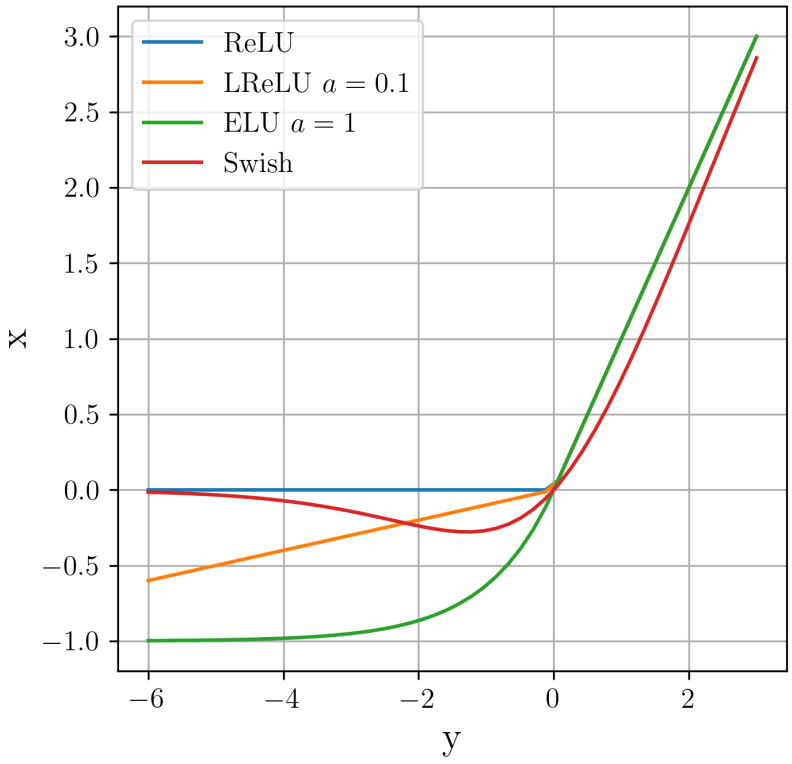
Kolejną popularną funkcją aktywacji jest funkcja ELU (Exponential Linear Unit), którą od funkcji ReLU odróżnia jej postać dla argumentów $x \leqslant 0$. W tym obszarze funkcja przybiera postać krzywej logarytmicznej. Funkcja opisana jest wzorem:

$$y = \begin{cases} x, & \text{dla } x > 0 \\ a(e^x - 1), & \text{dla } x \leqslant 0 \end{cases} \tag{3.9}$$

Funkcja Swish posiada szereg właściwości usprawniających uczenie: ograniczenie z góry, ograniczenie z dołu, niemonotoniczność i regularność. Niemonotoniczność oraz regularność jest cechą wyróżniająca funkcję Swish od prezentowanych wcześniej funkcji. Funkcja opisana jest wzorem:

$$y = x \cdot \text{sigmoid}(x) = \frac{x}{1 + e^{-x}} \tag{3.10}$$

Wykresy opisanych funkcji aktywacji znajdują się na rys. 3.5.



Rysunek 3.5: Popularne funkcje aktywacji

Metody inicjalizacji parametrów

Przed rozpoczęciem treningu sieci neuronowej należy wybrać początkowe wartości parametrów. Parametry można dobrać w sposób losowy lub zastosować parametry innej sieci poddanej uczeniu na innym zadaniu. W tej sekcji zostaną opisane losowe metody inicjalizacji parametrów. Poprawna inicjalizacja jest istotnym czynnikiem, ponieważ niewłaściwy dobór początkowych wartości parametrów może prowadzić do nasycenia funkcji aktywacji, pojawienia się martwych neuronów, a nawet prowadzić do utknięcia rozwiązania w lokalnym minimum, co negatywnie wpływa na finalną efektywność modelu.

W sieciach neuronowych, zawierających kilka warstw parametry losowano z rozkładu normalnego lub jednostajnego. Parametry rozkładów dobierane były heurystycznie, uwzględniając hiperparametry sieci neuronowej. W przedstawionym równaniu, parametry rozkładu parametrów warstwy zależą od liczby neuronów w poprzedniej warstwie:

$$W_{ij} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \tag{3.11}$$

gdzie $U[-a, a]$ jest rozkładem jednostajnym w przedziale $(-a, a)$, a n jest liczbą neuronów w poprzedniej warstwie. Taki sposób inicjalizacji sprawdzał się w sieciach składających się z niewielu warstw z sigmoidalną lub tangensoidalną funkcją aktywacji. W sieciach z większą liczbą warstw ten sposób inicjalizacji prowadził do nasycania się neuronów.

Autorzy [37] zaproponowali sposób inicjalizacji warstw uwzględniający multiplikatywny efekt wynikający ze stosowania wielu warstw. Efekt polega na tym, że niewielka zmiana wartości wejścia lub parametru sieci neuronowej może skutkować dużą zmianą wyjścia. Zaproponowane parametry rozkładów prawdopodobieństwa, które prowadziły do zmniejszenia problemu nasycania się neuronów. Sposób inicjalizacji został nazwany *Xavier* od imienia autora publikacji.

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right] \tag{3.12}$$

Inicjalizacja *Xavier* sprawdzała się dla funkcji ograniczonych z dołu oraz z góry, nie sprawdzała się jednak w przypadku funkcji ReLU.

Autorzy [36] zaproponowali sposób inicjalizacji warstw dla wielowarstwowych sieci neuronowych z funkcją aktywacji ReLU:

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_l}}\right) \tag{3.13}$$

gdzie $\mathcal{N}(\mu, \sigma)$ to rozkład normalny o średniej μ i odchyleniu standardowym σ. Jest to domyślnie stosowany sposób inicjalizacji parametrów w popularnych bibliotekach m.in. Pytorch czy Tensorflow.

Warstwy grupujące

Warstwy grupujące zmniejszają rozmiar map cech w głębokiej sieci neuronowej, co przyspiesza trening oraz ma właściwości regularyzacyjne. Najczęściej stosowane warstwy grupujące to warstwa MaxPooling oraz AveragePooling. Ich działanie polega na podziale wejściowej mapy cech na kwadratowe, nienakładające się na siebie obszary, a następnie agregacji wartości z obszarów. W przypadku warstwy MaxPooling wybierana jest wartość maksymalna, a w przypadku warstwy AveragePooling – wartość średnia. Otrzymane wartości tworzą zredukowaną mapę cech.

Szczególnym rodzajem warstw grupujących są warstwy typu global, np. Global AveragePooling czy Global MaxPooling. W odróżnieniu od warstw MaxPooling czy AveragePooling mapa cech nie jest dzielona na obszary, a operacja średniej czy maksimum dokonywana jest na wszystkich wartościach z mapy cech. Zazwyczaj te warstwy stosowane są na wyjściu ostatniej warstwy konwolucyjnej w strukturze, gdzie rozmiar map cech jest niewielki. Wejście warstwy global może być dowolnego rozmiaru, co umożliwia stosowanie wejścia sieci neuronowej o różnym rozmiarze.

Warstwy normalizujące

W głębokich sieciach neuronowych złożonych z wielu warstw obserwuje się tzw. efekt multiplikatywny. Jednym ze sposobów radzenia sobie z negatywnymi skutkami tego efektu są opisane wcześniej metody inicjalizacji parametrów. Oprócz tego często stosowaną praktyką jest wybranie niskiego współczynnika uczenia oraz przeprowadzenie normalizacji danych wejściowych. Innym rozwiązaniem, jest dodanie do architektury sieci neuronowej warstwy normalizujące. Idea tych warstw opiera się na fakcie, że normalizacja wejść modelu usprawnia proces uczenia. Stąd można wnioskować, że normalizacja wejść każdej warstwy może skutkować lepszym

treningiem. Powstało szereg metod normalizujących warstwy, wśród których dominującymi rozwiązaniami są: metoda Batch Normalization oraz metoda Layer Normalization, które zostaną przedstawione w kolejnych sekcjach.

Metoda Batch Normalization

Metoda Batch Normalization [18] jest najczęściej stosowaną warstwą normalizacji. Metoda polega na sprowadzeniu średniej wartości aktywacji warstwy do zera oraz odchylenia standardowego do jeden. Średnią oraz odchylenie standardowe wyznacza się po pojawieniu się paczki danych (ang. *batch*), stąd nazwa. Działanie warstwa opisane jest następującym równaniem:

$$\hat{x} = \frac{x - \mathbb{E}[x]}{\sqrt{\mathrm{Var}[x]}} \tag{3.14}$$

$$y = \gamma \hat{x} + \beta \tag{3.15}$$

gdzie:

x – wejście warstwy

y – wyjście warstwy

W warstwie wprowadza się dodatkowe parametry γ oraz α, których wartości dobierane są w trakcie uczenia. Parametry umożliwiają modyfikację wartości wyjścia warstwy po przeprowadzeniu normalizacji. Możliwe jest również odtworzenie wartości aktywacji przed normalizacją poprzez odpowiednie dobranie współczynników γ oraz α. Warstwa Batch Normalization jest obecnie standardowym rozwiązaniem, stosowanym w wielu sieciach takich jak ResNet, VGG, DenseNet czy Wide ResNet.

Metoda Layer Normalization

Warstwa Batch Normalization wymaga stosowania dużych paczek danych zawierających wiele przykładów uczących, aby poprawnie obliczyć średnią i odchylenie standardowe, co stanowi jej ograniczenie. Warstwa Layer Normalization [38] rozwiązuje ten problem poprzez wyznaczanie średniej oraz odchylenia standardowego na podstawie jednego przykładu uczącego. Średnia oraz odchylenie standardowe wyznaczane są na podstawie wartości aktywacji wszystkich neuronów

danej warstwy. To podejście pozwala na działanie warstwy niezależnie od rozmiaru paczki danych, co umożliwia trening nawet z paczką zawierająca jeden przykład uczący.

Równania opisujące warstwę Layer Normalization są identyczne jak w warstwie Batch Normalization. Różni się jednak wartość, na podstawie których wyznaczane są średnia i odchylenie standardowe. Podobnie jak w metodzie Batch Normalization, stosowane są dodatkowe parametry γ oraz β, które dobierane są w trakcie uczenia.

$$\hat{x} = \frac{x - \mathbb{E}[x]}{\sqrt{\mathrm{Var}[x]}} \tag{3.16}$$

$$y = \gamma\hat{x} + \beta \tag{3.17}$$

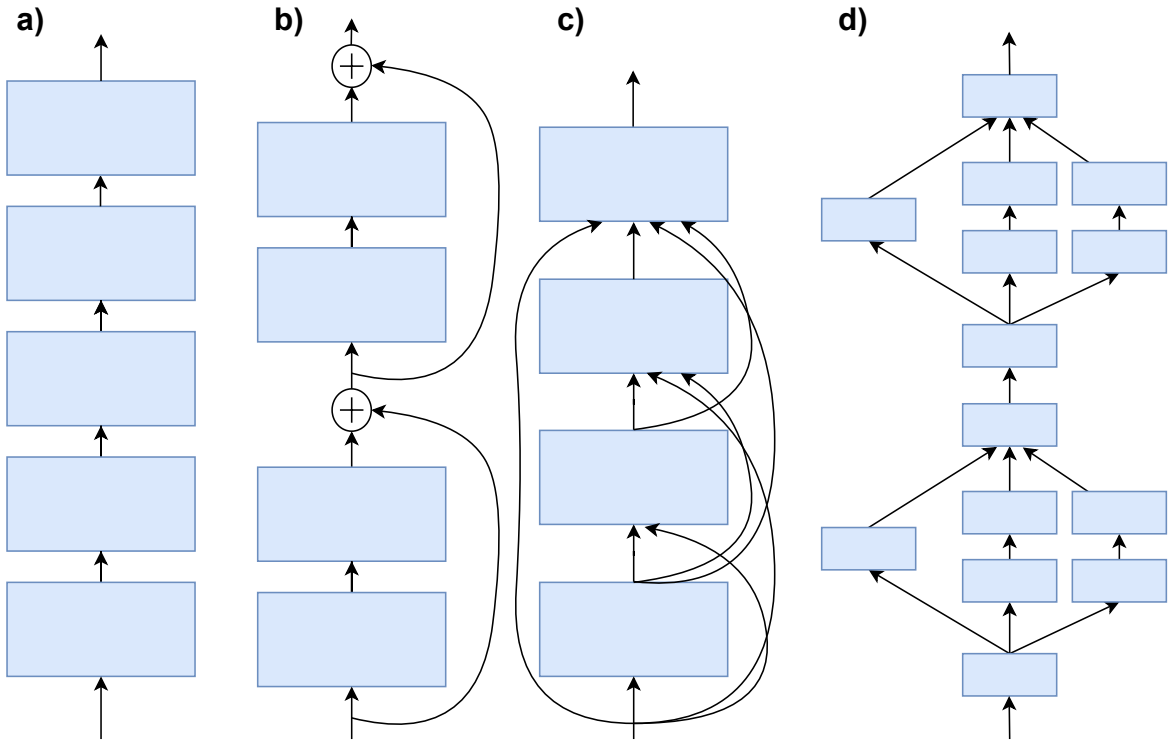
3.2. Popularne architektury głębokich sieci

Struktura głębokiej sieci neuronowej jest kluczowym elementem wpływającym m.in. na skuteczność, zdolność generalizacji, wymagania obliczeniowe, łatwość uczenia oraz zapotrzebowanie na dane. Rozwój architektur sieci neuronowych miał znaczący wpływ na rozwój głębokiego uczenia. Wiele z opracowanych struktur takich jak ResNet, VGG czy EfficientNet stały się standardowymi rozwiązaniami w zadaniach analizy obrazu. Struktura modelu bezpośrednio wpływa na jego pojemność, która powinna być dostosowana do trudności zadania oraz ilości danych. Projektowanie architektur głębokich sieci neuronowych jest jednym z głównych kierunków badań w ramach głębokiego uczenia.

Sieci neuronowe o dużej głębokości charakteryzują się różnorodnymi wzorcami połączeń, które obejmują zarówno proste jednokierunkowe sieci, jak i skomplikowane, wielogałęziowe struktury. Na rysunku 3.6 przedstawiono przykłady typowych wzorców połączeń warstw w sieciach neuronowych wraz z odpowiadającymi architekturami.

Sieć AlexNet

Opracowanie struktury AlexNet [2] uważane jest za przełomowy moment w rozwoju głębokiego uczenia. Sieć została opracowana w celu udziału w konkursie ImageNet Large Scale Visual Recognition 2012, gdzie zadaniem było stworzenie klasyfikatora obrazów o wysokiej dokładności.



Rysunek 3.6: Popularne schematy połączeń w sieciach neuronowych. a) Sposób połączeń stosowany w sieci VGG b) Schemat połączeń stosowany w sieci ResNet c) Schemat połączeń stosowany w sieci DenseNet d) Schemat połączeń stosowany w sieci GoogleLeNet

Wejście sieć AlexNet stanowi obraz o rozmiarze $224 \times 224 \times 3$. Sieć zawiera pięć warstw konwolucyjnych oraz trzy warstwy w pełni połączone. Z powodu ograniczonego rozmiaru pamięci kart graficznych użytych w eksperymentach, zastosowano dwie gałęzie, przechowywane osobno w pamięci kart. Filtry w warstwach konwolucyjnych mają rozmiary: $11 \times 11 \times c$, $5 \times 5 \times c$ oraz $3 \times 3 \times c$, gdzie c oznacza liczbę kanałów wejściowych. W celu redukcji rozmiaru map cech zastosowano krok filtru większy niż jeden. Oprócz tego, pomiędzy niektórymi warstwami wprowadzono warstwę MaxPooling. Ponadto, w strukturze wprowadzono warstwy normalizujące oraz warstwę dropout. Sieć AlexNet zakończona jest trzema warstwami w pełni połączonymi składającymi się z 4096 neuronów każda oraz warstwą Softmax o rozmiarze 1000 elementów, odpowiadającym liczbie klas.

Architektura AlexNet odegrała znaczącą rolę w rozwoju architektur sieci neuronowych. Sieć umożliwiła uzyskanie znacznie wyższej dokładności klasyfikacji niż wcześniejsze rozwiązania oparte na klasycznej ekstrakcji cech i prostych klasyfikatorach. Wprowadzone innowacje, takie jak funkcja aktywacji ReLU, normalizacja warstw, czy metoda dropout stały się standardem w projektowaniu głębokich sieci neuronowych.

Sieć VGG

Kolejną rodziną struktur, która odegrała ważną rolę w rozwoju głębokiego uczenia jest rodzina sieci VGG [14]. Struktury, przedstawione w publikacji, składają się z różnej liczby warstw – od 11 do 19, w momencie opracowania tej struktury taka liczba warstw uchodziła za bardzo wysoką. Sieci zawierały od 133 do 144 mln parametrów, z czego większą część parametrów stanowiły parametry warstw w pełni połączonych umiejscowionych na końcu sieci. Wejściem sieci jest obraz RGB o rozmiarze $224 \times 224 \times 3$. Struktury wyposażone są w warstwy konwolucyjne o niewielkim rozmiarze filtru – $3 \times 3 \times c$ lub $1 \times 1 \times c$, oraz o kroku równym jeden.

Autorzy architektury VGG zauważyli, że kilka kolejnych warstw o rozmiarze filtru $3 \times 3 \times c$, może imitować działanie jednej warstwy o większym rozmiarze filtru. Na przykład dwie warstwy $3 \times 3 \times c$ odpowiadają warstwie o rozmiarze $5 \times 5 \times c$, a trzy warstwy $3 \times 3 \times c$ odpowiadają jednej warstwie $7 \times 7 \times c$. Stosowanie kilku mniejszych filtrów zamiast jednego o większym rozmiarze zmniejsza liczbę parametrów. Ponadto, jeśli pomiędzy warstwami występują nielinowe warstwy aktywacji, to kilka takich warstw ma lepsze możliwości odwzorowywania złożonych funkcji niż jedna warstwa z dużym filtrem.

W sieciach VGG zastosowano warstwy MaxPooling umiejscowione po niektórych warstwach konwolucyjnych. Sieci zakończone są dwiema warstwami w pełni połączonymi o rozmiarze 4096 i warstwą Softmax o rozmiarze 1000, który odpowiada liczbie klas. W sieciach stosowane są dwie warstwy dropout umiejscowione po warstwach w pełni połączonych. Tabela 3.1 zawiera szczegóły architektoniczne rodziny VGG.

Sieć ResNet

Po pojawieniu się sieci VGG podjęto próby dalszego rozwoju tych struktur. Próbowano zwiększyć liczbę warstw, aby uzyskać dalszy wzrost dokładności, jednak eksperymenty wykazywały pogorszenie wyników spowodowane m.in. problem zanikającego gradientu. W badaniach wykazano, że spadek dokładności nie był spowodowany nadmiernym dopasowaniem [39], [40].

W rodzinie struktur ResNet zaproponowano rozwiązania, które niwelowały ograniczenia sieci VGG [13]. Rodzina obejmowała sieci z jeszcze większą liczbą warstw — największa z nich składała się aż ze 150 warstw. Rozwiązaniem, które umożliwiło skuteczne uczenie tak dużych sieci było wprowadzenie połączeń skrających (ang. *shortcute connection*) widocznych na rys. 3.7, które pozwalają na przekazanie informacji z jednej warstwy do dowolnie odległej warstwy w sieci, pomijając warstwy pośrednie. Rozwiązanie to niweluje problem zanikającego gradientu, umożliwiając tworzenie i uczenie bardzo głębokich sieci neuronowych. Dodanie połączeń skracających do architektury nie prowadzi do zwiększenia liczby parametrów, a wynikający z tego wzrost ilość obliczeń jest pomijalnie niski.

Blok budujący sieci ResNet opisany jest równaniem:

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}) + \boldsymbol{x} \tag{3.18}$$

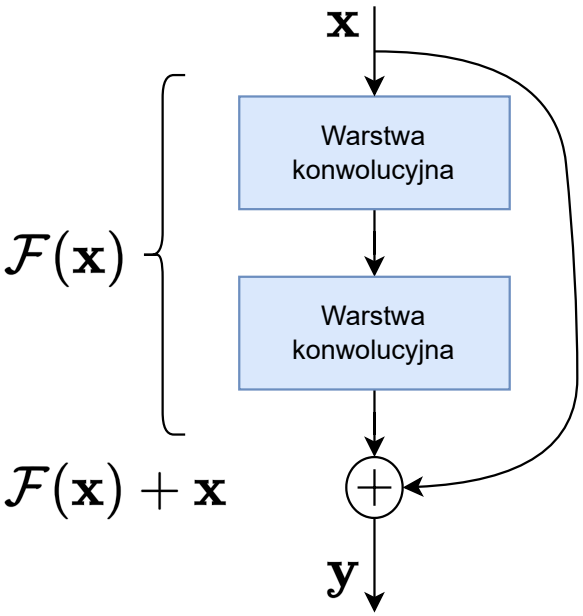
gdzie $\boldsymbol{y}$ to wyjście bloku, $\boldsymbol{x}$ wejście bloku, $\mathcal{F}$ funkcja realizowana przez zbiór warstw.

Jeśli rozmiary wejścia $\boldsymbol{x}$ i wyjścia bloku warstw $\mathcal{F}(x)$ są różne, niemożliwe jest dodanie do siebie tych wartości. Jednym ze sposobów wyrównania rozmiarów jest uzupełnienie map cech zerami. Jednakże, w eksperymentach przeprowadzonych przez autorów wykazano, że bardziej efektywnym rozwiązaniem jest zastosowanie projekcji liniowej, która realizowana jest poprzez wprowadzenie dodatkowej macierzy $\boldsymbol{W}_s$:

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}) + \boldsymbol{W}_s\boldsymbol{x} \tag{3.19}$$

Tabela 3.1: Rodzina sieci VGG

A	**B**	**C**	**D**	**E**
11 warstw	13 warstw	16 warstw	16 warstw	19 warstw
Wejście (obraz RGB 224 x 224)				
conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
Maxpooling				
conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
Maxpooling				
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
Maxpooling				
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
Maxpooling				
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
Maxpooling				
FC-4096				
FC-4096				
FC-1000				
Softmax				



Rysunek 3.7: Blok sieci ResNet

Autorzy zaproponowali sieci różnej liczbie warstw: 18, 34, 50, 101 i 152. W architekturach składających się z 50, 101 i 152 warstw zastosowano bloki typy bottleneck, składające się z trzech następujących po sobie warstw o następujących rozmiarach filtru: 1×1, 3×3, 1×1. Rolą pierwszej warstwy o rozmiarze 1×1 jest redukcja liczby kanałów, które trafią na warstwę o filtrze 3×3. Ostatnia warstwa o rozmiarze 1×1 odpowiada za przywrócenie wejściowej liczby kanałów. Bloki bottleneck służą ograniczeniu wymagań obliczeniowych sieci.

Zbiór warstw kownolucyjnych sieci ResNet zakończony jest warstwą Global AveragePooling, co umożliwia stosowania wejścia o dowolnym rozmiarze. W praktyce jednak zbyt duży rozmiar obrazów wejściowych utrudnia uczenie oraz powoduje gorsze wyniki. W tabeli 3.2 przedstawiono architekturę sieci ResNet18 oraz ResNet50.

Najczęściej stosowaną siecią rodziny ResNet jest sieć ResNet50. Oprócz tego powstały liczne warianty sieci ResNet, takie jak SeResNet [41], ResNeXt czy SeResNeXt [42].

Sieć Inception

Sieci VGG oraz ResNet miały liniową strukturę, natomiast sieć Inception [12] składa się z modułów, które zawierają działające równolegle warstwy o różnych rozmiarach

Tabela 3.2: Rodzina sieci ResNet

Nazwa warstwy	**Rozmiar wyjścia**	**ResNet-18**	**ResNet-50**
conv1	112×112	7×7, 64, krok 2	
conv_2x	56×56	3×3 maxpooling, krok 2	
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv_3×	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv_4x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv_5x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pooling, 1000-d fc, softmax	

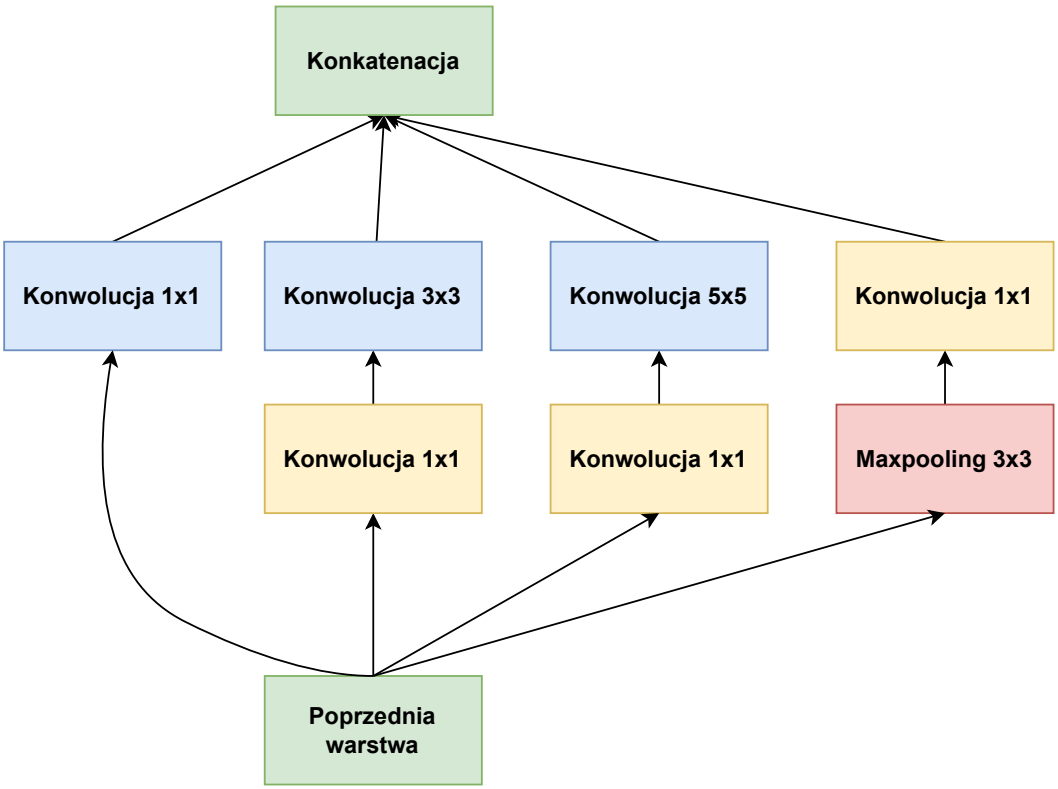
filtru. Moduł Inception przedstawiony na rys. 3.8 składa się z warstw konwolucyjnych z filtrami o rozmiarze 1×1, 3×3, oraz 5×5. Moduły w sieci uporządkowane są jeden po drugim w sposób analogiczny do liniowego uporządkowania warstw.

Zastosowanie równoległych filtrów o różnych rozmiarach, działających w ramach modułu, umożliwia analizę cech różnej skali, co zwiększa różnorodność cech generowanych przez moduł. Podobnie jak w bloku bottleneck sieci ResNet, przed warstwami zawierającymi filtry o rozmiarach 3×3 oraz 5×5 stosowane są warstwy 1×1 w celu redukcji liczby kanałów i zmniejszenia wymagań obliczeniowych. Wyjścia równoległych gałęzi w module agregowane są metodą konkatenacji.

Powstało wiele wariantów sieci neuronowych opartych o sieć Inception m.in. Inception-v2 oraz Inception-v3 [43], Inception-ResNet i Inception-v4 [44].

DenseNet

Cechą charakterystyczną sieci DenseNet [45] jest duża liczba połączeń między warstwami. Wejściem każdej warstwy konwolucyjnej w strukturze są wszystkie wyjścia warstw poprzedzających. Wyjścia poprzednich warstw, przed podaniem na wejścia kolejnych warstw, są agregowane metodą konkatenacji. Taki sposób



Rysunek 3.8: Blok Inception

organizacji warstw wprowadza $\frac{L(L+1)}{2}$ połączeń zamiast L jak w głębokich sieciach neuronowych o liniowej strukturze. Architektura sieci DenseNet przedstawiona jest w tabeli 3.3.

Opisany sposób organizacji połączeń minimalizuje problem zanikającego gradientu, poprawia propagację cech oraz umożliwia wykorzystanie cech w kolejnych warstwach. Prowadzi to do poprawy dokładności, zmniejszenia liczby parametrów oraz przyspieszenia treningu.

Tabela 3.3: Rodzina sieci DenseNet

Warstwa/blok	**Rozmiar wyjścia**	**DenseNet-121**	**DenseNet-169**
Konwolucyjna	112×112	7×7, krok 2	
Maxpooling	56×56	3×3, krok 2	
Blok Dense	56×56	$\begin{bmatrix} 1\times 1 \\ 3\times 3 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times 1 \\ 3\times 3 \end{bmatrix} \times 6$
Blok Transition	56×56	Konwolucja 1×1, krok 1	
	28×28	Average pooling 2×2, krok 2	
Blok Dense	28×28	$\begin{bmatrix} 1\times 1 \\ 3\times 3 \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times 1 \\ 3\times 3 \end{bmatrix} \times 12$
Blok Transition	28×28	Konwolucja 1×1, krok 1	
	14×14	Average pooling 2×2, krok 2	
Blok Dense	14×14	$\begin{bmatrix} 1\times 1 \\ 3\times 3 \end{bmatrix} \times 24$	$\begin{bmatrix} 1\times 1 \\ 3\times 3 \end{bmatrix} \times 32$
Blok Transition	14×14	Konwolucja 1×1, krok 1	
	7×7	Average pooling 2×2, krok 2	
Blok Dense	7×7	$\begin{bmatrix} 1\times 1 \\ 3\times 3 \end{bmatrix} \times 16$	$\begin{bmatrix} 1\times 1 \\ 3\times 3 \end{bmatrix} \times 32$
Klasyfikacja	1×1	Global average pooling 7×7	
		Softmax	

Sieć EfficientNet

Sieci z rodziny EfficientNet [46] podobnie jak sieć Inception, składają się z modułów, które zawierają warstwy przetwarzające sygnał w sposób równoległy. Struktura modułów nie została jednak dobrana ręcznie, a w wyniku działania

algorytmu doboru struktury, który miał na celu maksymalizację dokładności, przy minimalizacji wymagań obliczeniowych. W ramach rodziny EfficientNet utworzono osiem wariantów sieci, oznaczonych symbolem EfficientNet-B0...7 złożonych z modułów wygenerowanych przez algorytm. Warianty różnią się rozmiarem wejścia, liczbą warstw oraz ich szerokością. Częstą praktyką w zastosowaniach sieci neuronowych jest skalowanie tylko jednego hiperparametru np. liczby warstw. W rodzinie EfficientNet zastosowano równoległe skalowanie rozmiaru wejścia, liczby warstw oraz szerokości warstw.

Następcą rodziny sieci EfficientNet jest rodzina EfficientNetv2, w której wprowadzono rozwiązania prowadzące do większej efektywności i szybszego uczenia. Podobnie jak w pierwszej wersji zastosowano algorytm doboru struktury w celu znalezienia optymalnej struktury modułu. Jednak w tym przypadku, optymalizowano nie tylko dokładność i wymagania obliczeniowe, ale również liczbę parametrów. Ponadto zastosowano niejednorodne skalowanie głębokości, szerokości i rozmiaru wejścia oraz ograniczono możliwość nadmiernego wzrostu rozmiaru wejścia.

3.3. Trening głębokich sieci neuronowych

Sieć neuronowa nabywa zdolność realizacji zadania w trakcie uczenia, które polega na dostrajaniu parametrów w celu optymalizacji określonej funkcji celu, będącą miarą skuteczności działania modelu. Proces uczenia sieci neuronowej przebiega w sposób iteracyjny i polega na prezentowaniu kolejnych przykładów uczących, generowaniu odpowiedzi przez model i modyfikacji parametrów modelu, w ten sposób, aby minimalizować różnicę pomiędzy odpowiedzią modelu a odpowiedzią oczekiwaną. Dzięki zastosowaniu różniczkowalnej funkcji celu oraz struktury sieci neuronowej możliwe jest zastosowanie gradientowych metod optymalizacji. Gradient wyznaczany jest poprzez zastosowanie metody propagacji wstecznej błędu [47], która opiera się na łańcuchowej regule wyznaczania pochodnych.

W następnych sekcjach przedstawione zostaną najczęściej stosowane techniki uczenia. W praktyce stosuje się metody optymalizacji ogólnego przeznaczenia takie jak SGD. Oprócz tego stosowane są również algorytmy optymalizacji zaprojektowane specjalnie na potrzeby uczenia sieci neuronowych np. Adam czy RMSProp.

Metoda stochastycznego spadku wzdłuż gradientu

Metoda stochastycznego spadku wzdłuż gradientu (ang. *Stochastic Gradient Descent* – SGD) oraz jej warianty są algorytmami optymalizacji gradientowej, które znajdują zastosowanie w różnych dziedzinach, w tym w uczeniu maszynowym. Mimo swej prostoty algorytm sprawdza się w uczeniu nawet bardzo złożonych struktur sieci neuronowych zawierających miliony parametrów. Losowość (*stochastic*) algorytmu wynika z faktu, że w każdej iteracji do wyznaczenia gradientu stosowana jest losowa część zbioru uczącego nazywana wsadem (ang. *batch*). Rozmiar wsadu wpływa na proces uczenia. Zbyt mały rozmiar wsadu prowadzi do szumu w procesie uczenia, natomiast stosowanie dużego wsadu ograniczone jest przez parametry sprzętowe, m.in. rozmiar pamięci karty graficznej.

Zmiana parametrów modelu w każdej iteracji przebiega zgodnie z równaniem:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \eta \nabla_{\theta_n} \mathcal{L}(\boldsymbol{\theta}_n) \tag{3.20}$$

gdzie:

- $\boldsymbol{\theta}$ – parametry modelu,
- η – współczynnik uczenia,
- $\mathcal{L}$ – funkcja celu (kryterium jakości),
- n – aktualna iteracja.

Metoda SGD z momentem

Niewielkie wartości gradientu lub szum mogą spowolnić uczenie. Aby tego uniknąć, wprowadzono dodatkowy człon v, który akumuluje kierunki gradientów z poprzednich iteracji w postaci średniej eksponencjalnej. Dzięki temu algorytm oprócz aktualnego kierunku gradientu wykorzystuje informacje o poprzednich kierunkach, co ogranicza wpływ szumu i czyni trening bardziej stabilnym. Ponadto, zastosowanie tego członu zmniejsza prawdopodobieństwo utknięcia rozwiązania w obszarze siodłowym lub minimum lokalnym. Metoda SGD z momentem posiada interpretację fizyczną. Człon v może być interpretowany jako szybkość i kierunek zmian parametrów $\boldsymbol{\theta}$ w przestrzeni parametrów.

W metodzie SGD z momentem wprowadzony jest hiperparametr $\alpha \in [0, 1)$, określający wpływ członu v na kierunek zmian parametrów. Aktualizacja parametrów

sieci przebiega w następujący sposób:

$$\boldsymbol{v}_n = \alpha \boldsymbol{v}_{n-1} - \eta \nabla_{\theta_n} \mathcal{L}(\boldsymbol{\theta}_n) \tag{3.21}$$

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \boldsymbol{v}_n \tag{3.22}$$

Metoda SGD z momentem typu Nesterov

Metoda SGD z momentem typu Nesterov jest kolejnym wariantem algorytmu SGD. Algorytm ten różni się od poprzedniej metody sposobem wyznaczania gradientu. Gradient funkcji celu nie jest wyznaczany względem aktualnych parametrów $\boldsymbol{\theta}_n$, a względem parametrów z uwzględnieniem członu $v - \boldsymbol{\theta}_n + \alpha \boldsymbol{v}_{n-1}$. Ta modyfikacja pozwala na szybszą reakcję na zmiany kształtu funkcji celu.

Aktualizacja parametrów w trakcie uczenia odbywa się zgodnie z następującymi wzorami:

$$\boldsymbol{v}_n = \alpha \boldsymbol{v}_{n-1} - \eta \nabla_{\theta_n} \mathcal{L}(\boldsymbol{\theta}_n + \alpha \boldsymbol{v}_{n-1}) \tag{3.23}$$

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \boldsymbol{v}_n \tag{3.24}$$

Metoda Adam

Wartość współczynnika uczenia η jest jednym z kluczowych czynników wpływających na skuteczność treningu. Zbyt niski współczynnik uczenia może spowolnić trening, natomiast zbyt wysoki może spowodować niestabilność procesu uczenia. Jednym z rozwiązań tego problemu, są metody optymalizacji z adaptacyjnym współczynnikiem uczenia. Najbardziej znanymi algorytmami tego typu są AdaGrad [48], RMSProp [49], oraz Adam [15].

Dominującą metodą w tej grupie jest metoda Adam (Adaptive Moment Estimation), w której każdy parametr sieci ma swój osobny, adaptacyjny współczynnik uczenia. Aktualizacja parametrów w metodzie odbywa się zgodnie z równaniami:

$$\boldsymbol{g}_n = \nabla_{\theta_n} \mathcal{L}(\boldsymbol{\theta}_n) \tag{3.25}$$

$$\boldsymbol{s}_{n+1} = \rho_1 \boldsymbol{s}_n + (1 - \rho_1) \boldsymbol{g}_n \tag{3.26}$$

$$\boldsymbol{r}_{n+1} = \rho_2 \boldsymbol{r}_n + (1 - \rho_2) \boldsymbol{g}_n \odot \boldsymbol{g}_n \tag{3.27}$$

$$\hat{\boldsymbol{s}}_{n+1} = \frac{\boldsymbol{s}_{n+1}}{1 - \rho_1^{n+1}} \tag{3.28}$$

$$\hat{\boldsymbol{r}}_{n+1} = \frac{\boldsymbol{r}_{n+1}}{1-\rho_2^{n+1}} \tag{3.29}$$

$$\Delta\boldsymbol{\theta} = -\eta \frac{\hat{\boldsymbol{s}}_{n+1}}{\sqrt{\hat{\boldsymbol{r}}_{n+1}} + \delta} \qquad \text{operacja element po elemencie} \tag{3.30}$$

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \Delta\boldsymbol{\theta} \tag{3.31}$$

gdzie:

- $\boldsymbol{s}$ – estymata pierwszego momentu gradientu
- $\hat{\boldsymbol{s}}$ – korekta estymaty pierwszego momentu gradientu
- $\boldsymbol{r}$ – estymata drugiego momentu gradientu
- $\hat{\boldsymbol{r}}$ – korekta estymaty drugiego momentu gradientu
- ρ_1, ρ_2 – współczynniki zanikania

Harmonogram współczynnika uczenia

Współczynnik uczenia zazwyczaj dobierany jest metodą prób i błędów, co wymaga doświadczenia, intuicji, oraz czasu. Właściwe dobranie współczynnika uczenia jest kluczowe dla osiągnięcia optymalnej wydajności w procesie uczenia. Zbyt duży współczynnik uczenia może powodować oscylacje, a nawet zmianę wartości w niepożądanym kierunku np. wzrost miary w przypadku minimalizacji. Z kolei zbyt niski współczynnik uczenia spowalnia uczenie oraz zwiększa prawdopodobieństwa utknięcia rozwiązania w lokalnym minimum. Ponadto preferowane wartości współczynnika uczenia zmieniają się od fazy treningu. Współczynnik uczenia o większej wartości preferowany jest w początkowych fazach, wraz z postępem uczenia preferowane są coraz niższe wartości.

Krokowa zmiana współczynnika uczenia

Metoda krokowej zmiany współczynnika uczenia stanowi stosunkowo prostą techniką modyfikacji tego hiperparametru podczas treningu. Zmiana współczynnika uczenia może być dokonywana w wybranych chwilach treningu, np. po upływie określonej liczby epok, lub na podstawie obserwacji zmian wartości wybranej miary. W przypadku zmian na podstawie wartości miary, współczynnik uczenia jest zmniejszany, kiedy wartość obserwowanej miary nie nie ulega poprawie w wyznaczonym okresie.

Metoda SGDR

Metoda SGDR (Stochastic Gradient Descent with warm Restart) [16] jest harmonogramem zmian współczynnika uczenia, gdzie do wyznaczenia jego wartości wykorzystuje się funkcję cosinus. Współczynnik uczenia stopniowo maleje wraz z postępem treningu, aż do osiągnięcia ustalonej wartości minimalnej, po czym ponownie jest ustawiany na wartość maksymalną. Nagły wzrost współczynnika uczenia powoduje nagłą i znaczną zmianę wartości parametrów, co prowadzi do pogorszenia mierzonych metryk. Jednak czasowe pogorszenia wyników umożliwia wyjście rozwiązania z lokalnych minimów, co przy dalszym treningu modelu może prowadzić do osiągnięcia lepszej skuteczności.

Współczynnik uczenia dla danej iteracji opisany jest następującym wzorem:

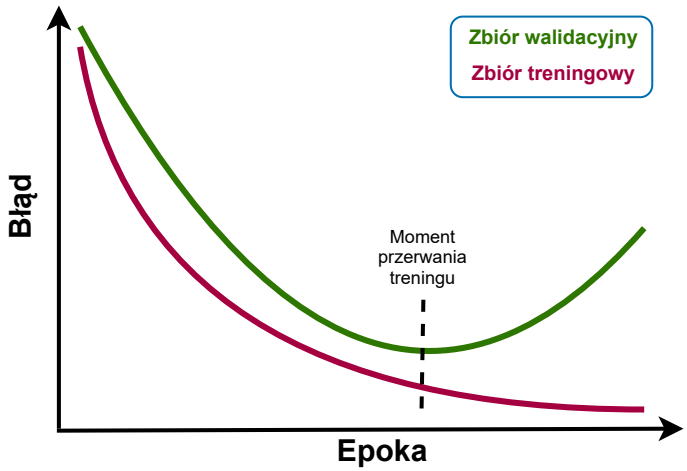
$$\eta_t = \eta^i_{min} + \frac{1}{2}(\eta^i_{max} - \eta^i_{min})(1 + \cos(\frac{T_{cur}}{T_i}\pi)) \tag{3.32}$$

gdzie: η^i_{max} i η^i_{min} to minimalna i maksymalna wartość współczynnika uczenia, T_{cur} to epoka wyrażona w postaci ułamka dziesiętnego, co umożliwia reprezentację np. połowy epoki. Wartość T_i określa długość trwania cyklu w epokach od osiągnięcia maksymalnej do osiągnięcia minimalnej wartości współczynnika uczenia.

3.4. Inne techniki wspomagające uczenie

Wczesne zatrzymanie treningu

Zbiór walidacyjny pozwala określić skuteczność sieci na danych niewykorzystanych w trakcie treningu. Częstym zjawiskiem występującym w trakcie treningu jest spadek, a następne wzrost wartości funkcji celu szacowanej na zbiorze walidacyjnym. Wzrost ten spowodowany jest nadmiernym dopasowaniem sieci do zbioru treningowego. Dalszy trening sieci po osiągnięciu minimalnej wartości funkcji celu mierzonej na zbiorze walidacyjnym jest niewskazany, ponieważ prowadzi do nadmiernego dopasowania i pogorszenia skuteczności. Metoda wczesnego zatrzymania treningu (ang. *early stopping*) wykrywa moment, w którym wartość funkcji celu przestaje się poprawiać i zaczyna wzrastać, a następnie przerywa uczenie (rys. 3.9).



Rysunek 3.9: Metoda wczesnego zatrzymania treningu

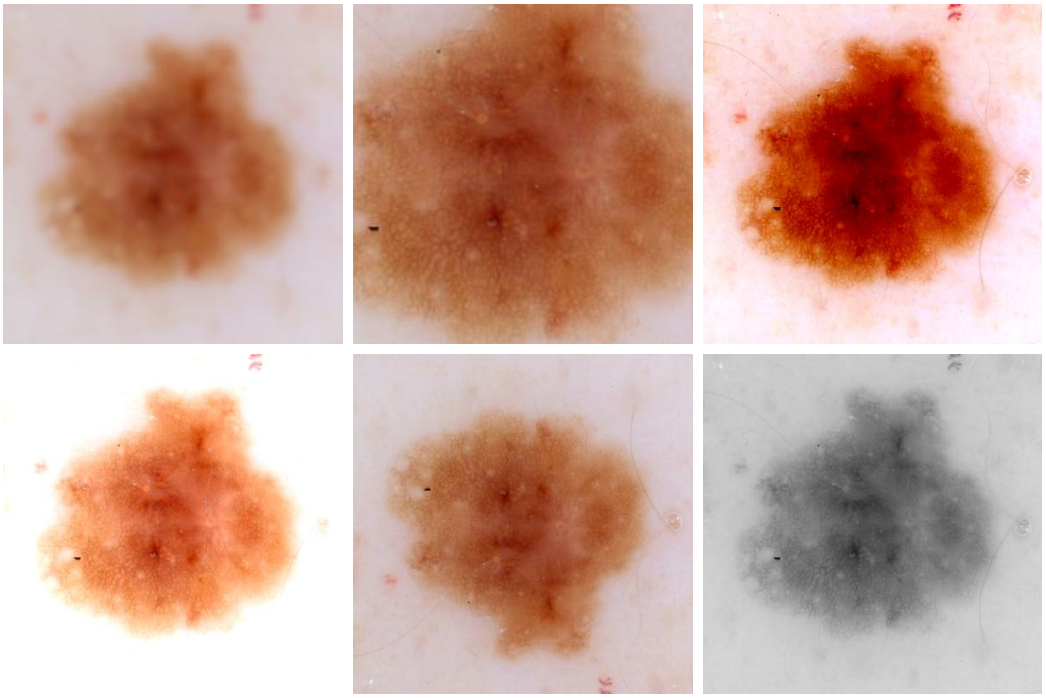
Rozszerzanie danych

Zwiększenie rozmiaru zbioru uczącego jest jednym z najskuteczniejszych sposobów na poprawę generalizacji modelu. Z powodu trudnego i kosztownego pozyskiwania i oznaczania danych, popularnym rozwiązaniem stało się generowanie sztucznych, dodatkowych przykładów uczących (ang. *data augmentation*).

W obszarze przetwarzania obrazu generowanie nowych przykładów uczących może odbywać się poprzez modyfikację obrazów w istniejącym zbiorze danych. Transformacje geometryczne takie jak przesunięcia obrazu o określoną liczbę pikseli, obroty czy skalowanie okazały się skutecznymi metodami zwiększenia zbioru uczącego prowadzącymi do zwiększenia skuteczności głębokich sieci neuronowych. Oprócz transformacji geometrycznych stosowane są również operacje na wartościach pikseli np. dodanie szumu, zmiana jasności i kontrastu. Przykład zastosowania metod augmentacji danych widoczny jest na rys. 3.10 (obraz znamienia pochodzi z bazy ISIC [50]).

Metoda transfer learning

Uzyskanie wysokiej skuteczności modelu, wiąże się z zastosowaniem zbioru uczącego o odpowiednim rozmiarze. Pozyskanie i oznaczenie danych jest kosztowne i czasochłonne, szczególnie w dyscyplinach z licznymi ograniczeniami i regulowanymi przez prawo takich jak medycyna. W przypadkach, kiedy dostępny zbiór danych jest niewielki, stosuje się metodę transfer learning, która prowadzi do poprawy wyników [51], [52].



Rysunek 3.10: Przykład zastosowania metod augmentacji danych

Wykorzystanie metody transfer learning składa się z dwóch etapów. Pierwszy etap polega na wstępnym treningu sieci neuronowej na innym zadaniu, w którym można wykorzystać duży zbiór uczący. We wstępnym treningu zazwyczaj stosowane są duże, publicznie zbiory danych takie jak ImageNet [53], Microsoft COCO [54] czy PascalVOC [55], a trening przebiega w sposób nadzorowany. W trakcie treningu sieć nabywa zdolności ekstrakcji uniwersalnych cech istotnych w zadaniach przetwarzania obrazu. Okazuje się, że w wielu zadaniach przetwarzania obrazu ekstrakcja cech przebiega w podobny sposób tj. realizowane są podobne operacje takie jak wykrywanie krawędzi czy identyfikacja prostych kształtów. Dlatego wstępnie przetrenowana sieć neuronowa, po dostrojeniu parametrów może zostać użyta do realizacji innych zadań. W praktyce, zamiast przeprowadzać wstępne uczenie, stosuje się sieci neuronowe, które zostały wcześniej przetrenowane i udostępnione w publicznych repozytoriach np. udostępnianych wraz z bibliotekami programistycznymi takimi jak Pytorch czy Tensorflow.

Drugi etap polega na treningu na zadaniu docelowym, gdzie zamiast losowej inicjalizacji parametrów jako parametry początkowe sieci stosuje się parametry otrzymane w wyniku wstępnego uczenia. Aby zapobiec utracie wiedzy zdobytej w trakcie wstępnego uczenia, stosuje się współczynnik uczenia o niższej wartości.

Częstą praktyką jest również dostrajanie parametrów tylko części warstw sieci neuronowej.

Zastosowanie metody transfer learning może przynieść korzyści w postaci zwiększonej skuteczności oraz przyspieszenia procesu uczenia. Metoda działa najlepiej, gdy zbiory danych wykorzystane do wstępnego uczenia i uczenia zadania docelowego są do siebie zbliżone. Zastosowanie tej techniki jest najbardziej skuteczne w zadaniach rozpoznawania obrazów z życia codziennego, ponieważ udostępniane jest wiele wstępnie przeuczonych sieci na zadaniu ImageNet. W innych dziedzinach takich jak medycyna czy biologia, zastosowanie metody transfer learning może być ograniczone przez niską dostępność zbiorów danych. Nawet gdy zbiory są dostępne, często są one nieoznaczone, dlatego, korzystne jest zastosowanie metod wstępnego uczenia niewymagających posiadania oznaczeń.

Grupowanie modeli

Grupowanie modeli (ang. *model ensembling*) jest techniką polegającą na agregacji wyjść kilku przeuczonych modeli w celu otrzymania ostatecznej odpowiedzi. Metoda ma na celu zwiększenie skuteczności systemu uczenia maszynowego. Agregacja może być realizowana na różne sposoby, jedną z najprostszych sposobów jest wyznaczenie średniej wyjść modeli. Innym podejściem jest zastosowanie wyjść jako wejścia innego prostego modelu uczenia maszynowego (np. SVM czy regresja logistyczna), w celu wygenerowania ostatecznej odpowiedzi.

W ramach grupowania modeli można zastosować kilka modeli o tej samej strukturze. Innym podejściem jest zastosowanie modeli o różnej strukturze, modeli poddanych różnym procedurom uczenia i trenowanych na różnych zbiorach uczących. Skuteczność metody wynika z losowości podejmowanych decyzji przez model. Źródłami losowości są losowe parametry, losowy dobór przykładów uczących do wsadu, losowe metody rozszerzania danych, a nawet losowość wynikająca z realizacji niektórych operacji matematycznych, gdzie losowe implementacje metody prowadzą do szybszych obliczeń niż implementacje deterministyczne.

Wadą metody grupowania modeli jest konieczność przeprowadzenia wielokrotnego treningu, co skutkuje wydłużeniem czasu uczenia systemu. Dodatkowo generowanie odpowiedzi jest kosztowne obliczeniowo, ponieważ stosowane jest wiele modeli zamiast jednego.

Metoda Dropout

Metoda dropout [17] jest szeroko stosowaną techniką regularyzacji w obszarach takich jak analiza obrazu, mowy czy tekstu. Działanie metody może być interpretowane jako aproksymacja metody grupowania modeli (ang. *model ensembling*). Zastosowanie wielu modeli w generowaniu odpowiedzi zapewnia zazwyczaj lepszą skuteczność niż pojedynczy model [12], jednak wiąże się to z dłuższym uczeniem oraz kosztownym obliczeniowo generowaniem odpowiedzi. Metoda dropout jest próbą ograniczenia tych wad poprzez aproksymację działania wielu modeli przy użyciu pojedynczego modelu.

W praktyce metoda dropout jest reprezentowana jako warstwa, którą można umieścić w strukturze sieci neuronowej. Podczas treningu, warstwa dropout losowo usuwa neurony z poprzedniej warstwy, poprzez zerowanie ich parametrów. Zaletą takiego podejścia jest brak potrzeby modyfikacji w algorytmie propagacji wstecznej. Dodatkowo metoda dropout z powodzeniem może być stosowana z innymi technikami regularyzacji takimi jak L_1 czy L_2.

Inną interpretacją działania metody dropout odnosi się do działania neuronów. Metoda dropout utrudnia wspólną adaptację neuronów. Dzięki temu neurony osiągają większą niezależność działania, ponieważ w trakcie treningu zmieniany jest zbiór wejść neuronów, co zwiększa możliwość sieci do pracy z niekompletnym zestawem cech, co prowadzi do poprawy możliwości generalizacji sieci neuronowej.

Metoda Stochastic Depth

Metoda Stochastic Depth [19] jest metodą regularyzacji stosowaną podczas treningu wielowarstwowych sieci neuronowych. Podobnie jak metoda dropout, wprowadza zmiany w strukturze sieci neuronowej w trakcie uczenia. Metoda Stochastic Depth zmniejsza losowo liczbę warstw w trakcie treningu, co ułatwia uczenie oraz zmniejsza wymagania obliczeniowe.

Dobór liczby warstw w sieciach neuronowych wymaga kompromisu. Niewielka liczba warstw ułatwia przepływ sygnału, zmniejsza prawdopodobieństwo nasycenia funkcji i wystąpienia zanikających gradientów, co prowadzi do łatwiejszego uczenia. Z drugiej strony płytkie sieci neuronowe nie są w stanie reprezentować złożonych funkcji, realizujących złożone zadania takie jak przetwarzanie obrazu. Sieci składające się z wielu warstw mogą realizować złożone zadania, jednak ich trening jest trudny i długotrwały.

Metoda stochastic deptch realizowana jest poprzez wprowadzenie połączeń skracających wokół kilku warstw. Podczas treningu bloki warstw są losowo wyłączane, a sygnał przepływa tylko przez połączenie skracające. Prawdopodobieństwo wyłączenia bloku zmienia się w sposób liniowy wraz z głębokością sieci. Podobnie jak metoda dropout, metoda Stochastic Depth może być interpretowana jako aproksymacja metody grupowania modeli. W praktyce, metoda Stochastic Depth jest o wiele rzadziej stosowana niż metoda dropout.

Podsumowanie metod wspomagających uczenie

Należy zaznaczyć, że zastosowanie jednej z opisanych metod nie wyklucza zastosowania innych w celu osiągnięcia jak najlepszych wyników. Częstą praktyką jest stosowanie metody transfer learning, rozszerzania danych, oraz grupowania modeli równocześnie. Istnieją również wyjątki, na przykład zastosowanie metody dropout oraz batch normalization równocześnie prowadzi do niskich wyników [56].

3.5. Ocena działania systemów klasyfikacji dwuklasowej

Skuteczna implementacja systemu uczenia maszynowego wymaga odpowiedniej ewaluacji modelu, która może być dokonana w sposób ilościowy lub jakościowy. Jakościowa ocena modelu opiera się na analizie działania modelu oraz generowanych wyjść. W analizie jakościowej dużą rolę odgrywają metody objaśnialnej sztucznej inteligencji (ang. *Explainable Artificial Intelligence – XAI*, które umożliwiają wskazanie przesłanek, którymi kierował się model w trakcie generowania odpowiedzi. Do popularniejszych metod w tym obszarze należą: GradCAM [57], LRP [58], LIME [59] czy SHAP [60]. Ocena ilościowa opiera się analizie miar, które niosą informację o poprawności działania modelu. Wnioski wyciągnięte z oceny modelu, mogą posłużyć m.in. ocenie przydatności i zakresu stosowalności modelu, usprawnieniu procesu uczenia, identyfikacji trudnych przypadków, doborze struktury sieci i jej hiperparametrów. Istnieje wiele miar, które dobiera się w zależności od realizowanego zadania przez model. Na przykład w zadaniu regresji stosowane są miary takie jak błąd średniokwadratowy, R^2, w segmentacji IoU, DICE. Natomiast w rozprawie przedstawione zostaną miary stosowane w ewaluacji modeli realizujących zadanie klasyfikacji dwuklasowej.

Dokładność (ang. *accuracy*) jest jedną z najprostszych miar stosowanych w zadaniu klasyfikacji zarówno dwu oraz wieloklasowej. Dokładność określa, jaka część przykładów ze zbioru testowego została poprawnie sklasyfikowana przez model. Miara może przybierać wartości od zera (wszystkie przykłady sklasyfikowane niepoprawnie) do jeden (wszystkie przykłady sklasyfikowane poprawnie). Dokładność może być wyrażona w następujący sposób:

$$\text{acc} = \frac{n}{N} \tag{3.33}$$

gdzie: n – liczba poprawnie sklasyfikowanych przykładów, N – liczba przykładów w zbiorze testowym.

W klasyfikacji dwuklasowej, szczególnie w zastosowaniach medycznych, każdy z przykładów może zostać przypisany do jednej z czterech grup w zależności od prawdziwej oraz przewidywanej przez model klasy:

- TP – (ang. *true positive*) – grupa przykładów prawdziwie pozytywnych,
- FP – (ang. *false positive*) – grupa przykładów fałszywie pozytywnych,
- TN – (ang. *true negative*) – grupa przykładów prawdziwie negatywnych,
- FN – (ang. *false negative*) – grupa przykładów fałszywie negatywnych.

Tabela 3.4: Przykładowa macierz błędów

		Klasa wygenerowana	
		Negatywna	Pozytywna
Klasa rzeczywista	Negatywna	TN	FP
	Pozytywna	FN	TP

Na przykład, w kontekście medycznym wynik pozytywny oznacza chorobę, natomiast wynik negatywny jej brak. Częstą praktyką jest stosowanie macierzy błędów, która przedstawia liczebność każdej z wymienionych grup. Przykładowa macierz błędów przedstawiona jest w tabeli 3.4.

Stosując przedstawiony podział, dokładność można zapisać następująco:

$$\text{acc} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.34}$$

Zastosowanie dokładności nie jest wystarczające do wiarygodnej oceny skuteczności modelu. Dokładność zależna jest od proporcji przykładów pozytywnych i negatywnych w zbiorze testowym. Na przykład, w zbiorze złożonym z 95 przykładów pozytywnych i 5 przykładów negatywnych, model zwracający wartość pozytywną dla każdego przykładu osiągnie dokładność wynoszącą 0,95. Dlatego, w celu lepszej oceny modelu stosowane są dodatkowe miary takie jak czułość, swoistość oraz ROC AUC.

Czułość określa zdolność wykrycia przez model przykładów pozytywnych pośród wszystkich przykładów pozytywnych. Czułość równa 1 oznacza, że udało się wykryć wszystkie przykłady pozytywne. Miara definiowana jest następująco:

$$\text{sst} = \frac{TP}{TP + FN} \tag{3.35}$$

Miarą podobną do czułości jest swoistość, która określa zdolność wykrycia przykładów negatywnych wśród wszystkich przykładów negatywnych. Wartość miary równa 1 oznacza, że wykryto wszystkie negatywne przykłady wśród badanych przykładów. Swoistość definiowana jest następująco:

$$\text{spc} = \frac{TN}{TN + FP} \tag{3.36}$$

Analiza czułości i swoistości wraz z dokładnością daje lepszy obraz skuteczności działania modelu.

Precyzja jest kolejną miarą stosowaną w ewaluacji systemów klasyfikacji dwuklasowej. Miara określa stosunek poprawnie przewidzianych pozytywnych przypadków, do liczby przypadków, które zostały określone przez model jako pozytywne. Precyzję, czułość oraz swoistość można kontrolować poprzez zmianę progu klasyfikacji. Zwiększenie precyzji prowadzi do zmniejszania czułości. Na przykład, klasyfikator, który określa każdy przypadek jako pozytywny, będzie miał czułość wynoszącą 1. Precyzja będzie jednak niska, ponieważ niewielka część przykładów oznaczonych jako pozytywne, faktycznie jest pozytywna. Precyzja opisana jest następującym wzorem:

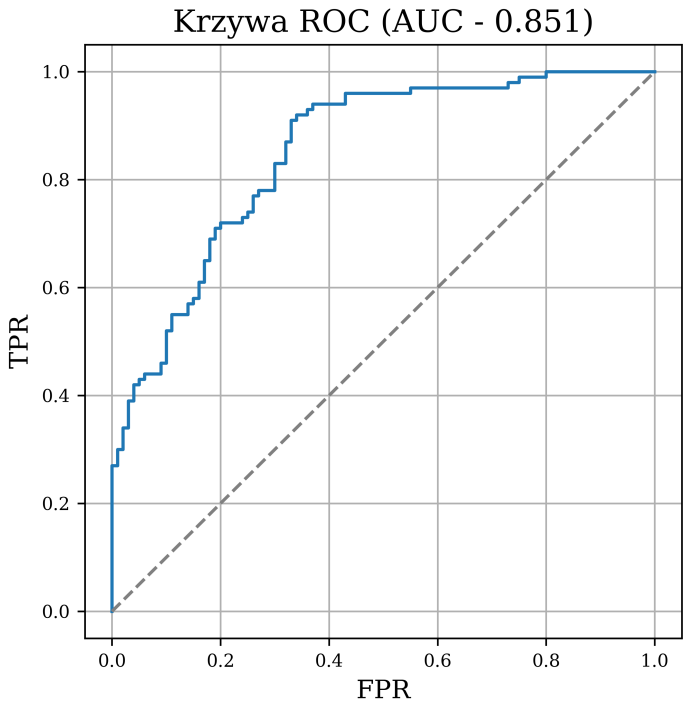
$$\text{prc} = \frac{TP}{TP + FP} \tag{3.37}$$

Czułość oraz precyzja dają dobry obraz skuteczności modelu. Miara F1 łączy te dwie metryki poprzez wyznaczenie średniej harmonicznej tych wartośći, zgodnie ze

wzorem:

$$\mathrm{F1} = \frac{2 * (\mathrm{prc} * \mathrm{sst})}{\mathrm{prc} + \mathrm{sst}} = \frac{2TP}{2TP + FP + FN} \tag{3.38}$$

Kolejną metodą oceny modelu jest analiza krzywej ROC (ang. *Receiver Operating Characteristic*). Krzywą przedstawiana jest na wykresie, gdzie oś x reprezentuje wartość 1 − swoistość, a oś y reprezentuje czułość. Zmieniając próg klasyfikacji i odczytując te wartości, uzyskuje się kolejne punkty na krzywej ROC. Przykładowa krzywa ROC widoczna jest na rysunku 3.11.



Rysunek 3.11: Przykładowa krzywa ROC

Powierzchnia pod krzywą ROC jest miarą skuteczność klasyfikatora, gdzie większa powierzchnia pod wykresem oznacza lepszą skuteczność. Miara oznaczana jest skrótem ROC AUC (ang. *Area Under Curve*) i może przyjmować wartości od 0,5 do 1.

Przedstawione miary nie mogą być stosowane jako funkcja celu w procesie uczenia sieci neuronowych, ponieważ nie są różniczkowalne. W trakcie uczenia, w problemie klasyfikacji dwuklasowej, stosuje się miarę binarnej entropii krzyżowej (ang. *binary cross-entropy*). Miara ta jest różniczkowalna, dzięki czemu może być stosowana jako funkcja celu w procesie uczenia metodami gradientowymi. Binarna entropia krzyżowa

dla pojedynczego przykładu opisana jest następującym wzorem:

$$\mathcal{L}_{BCE}(y_i, p_i) = -\left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\right) \tag{3.39}$$

gdzie:

y_i – odpowiedź oczekiwana (0 lub 1),

p_i – odpowiedź modelu (w zakresie od 0 do 1).

Dla $y_i = 1$ funkcja przyjmuje postać:

$$\mathcal{L}_{BCE}(y_i, p_i) = -\left(y_i \log(p_i)\right) \tag{3.40}$$

W tej formie, minimum funkcji znajduje się w punkcie $p_i = 1$, a dla $\lim_{p_i \to 0} \mathcal{L}_{BCE}(y_i = 1, p_i) = \infty$. Więc minimalizacja tej funkcji celu dla pozytywnego przykładu wejściowego, prowadzi do generowania na wyjściu modelu wartości $p_i = 1$. Funkcja zachowuje się w sposób analogiczny dla $y_i = 0$ Zazwyczaj wartości tej miary nie raportuje się w literaturze.

3.6. Procedura testowania modeli

Przedstawione miary umożliwiają kompleksową ocenę działania modelu, jednak wymagają odpowiedniego zastosowania. W uczeniu maszynowym w celu poprawnej oceny modelu dostępny zbiór danych dzieli się na trzy grupy: zbiór uczący, zbiór walidacyjny oraz zbiór testowy. Każdy z tych zbiorów pełni inną funkcję.

Zbiór treningowy stosowany jest w trakcie uczenia modelu, do doboru jego parametrów przez algorytm optymalizacji. Z kolei, zbiór walidacyjny stosowany jest do nadzorowania treningu, doboru hiperparametrów i struktury modelu. Technika early-stopping wyznaczająca moment przerwania treningu jest jednym z przykładów wykorzystania zbioru walidacyjnego. Po doborze ostatecznego modelu wykorzystując zbiór walidacyjny, końcowa ocena modelu dokonywana jest na zbiorze testowym.

Technika sprawdzianu krzyżowego

Aby dokładnie określić skuteczność modelu, konieczne jest zastosowanie dużych zbiorów walidacyjnych i testowych. Zbyt mały rozmiar tych zbiorów prowadzi do szumu w metrykach, utrudnia dobór hiperparametrów oraz powoduje niepoprawne działanie

metod takich jak wczesne zatrzymanie treningu. Na przykład, jeśli zbiór testowy składa się z zaledwie 50 obrazów, to jeden poprawnie sklasyfikowany obrazu prowadzi do wzrostu dokładności aż o 2%.

W przypadku, kiedy dostępny zbiór danych jest niewielki można zastosować metodę k-krotnego sprawdzianu. Metoda polega na podziale zbioru na k grup o równej liczbie przykładów. Następnie dokonuje się k treningów, gdzie podczas każdego treningu jedna z grup pełni role zbioru testowego, a pozostałe grupy pełnią rolę zbioru treningowego. Po zakończeniu treningów, wyniki uzyskane na zbiorach testowych uśrednia się, co umożliwia otrzymanie wiarygodnych wyników oceny modelu. Wadą stosowania tej metody jest konieczność przeprowadzania wielu treningów.

4. Rozwiązania architektoniczne w głębokich sieciach neuronowych

4.1. Wprowadzenie

Struktura sieci neuronowej oraz hiperparametry mają znaczący wpływ na osiągane wyniki. Projektowanie struktury sieci neuronowej jest złożonym zadaniem, dlatego częstą praktyką jest stosowanie gotowych rozwiązań opracowanych przez innych badaczy. Ręczny dobór hiperparametrów oraz architektury wymaga przeprowadzenia wielu eksperymentów, co jest kosztowne obliczeniowo oraz czasochłonne. Proces ten wymaga również wiedzy, doświadczenia i intuicji.

Struktury sieci neuronowych oraz metody i techniki uczenia opisywane w literaturze często poddawane są testom z wykorzystaniem tzw. *benchmarków*, czyli zbiorów danych przygotowanych w celu oceny i porównywania skuteczności proponowanych rozwiązań. Zbiory te zazwyczaj składają się z dużej liczby zdjęć i klas, a liczebność klas jest równomierna. Przykładem takiego zbioru jest zbiór ImageNet zawierający około 1 mln obrazów podzielonych na 1000 klas, gdzie każda klasa zawiera około 1000 obrazów. Zbiór wykorzystywany jest do testowania rozwiązań klasyfikacji obrazów.

Jednym z istotnych czynników wpływających na skuteczność modelu jest rozmiar zbioru uczącego użytego w trakcie treningu. Najczęściej w praktycznych zastosowaniach głębokich sieci neuronowych dostępne zbiory danych nie są tak liczne, jak w benchmarkach. Zazwyczaj liczba obrazów jest niewielka, dodatkowo występują dysproporcje w liczebności przykładów należących do poszczególnych klas. Implementacja systemu uczenia maszynowego z wykorzystaniem małego zbioru jest

trudna i wymaga specjalnego podejścia. Przykładem takich zadań, są zadania w obszarze medycyny, gdzie ilość dostępnych danych jest niewielka, a proces ich pozyskania jest kosztowny i czasochłonny.

W ramach niniejszej rozprawy podjęto temat analizy zadań małej skali, czyli takich, w których dostępny zbiór uczący jest niewielki. W celu testowania wybranych metod i technik wybrano zbiór znamion skórnych zawierający kilkanaście tysięcy obrazów. Oprócz niewielkiego rozmiaru zbiór cechuje się zróżnicowanym rozmiarem obrazów, różną proporcją szerokości i wysokości oraz dysproporcją w liczebności klas. Wymienione cechy zbioru utrudniają opracowanie skutecznego klasyfikatora. Kolejną cechą odróżniającą ten zbiór od typowych zbiorów benchmarkowych jest trudność samego zadania. Zbiory benchmarkowe zwykle zawierają klasy znane z życia codziennego (np. samochód, łódka, samolot, ptak), których rozróżnienie nie wymaga specjalistycznej wiedzy, a dokładność osiągana przez człowieka jest bardzo wysoka [61]. W przypadku zadań medycznych konieczne jest posiadanie specjalistycznej wiedzy i długoletniego doświadczenia. Co więcej, nawet bardzo doświadczeni lekarze nie osiągają stuprocentowej dokładności w wykrywaniu schorzeń [3].

W podjętych badaniach skupiono się nad wpływem architektury sieci neuronowej, metod oraz hiperparametrów na osiągane wyniki. Zbadano wpływ funkcji aktywacji oraz metod regularyzacji na dokładność. Przeanalizowano także, w jakim stopniu metoda transfer learning wpływa na poprawę dokładności klasyfikacji. Dodatkowo przeanalizowano wpływ grupowania modeli na osiągane wyniki. Wyniki otrzymane w ramach badań opublikowano w [27].

4.2. Zadanie klasyfikacji znamion skórnych

Nowotwór skóry jest jednym z najczęściej diagnozowanych nowotworów. Każdego roku na świecie wykrywa się kilkaset tysięcy zachorowań, a kilka tysięcy osób rocznie umiera. W samych Stanach Zjednoczonych Ameryki rocznie wykrywa się 5,4 mln przypadków [62]. Odsetek pacjentów, u których choroba kończy się śmiercią, jest wysoki, jednak wczesne wykrycie choroby zwiększa prawdopodobieństwo powrotu pacjenta do zdrowia [3].

Badanie zmian skórnych pod kątem nowotworu polega na obserwacji zmiany okiem nieuzbrojonym lub przy pomocy specjalnego urządzanie zwanego dermatoskopem. Obserwacja okiem nieuzbrojonym umożliwia wykrycie choroby

w zaawansowanym stadium, jednak ten sposób nie sprawdza się w diagnozie wczesnych faz choroby. Dlatego, w celu wykrycia wczesnych etapów rozwoju choroby stosowany jest dermatoskop, który umożliwia obserwację znamienia w dużym powiększeniu. Dermatoskop wyposażony jest w układ optyczny oraz dodatkowe źródło światła pozwalające na dokładniejszą analizę badanej zmiany. Stosowane dermatoskopy można podzielić ze względu na sposób stosowania: na kontaktowe oraz bezkontaktowe, oraz ze względu na zastosowany rodzaj światła – spolaryzowane i niespolaryzowane. Każdy rodzaj dermatoskopii – kontaktowa bez polaryzacji, kontaktowa z polaryzacją oraz bezkontaktowa z polaryzacją ma różne właściwości i umożliwia wykrywanie innych cech wizualnych badanych zmian. Na przykład żyły oraz czerwone obszary są lepiej widoczne w dermatoskopie ze światłem spolaryzowanym, a obszary o jasnych kolorach lepiej widoczne w dermatoskopie o świetle niespolaryzowanym [63].

Środowisko medyczne opracowało różne metody diagnostyki znamion skórnych. Do najczęściej stosowanych należą m.in. metoda 7-punktowa, metoda Menziesa oraz metoda ABCD [64]. Metody te opierają się na wizualnej ocenie cech znamienia.

W celu zaprezentowania sposobu działania metod przedstawiona zostanie metoda ABCD. Metoda opiera się na ocenie czterech kryteriów: asymetrii (A – *assymetry*), krawędzi (B – *border*), koloru (C – *color*) oraz widocznych struktur wewnątrz znamienia (D – *differential structures*) [64]. Każde z kryteriów jest poddawane ocenie, a następnie przypisywane są punkty, przy czym większa liczba punktów wskazuje na większe ryzyko obecności zmiany nowotworowej.

Pierwszym kryterium oceny jest symetria. Za pełną symetrię przyznaje się zero punktów, za jedną oś symetrii jeden punkt, a za brak osi symetrii dwa punkty. Ocena krawędzi znamienia odbywa się poprzez podział krawędzi na osiem fragmentów. Następnie za każdą krawędź przypisuje się jeden punkt, jeśli krawędź jest poszarpana, urywa się lub jest niejednolita kolorystycznie. Ocena koloru znamienia może wynosić od zera do sześciu punktów. Za obecność czerwonego, białego, jasnobrązowego, ciemnobrązowego, czarnego, lub szaro-niebieskiego koloru przyznaje się po jednym punkcie. Punkt za kolor biały przyznawany jest tylko wtedy, gdy jest jaśniejszy niż kolor skóry. Punkty za obecność struktur przyznawane są w przypadku występowania siatki, kropek, rozgałęzionych smug lub globulek. Przy ocenie kropek wymagana jest obecność minimum dwóch, natomiast w przypadku globulek wystarczy obecność jednej do przyznania punktu.

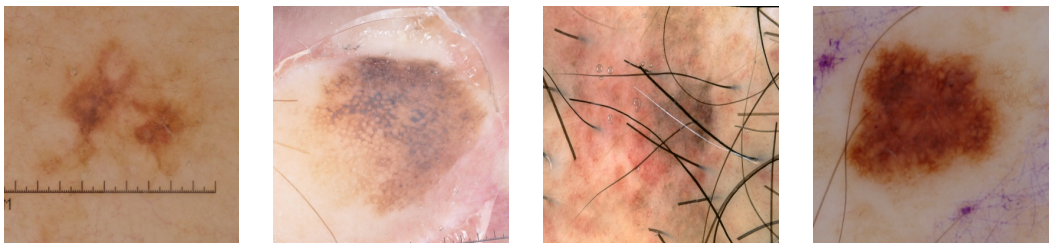
W celu postawienia diagnozy należy oszacować punktacje w ramach każdego z kryteriów. Następnie, punkty są sumowane z uwzględnieniem wag z tabeli 4.1. Wynik poniżej 4,75 świadczy o zmianie łagodnej, 4,75-5,45 oznacza znamię podejrzane i rekomendację dokonania biopsji, natomiast wynik powyżej 5,45 wskazuje na wysokie podejrzenie zmiany złośliwej.

Tabela 4.1: Metoda ABCD – kryteria oceny i punktacja

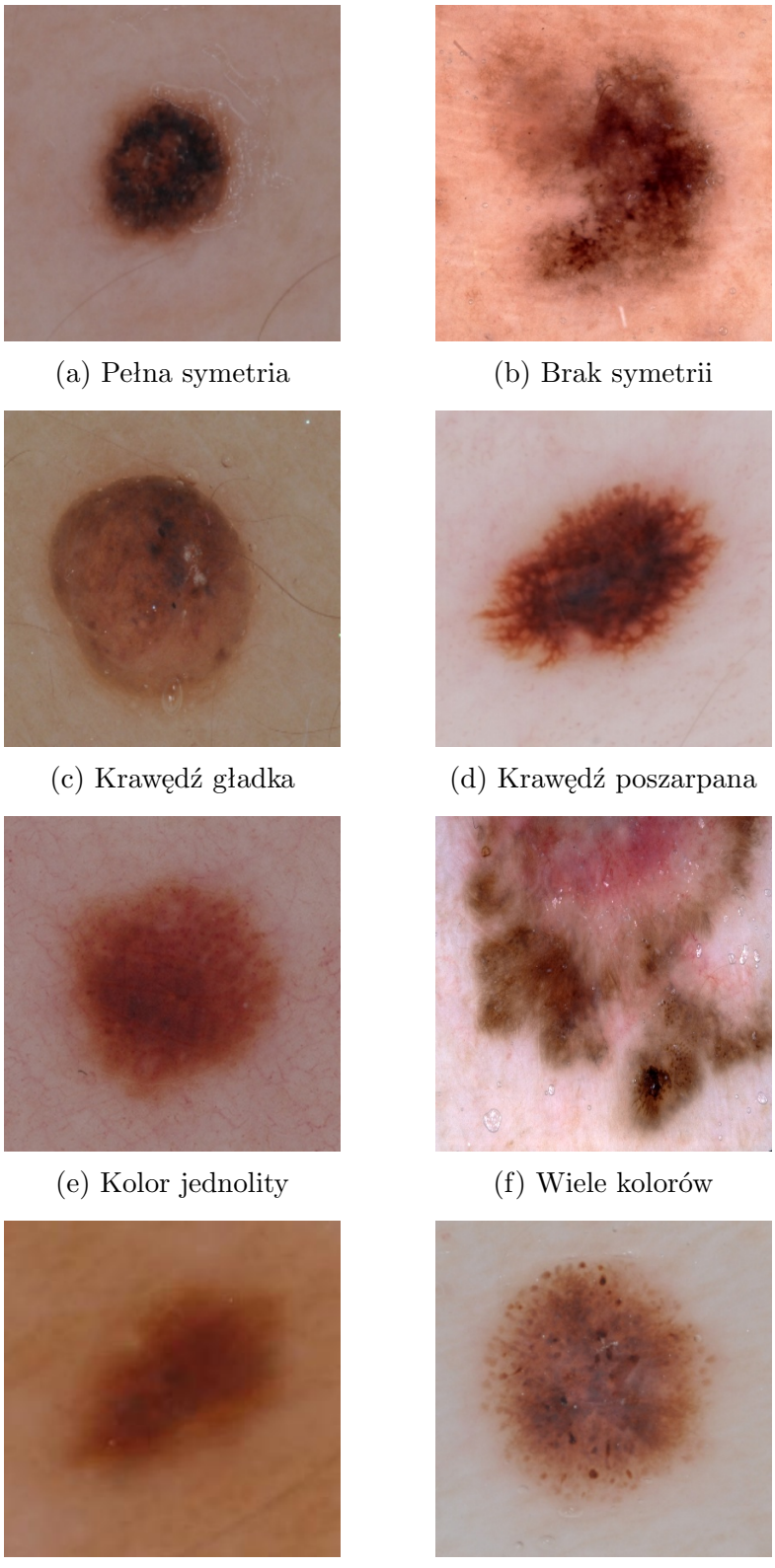
Kryterium	**Wynik**	**Współczynnik**
Asymetria	0-2 pkt	1,3
Krawędź	0-8 pkt	0,1
Kolor	1-6 pkt	0,5
Struktury	1-5 pkt	0,5

Przykłady różnic w ramach różnych kryteriów oceny widoczne są na rys. 4.1 (obrazy pochodzą z bazy ISIC [50]).

Głównym zadaniem systemów komputerowej analizy znamion skórnych jest wykrycie nietypowych zmian na zdjęciu. Duże podobieństwo między przypadkami zdrowymi i chorymi powoduje trudności w implementacji takich systemów. Kryteria oceny cech są słabo zdefiniowane np. nie jest możliwe wyznaczenie ostrej granicy, która określałaby kolor znamienia, który jednoznacznie wskazywałby na złośliwość. Brak precyzyjnych reguł oraz duże podobieństwo powoduje, że lekarze i algorytmy mogą stawiać różne diagnozy dla tego samego przypadku. Automatyczne systemy narażone są również na wpływ czynników technicznych, takich jak jakość obrazów, warunki oświetleniowe czy obecność artefaktów na obrazie. Ponadto, artefakty takie jak obecność owłosienia na skórze, oznaczenia graficzne nanoszone obraz, pęcherzyki powietrza, żel nanoszony na skórę, nie powinny być brane pod uwagę w procesie wnioskowania, ale mogą wpłynąć na automatyczną diagnozę [65]. Przykłady artefaktów pokazano na rys. 4.2 (obrazy pochodzą z bazy ISIC [50]).



Rysunek 4.2: Przykłady artefaktów obecnych w zbiorach danych znamion skórnych. Od lewej: naniesiona linijka, żel, owłosienie, ślady markera



(a) Pełna symetria (b) Brak symetrii

(c) Krawędź gładka (d) Krawędź poszarpana

(e) Kolor jednolity (f) Wiele kolorów

(g) Struktura jednolita (h) Występujące kropki

Rysunek 4.1: Kryteria oceny metody ABCD – przykład

4.2.1 Przegląd automatycznych systemów analizy znamion skórnych

Klasyfikacja zmian skórnych jest obszarem cieszącym się zainteresowaniem społeczności naukowej. Podobnie jak w innych dziedzinach przetwarzania obrazu, dawniej stosowane rozwiązania opierały się na metodach ekstrakcji cech i wykorzystaniu prostych klasyfikatorów lub metod regułowych. Te podejścia są określane dalej jako rozwiązania klasyczne. Współcześnie, dominującym rozwiązaniem stały się głębokie sieci neuronowe.

Metody klasyczne opierały się przede wszystkim na odwzorowaniu i automatyzacji procesu wnioskowania lekarzy. Algorytmy ekstrakcji cech służyły do pozyskiwania informacji o kolorze, symetrii czy kształcie znamienia. Przykładem zastosowania złożonej ekstrakcji cech jest praca [66]. Algorytm ekstrahował cechy, które można podzielić na trzy grupy: cechy opisujące kształt, kolor i teksturę. Cechy opisujące kształt obejmowały m.in.: powierzchnię, asymetrię czy kompaktowość zdefiniowaną jako stosunek powierzchni znamienia do powierzchni koła opisanego na znamieniu. Cechy opisujące kolor to m.in. średnia oraz odchylenie standardowe kanałów koloru, reprezentowane w sześciu różnych przestrzeniach barw. Ciekawym przykładem ekstrakcji cech motywowanej wiedzą kliniczną jest analiza centroidów znamienia. Zdrowe znamiona charakteryzują się jednolitą jasnością, natomiast zaburzenia w jasności mogą świadczyć o złośliwości. W celu wykrycia zaburzeń w jasności autorzy zaproponowali pomiar odległości pomiędzy centroidem geometrycznym a cetroidem jasności. Centroid jasności wyznaczany był przy użyciu ważonej metody, gdzie jasność pikseli odgrywała rolę wag. Duża odległość między tymi centroidami wskazywała na niejednolitą jasność, co mogło świadczyć o chorobie. Cechy opisujące tekstury otrzymano stosując metodę GLCM [67]. Łącznie w systemie użyto 437 cech (11 opisujących kształt, 354 opisujące kolor oraz 72 opisujące teksturę). Dokonano selekcji cech, które wykorzystano do trenowania klasyfikatora SVM. Autorzy przeprowadzili analizę zależności pomiędzy liczbą cech a osiąganą dokładnością. Do uzyskania najlepszego wyniku wystarczyło około 20 wybranych cech. Dodanie kolejnych cech nie prowadziło do poprawy wyników.

W pracy [68] zastosowano metody progowania i wykrywania krawędzi do segmentacji znamion. Znamię poddano analizie, w wyniku której wyodrębniono cechy takie jak powierzchnia, długość krawędzi, czy stopień asymetrii. Cechy wykorzystano jako wejście systemu regułowego, którego sposób wnioskowania opierała się o metodę

Stolza. W innym rozwiązaniu [69] zaproponowano system oparty o klasyfikator SVM, którego wejście stanowiło pięć cech opisujących asymetrię, jedna cecha opisującą krawędzie oraz cztery cechy opisujące kolor. Autorzy [70] opracowali algorytm generujący cechy odpowiadające metodzie ABCD. Zaproponowany algorytm dokonywał konwersji kolorów do przestrzeni barw CIEL*a*b, usuwał owłosienie i dokonywał segmentacji znamienia. Z wyodrębnionego znamienia generowano informacje dotyczące asymetrii, przebiegu krawędzi, koloru oraz widocznych struktur. Następnie dokonano selekcji cech, a do klasyfikacji zastosowano algorytm SVM.

W ostatnich latach obserwuje się szybki rozwój metod wykorzystujących głębokie sieci neuronowe. Istotny wpływ na rozwój miało pojawienie się konkursów, w których co roku udostępniane były nowe zbiory danych. Jednym z najbardziej znanych konkursów z tego obszaru jest ISIC (International Skin Imaging Collaboration) Challenge [50], organizowany w latach 2016-2020. Zbiory danych udostępnianie w ramach konkursów wykorzystywane były przez naukowców w realizowanych pracach badawczych.

Jedną z najbardziej znanych prac naukowych na temat rozpoznawania chorób skóry jest praca [3]. W badaniu wykorzystano bardzo duży zbiór danych liczący około 130 tys. obrazów. W rozwiązaniu zastosowano strukturę GoogleLeNet Inception, która była wstępnie uczona na zbiorze ImageNet. Większość zdjęć w bazie danych pochodziła z 18 publicznych zbiorów danych, a także z Centrum Medycznego Uniwersytetu Stanforda. Zdjęcia charakteryzowały się dużą różnorodnością, różnymi warunkami oświetleniowymi i przybliżeniem. Zdjęcia wykonane zostały rozmaitymi urządzeniami, jednak tylko niewielka część obrazów stanowiła obrazy dermatoskopowe. Sieć uczyła się przypisywania zmianie jednej z 757 klas. Skuteczność rozpoznawaniu rozmaitych chorób skóry była porównywalna do skuteczności dermatologów.

Autorzy [71] wykorzystali około 5000 zdjęć do wytrenowania klasyfikatora, który przyporządkowuje zdjęciom jedną z 14 klas. W pracy zastosowano architekturę GoogleLeNet, przetrenowana na zbiorze ImageNet. Wykorzystano zdjęcia o dużym rozmiarze 1000×1000 pikseli. W celu poprawy generalizacji zastosowano metody rozszerzania danych złożone z obrotów, rozmycia oraz zmian jasności.

W pracy [72] autorzy zastosowali zmodyfikowany blok sieci ResNet. W module, oprócz połączenia realizującego funkcję tożsamościową, wprowadzono moduł atencji. Mapa atencji generowana jest na podstawie wyjścia ostatniej warstwy konwolucyjnej

w module. W rozwiązaniu zastosowano atencję przestrzenną, wzmacniającą bądź osłabiająca poszczególne piksele mapy cech.

Mechanizm atencji wykorzystany został również w [73], gdzie zastosowano trzy ekstraktory cech: VGG16, DenseNet161 oraz ResNet50. Reprezentacje wygenerowane przez ekstraktory zostały połączone w jeden duży wektor cech. Jedną z wyróżniających cech tego rozwiązania była jego zdolność do hierarchicznego przypisywania klas. W tym celu zastosowano rekurencyjną sieć neuronową opartą o mechanizm LSTM, która generowała klasy o rosnącym poziomie szczegółowości. Wejściem bloku były: wygenerowana klasa wyższego poziomu, stan ukryty oraz wektor cech obrazu. Oprócz mechanizmu rekurencyjnego zastosowano mechanizm atencji, który operowały na poziomie kanałów i pikseli, wzmacniając obszary istotne dla modułu LSTM. Moduł atencji, oprócz poprawy dokładności, umożliwił wizualizację obszarów znamienia, które miały istotny wpływ na decyzję podejmowaną przez sieć.

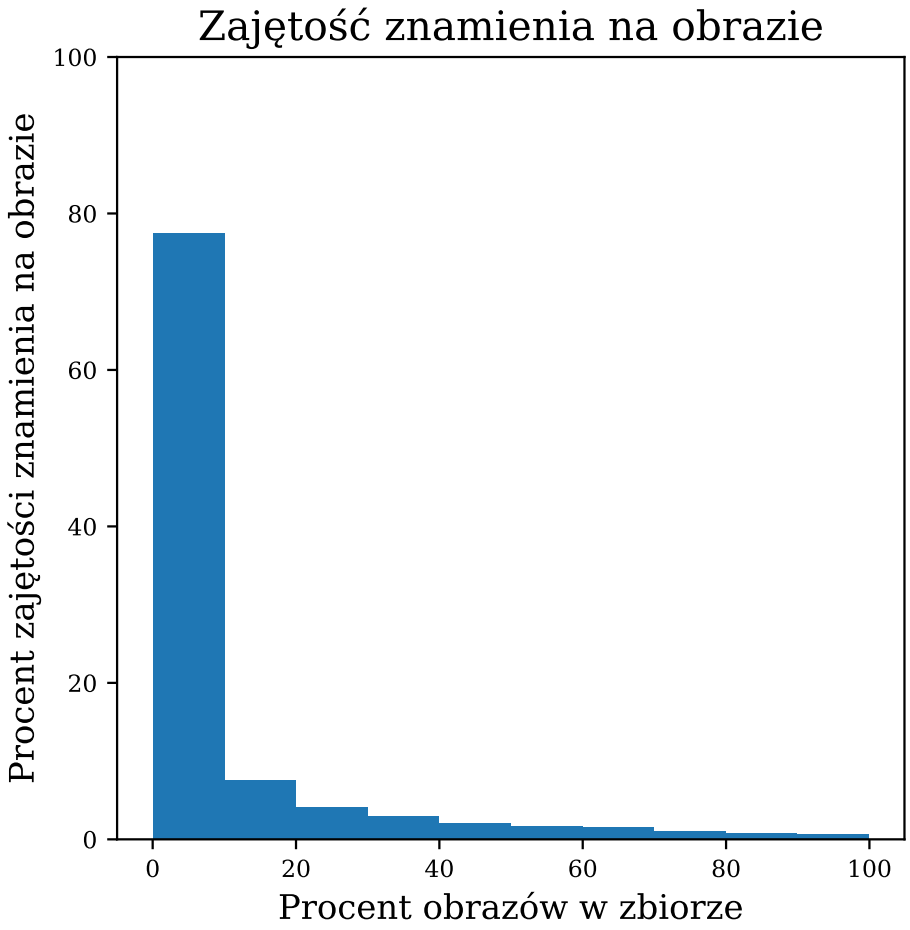
4.2.2 Opis bazy danych stosowanej w badaniach

W realizowanych badaniach wykorzystano zbiór obrazów znamion skórnych udostępniony publicznie przez organizację ISIC (International Skin Imaging Collaboration) [50]. Zbiór ten powstał we współpracy środowiska akademickiego i komercyjnego, w celu wsparcia rozwoju algorytmów klasyfikacji znamion skórnych. Baza składa się ze zdjęć dermatoskopowych wysokiej jakości i rozdzielczości, zebranych z klinik na całym świecie od pacjentów w różnym wieku i różnej płci. Do każdego obrazu przypisana jest diagnoza i dane dotyczące pacjenta. Diagnozy zostały postawione na podstawie badania laboratoryjnego, co zapewnia znacznie większą dokładność oznaczeń niż w przypadku diagnozy stawianej na podstawie obserwacji znamion przy pomocy dermatoskopu.

Wszystkie obrazy wraz z maskami zostały pobrane ze strony `https://www.isic-archive.com` za pomocą dostępnego API w listopadzie 2017 roku. Zbiór danych składa się z około 13 tys. obrazów, z których każde posiada binarną maskę wskazująca położenie znamienia na obrazie. Szczegółowe statystyki dotyczące zbioru są przedstawione w tabeli 4.2. Mimo że zdjęcia charakteryzują się dużymi rozmiarami, to samo znamię zajmuje tylko niewielką część obrazu, co jest widoczne na rys. 4.3.

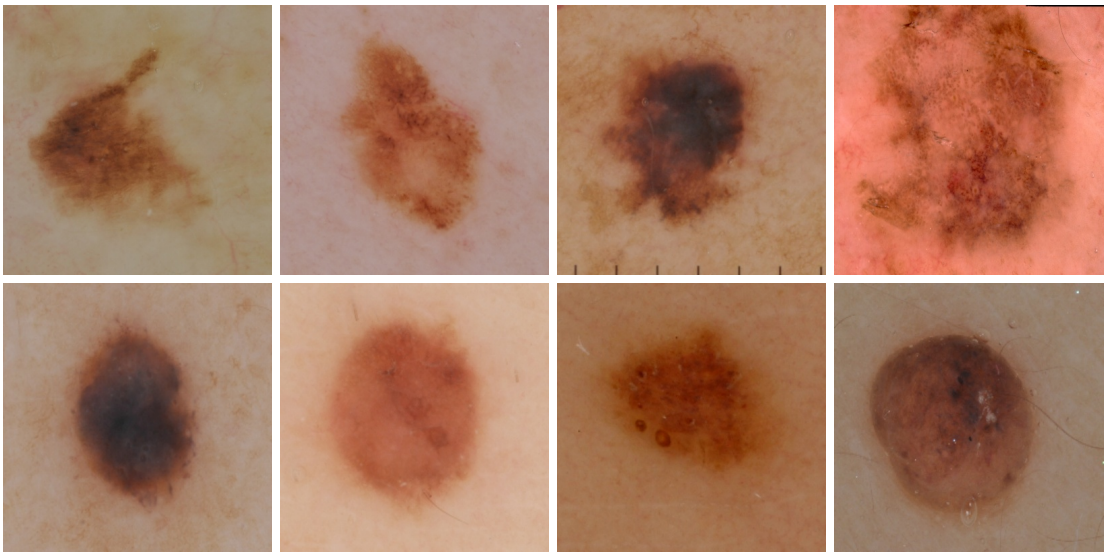
Tabela 4.2: Statystyki obrazów w zbiorze danych

Parametry obrazu	**Minimum**	**Maksimum**	**Średnia**	**Odchylenie standardowe**
Rozmiar (mln px)	0,384	30,449	7,976	6,406
Szerokość (px)	576	6768	3225	1240
Wysokość (px)	540	6688	2163	805
Powierzchnia znamienia (px)	2277	15011767	733661	2074392
Zajętość znamienia na obrazie (%)	1	99	9	17



Rysunek 4.3: Obszar znamienia na obrazie

Rozkład obrazów w badanym zbiorze danych jest nierównomierny – około 12500 obrazów reprezentuje znamiona niezłośliwe, natomiast tylko około 1100 obrazów reprezentuje znamiona złośliwe. Wybrane przykłady ze zbioru danych zaprezentowane są na rys. 4.4.



Rysunek 4.4: Przykład obrazów z bazy danych ISIC. W pierwszym wierszu znajdują się znamiona złośliwe, w drugim łagodne

4.3. Opis przeprowadzonych eksperymentów

W ramach pracy doktorskiej zaplanowano szereg eksperymentów mających na celu ocenę wpływu struktury sieci, metod uczenia oraz metod regularyzacji. Po obszernym badaniu literatury zdecydowano się zastosować rodzinę sieci VGG, która w czasie prowadzenia badań cieszyła się popularnością w wielu rozwiązaniach oraz uzyskiwała wysoką dokładność w zadaniu ImageNet. Struktura tych sieci jest stosunkowo prosta i nie zawiera skomplikowanych rozwiązań architektonicznych, co ułatwia analizę wyników.

Na ogół, pojemność modelu powinna być dobrana w zależności od trudności zadania oraz ilości posiadanych danych. Rodzina sieci VGG została pierwotnie zaprojektowana do klasyfikacji 1000 różnych obiektów, a zbiór uczący zawierał około 1 mln obrazów. Jednak zbiór znamion skórnych stosowany w eksperymentach zawiera zaledwie 13 tys. obrazów, co jest znacznie mniej w porównaniu do rozmiaru zbioru

ImageNet. Z tego powodu podejrzewano, że zmniejszenie rozmiaru sieci neuronowej i liczby parametrów, może ułatwić proces uczenia i przyczynić się do poprawy wyników. Dodatkowo stosowanie zbyt dużych modeli w stosunku do dostępnego zbioru danych może prowadzić do zjawiska nadmiernego dopasowania.

W ramach przeprowadzonych eksperymentów poddano testom struktury VGG11 oraz VGG16, składające się odpowiednio z 11 i 16 warstw. Dodatkowo zaproponowano mniejszą, 8-warstwową sieć neuronową VGG8, zbudowaną w sposób analogiczny do sieci VGG11 oraz VGG16. Struktura sieci badanych modeli przedstawiona jest w tabeli 4.3.

Tabela 4.3: Architektury analizowane w ramach eksperymentów

VGG8	**VGG11**	**VGG16**
conv64	conv64	conv64
		conv64
	maxpool	
conv128	conv128	conv128
		conv128
	maxpool	
conv256	conv256	conv256
	conv256	conv256
		conv256
	maxpool	
conv512	conv512	conv512
	conv512	conv512
		conv512
	maxpool	
conv512	conv512	conv512
	conv512	conv512
		conv512
	maxpool	
	FC-1024	
	FC-1024	
	Neuron sigmoidalny	

Rozmiar wejścia każdej sieci neuronowej przedstawionej w tabeli 4.3 wynosi $224 \times 224 \times 3$ pikseli, co odpowiada kolorowemu obrazowi o rozmiarze 224×224

z trzema kanałami koloru RGB. Sieci składają się z warstw konwolucyjnych, oznaczonych w tabeli jako `conv`, gdzie liczba po oznaczeniu odpowiada liczbie filtrów konwolucyjnych w danej warstwie, np. `conv-64` oznacza warstwę konwolucyjną z 64 filtrami. Każdy filtr konwolucyjnych ma rozmiar $3 \times 3 \times n$, a krok (ang. *stride*) wynosi 1. Aby zachować taki sam rozmiar obrazu wejściowego i wyjściowego warstw konwolucyjnych, krawędzie obrazu wejściowego wypełniono zerami (ang. *zero padding*). Warstwy w pełni połączone oznaczone są jako `FC`, gdzie liczba oznacza liczbę neuronów w warstwie. Z powodu niskiej liczby klas, zdecydowano się zmniejszyć liczbę neuronów warstwach w pełni połączonych z 4096 (zaproponowana przez autorów sieci VGG) do 1024. Każda badana sieć neuronowa zakończona jest neuronem sigmoidalnym, którego wartość wskazuje diagnozę.

Zaplanowano szereg eksperymentów, mających na celu ocenę działania struktur sieci neuronowych oraz często stosowanych metod regularyzacji i metod treningu. Analizie poddano wpływ liczby warstw sieci na osiągane wyniki. Następnie, dokonano analizy skuteczności sieci wyposażonych w warstwy dropout oraz batch normalization, będącymi metodami regularyzacji stosowanymi w licznych architekturach sieci neuronowych. Zbadano również wpływ metody transfer learning, gdzie wykorzystano sieć neuronową VGG16, poddaną wcześniejszemu uczeniu na zadaniu klasyfikacji obiektów z bazy danych ImageNet. Wyniki otrzymane przy zastosowaniu techniki transfer learning porównano do sieci z losowo zainicjalizowanymi parametrami. Dodatkowo przeprowadzono analizę działania sieci z różnymi funkcjami aktywacji – ReLU oraz Leaky ReLU. Łącznie zaproponowano siedem eksperymentów oznaczonych literami od A do G, które przedstawiono w tabeli 4.4. W celu osiągnięcia wiarygodnych wyników zastosowano walidację krzyżową z pięcioma zbiorami. Ponadto, z uwagi na losowość wyników wynikających z inicjalizacji parametrów, metod rozszerzania danych i doboru przykładów uczących w ramach wsadów, każdy trening w ramach zbioru walidacji krzyżowej powtórzono sześć razy.

Metoda grupowania modeli jest często stosowana w celu poprawy skuteczności działania modelu, dlatego zbadano jej wpływ na osiągane rezultaty. Ze względu na ograniczenia obliczeniowe oraz po przeglądzie literatury, zdecydowano się każdą z badanych sieci wytrenować sześć razy. Przetestowano dwa sposoby realizacji grupowania modeli: poprzez uśrednienie wyjść sześciu wytrenowanych sieci oraz poprzez zastosowanie regresji logistycznej. Metoda poprzez uśrednianie wyjść jest prostsza i polega na obliczeniu średniej odpowiedzi generowanej przez zbiór sieci.

Tabela 4.4: Lista przeprowadzonych eksperymentów

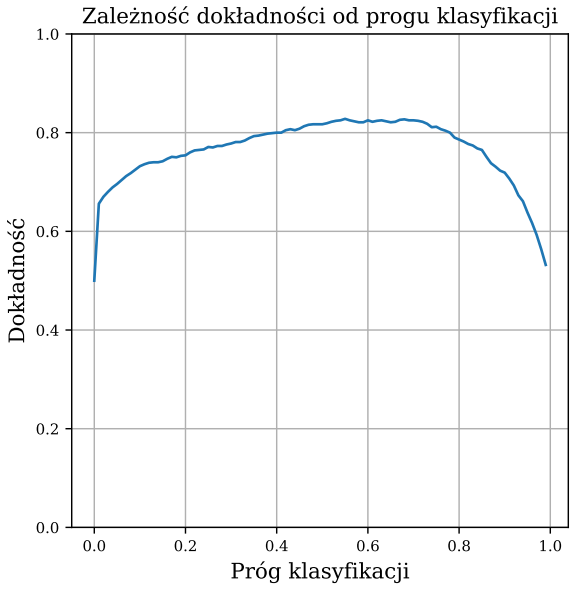
Eksperyment	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Architektura	VGG8	VGG8	VGG8	VGG8	VGG11	VGG16	VGG16
Funkcja aktywacji	ReLU	Leaky ReLu	ReLU	ReLU	ReLU	ReLU	ReLU
Batch normalization		✓	✓				
Dropout				✓	✓	✓	✓
Transfer learning							✓

Metoda regresji logistycznej jest bardziej złożona i polega na zastosowaniu wyjść sieci neuronowych jako wejść modelu regresji logistycznej. Model zawierał siedem parametrów: sześć parametrów odpowiadających sześciu wejściom oraz siódmy parametr będący progiem. Zaletą tego rozwiązania względem metody uśredniania jest to, że model może wykrywać relacje w wynikach osiąganych przez poszczególne sieci i np. przypisać mniejszą wagę sieci, które generuje błędne wyniki. Do treningu modelu regresji użyto 4000 zdjęć losowo wybranych ze zbioru treningowego.

Podczas testowania skuteczności sieci zaobserwowano, że dokładność klasyfikacji zależy od dobranego progu klasyfikacji, a domyślny próg o wartości 0,5 nie zapewnia najlepszych wyników, zwłaszcza przy dużych dysproporcjach liczby próbek w klasach danych treningowych. Celem dobrania optymalnego progu klasyfikacji, przeanalizowano zależność dokładności od progu klasyfikacji. Zależność tę przedstawiono na rys. 4.5, który został wygenerowany na podstawie analizy na zbiorze walidacyjnym w eksperymencie B. Jak jest to widoczne na rysunku, zastosowanie domyślnego progu wynoszącego 0,5 nie prowadzi do uzyskania najwyższej dokładności. W związku z tą obserwacją podjęto decyzję o zastosowaniu optymalizacji progu klasyfikacji, która została przeprowadzona metodą siatki (ang. *grid search*), wykorzystując zbiór walidacyjny. Jednakże, stosowanie tej metody powoduje znaczne fluktuacje wartości czułości i swoistości.

4.3.1 Przygotowanie danych do treningu i proces uczenia

Rozmiar zdjęć w bazie danych ISIC jest zróżnicowany, a zdjęcia mają różne proporcje szerokości i wysokości. Z uwagi na fakt, że znamiona zajmują tylko niewielką cześć obrazu, zdecydowano się zastosować maski do wykadrowania obrazu tak, aby badane znamiona zawierały większą część obrazu. Maski, które zawierały mniej niż 10 pikseli, uznano za nieprawidłowe i odrzucono.



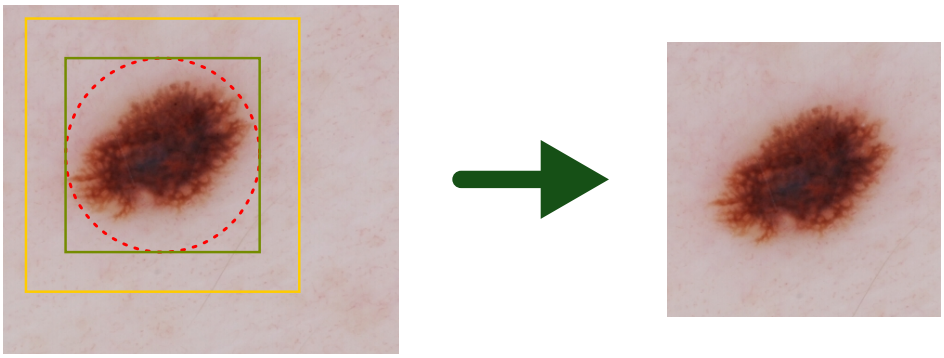
Rysunek 4.5: Zależność dokładności od wartości progu klasyfikacji

W celu wykadrowania znamion zaproponowano następującą procedurę. W pierwszej kolejności wyznaczono minimalny okrąg zawierający maskę znamienia, a następnie na okręgu opisano kwadrat o krawędziach równoległych do krawędzi obrazu. Aby zachować dodatkowy obszar wokół znamienia, rozmiar krawędzi kwadratu zwiększono o 44%. W ten sposób wyznaczony obszar został wycięty z oryginalnego obrazu. Operacja została zilustrowana na rys. 4.6

Następnie, rozmiar wyciętych fragmentów został zmieniony na 224×224 pikseli. Wybrany rozmiar jest kompromisem pomiędzy dokładnością klasyfikacji a wymaganiami obliczeniowymi sieci neuronowej. Ponadto taki rozmiar pozwala wykorzystać wstępnie przetrenowane sieci neuronowe, których rozmiar zazwyczaj wynosi 224×224 pikseli.

Zbiór danych został podzielony na trzy części: treningowy, walidacyjny oraz testowy. Zbiór treningowy zawierał 12333 znamiona łagodne oraz 884 znamiona złośliwe. Aby wyrównać liczbę obrazów w rozpatrywanych klasach, przykłady złośliwe zostały wielokrotnie skopiowane, czego wynikiem było 12376 znamion złośliwych. Zbiór walidacyjny i testowy zawierały po 200 zdjęć, po 100 zdjęć na klasę.

W celu zwiększenia dokładności klasyfikacji zastosowano metody rozszerzania danych, opierające się o transformacje geometryczne. Wykorzystano obroty w zakresie od 0 do 360 stopni, rozciągnięcia w pionie i poziomie w zakresie 0,1 rozmiaru obrazu,



Rysunek 4.6: Metoda wycinania znamienia z obrazu. Wyznaczenie minimalnego okręgu (czerwona, przerywana linia), opisanie kwadratu na okręgu (zielona linia) o krawędziach równoległych do krawędzi obrazu, powiększenie krawędzi kwadratu (żółta linia)

przybliżenia i oddalenia w zakresie od 90 do 110%, oraz odbicia w pionie i poziomie. Każda z wymienionych operacji miała prawdopodobieństwo zastosowania wynoszące 50%. Rozszerzanie danych było wykonywane w locie podczas uczenia.

W trakcie uczenia zastosowano algorytm SGD z momentem Nesterova o współczynniku momentum równym 0,9. Jako funkcję celu przyjęto binarną entropię krzyżową stosowaną w zadaniach klasyfikacji dwuklasowej. W wyniku analizy zmienności krzywej uczenia dokonano doboru początkowego współczynnika uczenia. W eksperymentach A, D, C oraz E przyjęto początkowy współczynnik uczenia równy 0,01. Eksperymenty B, F oraz G przeprowadzono z początkowym współczynnikiem uczenia wynoszącym 0,001. Przy wyższej wartości współczynnika uczenia proces uczenia nie był zbieżny, co objawiało się brakiem zmian wartości minimalizowanej funkcji celu.

W trakcie treningu zastosowano krokową redukcję współczynnika uczenia. Jeśli wartość funkcji celu mierzonej na zbiorze walidacyjnym nie zmniejszała się przez kolejne cztery epoki, to współczynnik uczenia dzielony był przez trzy. Dodatkowo zastosowano metodę wczesnego zatrzymania treningu, aby zapobiec nadmiernemu dopasowaniu. W przypadku braku spadku wartości funkcji celu na zbiorze walidacyjnym przez osiem kolejnych epok trening był zatrzymywany.

4.4. Wyniki

Przeprowadzono serię eksperymentów mających na celu ocenę wpływu różnych elementów struktury sieci neuronowej i metod uczenia na uzyskiwane wyniki. W tabeli 4.5 przedstawiono wyniki uzyskane przy zastosowaniu pojedynczej sieci, w tabeli 4.6 przedstawiono wyniki uzyskane w wyniku zastosowania metody grupowania modeli polegającej na wyznaczeniu średniej wyjść sześciu modeli, a w tabeli 4.7 przedstawiono wyniki uzyskane w wyniku zastosowania metody grupowania modeli z zastosowaniem modelu regresji logistycznej, którego wejściem były wyniki generowane przez sześć sieci. Otrzymane wyniki porównano pod względem rozmiaru sieci, czasu uczenia i liczbyepok (tabela 4.8).

Tabela 4.5: Wyniki – pojedyncza sieć. Rozwinięcia skrótów: LReLU – Leaky ReLU, BN – batch normalization, DO – dropout, TF – transfer learning

Eksperyment	**Dokładność**	**ROC AUC**	**Czułość**	**Swoistość**
A (VGG8 ReLU)	70,38	0,786	0,734	0,673
B (VGG8 LReLU BN)	72,12	0,808	0,754	0,688
C (VGG8 ReLU BN)	73,77	0,828	0,784	0,692
D (VGG8 ReLU DO)	72,87	0,804	0,775	0,683
E (VGG11 ReLU DO)	68,83	0,765	0,752	0,625
F (VGG16 ReLU DO)	67,78	0,749	0,687	0,668
G (VGG16 ReLU DO TF)	76,67	0,858	0,801	0,732

Tabela 4.6: Wyniki – zastosowanie grupowania modeli metodą uśredniania. Rozwinięcia skrótów: LReLU – Leaky ReLU, BN – batch normalization, DO – dropout, TF – transfer learning

Eksperyment	**Dokładność**	**ROC AUC**	**Czułość**	**Swoistość**
A (VGG8 ReLU)	72,50	0,802	0,842	0,608
B (VGG8 LReLU BN)	74,10	0,822	0,774	0,708
C (VGG8 ReLU BN)	76,80	0,851	0,796	0,740
D (VGG8 ReLU DO)	76,50	0,821	0,786	0,744
E (VGG11 ReLU DO)	71,20	0,779	0,816	0,608
F (VGG16 ReLU DO)	65,40	0,759	0,586	0,722
G (VGG16 ReLU DO TF)	79,40	0,882	0,868	0,720

Tabela 4.7: Wyniki – zastosowanie grupowania modeli poprzez regresję logistyczną. Rozwinięcia skrótów: LReLU – Leaky ReLU, BN – batch normalization, DO – dropout, TF – transfer learning

Eksperyment	**Dokładność**	**ROC AUC**	**Czułość**	**Swoistość**
A (VGG8 ReLU)	73,00	0,804	0,854	0,606
B (VGG8 LReLU BN)	73,60	0,823	0,738	0,734
C (VGG8 ReLU BN)	76,90	0,852	0,802	0,736
D (VGG8 ReLU DO)	75,50	0,821	0,822	0,688
E (VGG11 ReLU DO)	71,30	0,781	0,766	0,660
F (VGG16 ReLU DO)	66,20	0,758	0,628	0,696
G (VGG16 ReLU DO TF)	79,60	0,883	0,878	0,714

W przeprowadzonych badaniach dokonano oceny wpływu liczby warstw na dokładność klasyfikacji. W eksperymentach D, E oraz F, poddano testom sieci składające się z odpowiednio 8, 11 i 16 warstw. Najlepsze wyniki (dokładność – 72,87%) spośród trzech sieci uzyskała sieć składająca się z ośmiu warstw. Następnie, drugie miejsce zajęła sieć zawierająca 11 warstw (68,83%), a najgorszy wynik uzyskano dla sieci składającej się z 16 warstw (67,78%). Otrzymane wyniki pokazują istotę doboru odpowiedniej pojemności modelu do trudności zadania oraz posiadanego zbioru danych. W tym przypadku modele składające się z 11 i 16 warstw okazały się zbyt złożone dla danej ilości danych. Należy, również podkreślić, że wyniki są odmienne od tych raportowanych w [14], gdzie większa liczba warstw prowadziła do lepszych wyników uzyskiwanych w zadaniu klasyfikacji ImageNet.

Dokonano również analizy wpływu metody transfer learning na uzyskiwane wyniki. W tym celu przeprowadzono dwa eksperymenty z wykorzystaniem sieci VGG16, oznaczone jako eksperyment F oraz G. W eksperymencie F parametry sieci poddano losowej inicjalizacji, natomiast w eksperymencie G wykorzystano parametry uzyskane podczas treningu sieci na zbiorze danych ImageNet. Zastosowanie metody transfer learning spowodowało wyraźny wzrost dokładności o 10,24 punktów procentowych. Ponadto uzyskany wynik dla eksperymentu G był najlepszy ze wszystkich przeprowadzonych eksperymentów, natomiast sieć poddana losowej inicjalizacji w eksperymencie F osiągnęła wynik najgorszy. Otrzymane wyniki wykazują, że wybór początkowych parametrów sieci neuronowej ma istotny wpływ na proces uczenia sieci neuronowej oraz otrzymane wyniki.

W kolejnych eksperymentach, analizie poddano wpływ dwóch popularnych funkcji aktywacji. Porównano dwie sieci VGG8 wyposażone w warstwy batch normalization oraz funkcje aktywacji Leaky ReLU (eksperyment B) i ReLU (eksperyment D). Zastosowanie funkcji aktywacji ReLU umożliwiło uzyskanie dokładności lepszej o około 1,5 punktu procentowego w porównaniu do sieci wyposażonej w funkcję Leaky ReLU. Pomimo korzystnych właściwości wykazanych teoretycznie, sieć wyposażona w funkcję Leaky Relu nie uzyskała lepszych wyników. Różnica jest jeszcze bardziej widoczna w przypadku zastosowania grupowania modeli. Wnioskiem z przeprowadzonych eksperymentów jest, że dobór odpowiednich hiperparametrów do konkretnego zadania jest istotny, ponieważ metody działające w jednym zastosowaniu niekoniecznie prowadzą do dobrych wyników w innych.

Oprócz wpływu hiperparametrów analizowano również wpływ dwóch popularnych metod regularyzacji – batch normalization oraz dropout. Zastosowanie metody batch normalization (eksperyment C) pozwoliło uzyskać znaczne lepsze wyniki niż w przypadku sieci wyposażonych w warstwę dropout (eksperyment D). Sieć bez żadnej techniki regularyzacji (eksperyment A) osiągnęła najgorsze wyniki spośród wszystkich testowanych rozwiązań. Różnica jest również widoczna w wynikach przy zastosowaniu techniki grupowania modeli. Wyposażenie sieci neuronowej w obie te metody prowadzi do bardzo niskich wyników. Jest to zgodnie z wynikami teoretycznymi raportowanymi przez innych badaczy [56].

Zbadano również wpływ opisanych metod grupowania modeli (poprzez uśrednianie i zastosowanie regresji logistycznej) na wyniki klasyfikacji. W każdej z badanych struktur grupowanie, modeli umożliwiło poprawę wyników, jednak różnica pomiędzy wynikami uzyskanymi w wyniku zastosowania różnych metod jest nieznaczna. Metoda grupowania przez uśrednianie umożliwia uzyskanie podobnych wyników do metody grupowanie przez zastosowanie regresji liniowej, dodatkowo zaletą tej metody jest prostota i brak konieczności przeprowadzania uczenia, jak ma to miejsce w przypadku metody opartej o regresję liniową.

W tabeli 4.8 przedstawiono szczegóły dotyczące rozmiarów sieci oraz czasów uczenia. Czas uczenia został obliczony jako średni czas uczenia na pięciu podzbiorach wykorzystanych w metodzie k-krotnej walidacji. Najdłuższy czas uczenia wyniósł 5,2 h dla sieci VGG16 bez zastosowania metody transfer learning. Metoda transfer learning umożliwiła ponad dwukrotne skrócenie czasu uczenia z 5,2 h do 2,4 h. Liczba warstw ma wyraźny wpływ na czas uczenia, sieć 8-warstwowa uczona była średnio

2,1 h, 11-warstwowa przez 3,8 h a 16-warstwowa przez 5,2 h. Trening sieci bez zastosowania mechanizmów regularyzacji (eksperyment A) był najszybszy (1,5 h), ale prowadził do najgorszych wyników. Zastosowanie metody dropout wydłużyło czasu uczenia do 2,1 h a metody batch normalization do 3,5 h. Zastosowana funkcja aktywacji ma pomijalny wpływ na czas uczenia, wyniósł on 3,6 h dla funkcji Leaky ReLU oraz 3,5 h dla ReLU.

Tabela 4.8: Rozmiar sieci a czas uczenia. Wyniki dotyczą pojedynczego modelu. Rozwinięcia skrótów: LReLU – Leaky ReLU, BN – batch normalization, DO – dropout, TF – transfer learning

Eksperyment	**Rozmiar sieci (MB)**	**Liczba parametrów**	**Liczba epok**	**Czas uczenia (h)**
A (VGG8 ReLU)	240	30659587	17,8	1,5
B (VGG8 LReLU BN)	240	30659587	31,6	3,6
C (VGG8 ReLU BN)	240	30659587	29,8	3,5
D (VGG8 ReLU DO)	240	30659587	24,6	2,1
E (VGG11 ReLU DO)	281	35971843	20,4	3,8
F (VGG16 ReLU DO)	320	41456449	30,0	5,2
G (VGG16 ReLU DO TF)	320	41456449	13,4	2,4

4.5. Podsumowanie

W niniejszym rozdziale przedstawiono podejście do projektowania rozwiązań małej skali tj. takich, w których zbiór danych jest niewielki w porównaniu do zbiorów stosowanych w zadaniach typu benchmark. Przykładem takiego zadania jest klasyfikacja znamion skórnych na przypadki złośliwe i łagodne, na którym testowano analizowane podejścia.

Podjęte badania ukazują wyzwania, z jakimi należy się mierzyć w wielu praktycznych zastosowaniach głębokich sieci neuronowych. Badania obejmowały eksperymenty, w których analizowano wpływ struktury sieci neuronowej oraz stosowanych metod na otrzymywane wyniki. Eksperymenty wykazały znaczne różnice w wynikach w zależności od zastosowanych struktur i metod uczenia.

Najbardziej widoczny wzrost skuteczności wystąpił w wyniku zastosowania metody transfer learning. W przypadku sieci VGG16, zastosowanie tej metody

pozwoliło na wzrost dokładności o prawie 9 punktów procentowych, a wartości ROC AUC o 0,1. Wykorzystanie metody transfer learning przyczynia się do znacznej poprawy wyników klasyfikacji, zwłaszcza w przypadku zadań, gdzie dostępny zbiór danych jest niewielki. Dodatkowo zastosowanie metody transfer learning skraca czasu uczenia. Wyniki przeprowadzonych badań wykazują istotę wstępnego przeuczenia sieci przed treningiem na zadaniu docelowym. Badania nad metodami wstępnego uczenia sieci neuronowych zostaną opisane w rozdziale 6.

Kolejnym istotnym czynnikiem, mającym wpływ na wyniki jest liczba warstw. W eksperymentach wykazano, że w przypadku braku zastosowania metody transfer learning, najlepsze wyniki uzyskano przy użyciu najmniejszej sieci VGG8, a najgorsze wyniki przy użyciu sieci VGG16. Różnica dokładności między tymi sieciami wynosiła 5 punktów procentowych, a różnica ROC AUC – 0,05. Przy niewielkim rozmiarze zbioru uczącego, zalecane jest rozpoczęcie eksperymentów od sieci o niewielkim rozmiarze. Przyczyną takich wyników jest fakt, że rozmiar zbioru uczącego jest niewystarczający do skutecznego dobrania parametrów większych sieci.

W eksperymentach wykazano, że metody batch normalization oraz dropout powodują poprawę wyników. Niemniej jednak różnica w wynikach sieci, w których zastosowano batch normalization lub dropout nie jest aż tak znacząca. Zgodnie z wynikami raportowanymi w literaturze, jednoczesne zastosowanie obu metod prowadzi do pogorszenia wyników klasyfikacji. Zastosowanie funkcji aktywacji ReLU prowadzi do minimalnie lepszych wyników niż funkcji Leaky ReLU, co stoi w sprzeczności z niektórymi wynikami z literatury, gdzie metoda Leaky ReLU wykazuje się lepszą skutecznością. Wyniki te pokazują, że nie ma uniwersalnych reguł projektowania rozwiązań opartych o głębokie uczenie, a indywidualne podejście do każdego problemu jest kluczowe. Dobór struktury oraz hiperparametrów jest procesem czasochłonnym i wymagającym obliczeniowo, co motywuje do podjęcia dalszych badań nad automatycznymi metodami doboru struktury i hiperparametrów, co opisano w kolejnym rozdziale.

5. Automatyczny dobór struktury sieci neuronowych

5.1. Wprowadzenie

W poprzednim rozdziale wykazano, że architektura sieci neuronowej jest kluczowym czynnikiem wpływającym na osiągane wyniki. Dodatkowo architektura ma również wpływ na wymagania obliczeniowe, czas uczenia i generowania odpowiedzi. W niniejszym rozdziale przedstawiona zostanie propozycja algorytmu doboru struktury, oparta na wnioskach wyciągniętych z badań opisanych w poprzednim rozdziale oraz przeglądzie literatury.

Ręczny dobór struktury sieci neuronowej jest procesem czasochłonnym, składającym się z prób i błędów, oraz wymagającym wiedzy i doświadczenia. Dobór struktury klasycznej sieci neuronowej jest prostszy niż w przypadku głębokich sieci neuronowych z powodu mniejszej przestrzeni poszukiwań, która wynika z mniejszego rozmiaru sieci i mniejszej liczby hiperparametrów opisujących sieć. Co więcej, czas treningu takich sieci najczęściej był liczony w minutach, podczas gdy trening głębokich sieci neuronowych liczony jest w godzinach, a nawet dniach. Pomimo tego, od lat rozwijały się algorytmy doboru struktury klasycznych sieci neuronowych, z których jednym z najbardziej znanych jest algorytm NEAT [74]. Znaczny rozmiar obecnie stosowanych sieci neuronowych znacznie utrudnia dobór skutecznej i wydajnej architektury, co jest spowodowane dużą liczbą hiperparametrów opisujących te sieci. Pojawienie się nowych metod dodatkowo komplikuje to zadanie, poprzez zwiększenie liczby możliwych opcji wyboru.

Jednym z rozwiązań wyboru architektury jest wykorzystywanie gotowych struktur, opracowanych i przetestowanych przez inne grupy badawcze. Przykładami

szeroko stosowanych architektur w różnych zadaniach przetwarzania obrazu są ResNet [13], VGG [14], DenseNet [45], czy EfficientNet [46]. Struktury te zostały przystosowane do zadań dużej skali, takich jak zadanie klasyfikacji ImageNet, jednak wiele praktycznych zadań przetwarzania obrazu nie wymaga analizy tak dużej liczby klas, a dostępne zbiory danych są zazwyczaj znacznie mniejsze niż zbiór ImageNet, który składa się z około miliona obrazów. Ponadto, jak zauważono w badaniach opisanych w rozdziale 2, struktura sieci neuronowej ma bezpośredni wpływ na pojemność modelu, a ta powinna być przystosowana do trudności rozwiązywanego zadania oraz do rozmiaru zbioru danych. Zbyt duże sieci nie tylko mogą utrudnić proces uczenia (co wykazano w poprzednim rozdziale), ale również zwiększają wymagania obliczeniowe, pogarszają generalizację oraz wydłużają czas uczenia i generowania odpowiedzi.

Innym rozwiązaniem jest zastosowanie metod automatycznego doboru struktury (ang. *Neural Architecture Search* – NAS). Automatyczny dobór struktury jest dziedziną zaliczaną do jeszcze szerszej dyscypliny zwanej automatycznym uczeniem maszynowym, w literaturze angielskojęzycznej oznaczanej skrótem AutoML. Rozwiązania z obszaru AutoML pozwalają na automatyczny dobór struktury, a także innych hiperparametrów takich jak optymalizator, współczynnik uczenia, metody regularyzacji i augmentacji danych. Przykładowo, w pracy [75] algorytm dobierał prawdopodobieństwo wykonywania poszczególnych operacji rozszerzania danych oraz ich intensywność. Metody z rodziny AutoML minimalizują potrzebę ręcznego doboru wielu hiperparametrów.

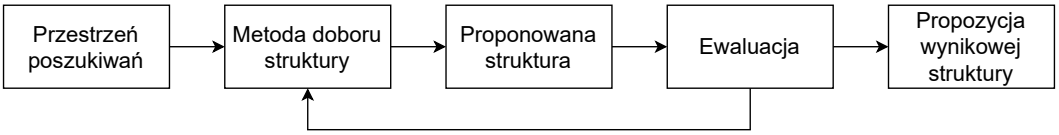
Celem algorytmów automatycznego doboru struktury jest optymalizacja struktury sieci neuronowej pod kątem wybranych kryteriów, takich jak dokładność, liczba parametrów czy czas generowania odpowiedzi. Pożądaną cechą algorytmu doboru struktury jest krótki czas poszukiwania i niskie wymagania obliczeniowe. Wczesne rozwiązania stosowane w doborze struktury głębokich sieci neuronowych wymagały dużych zasobów obliczeniowych liczonych w setkach jednostek GPU, co utrudniało praktyczne zastosowanie metod przez indywidualnych badaczy, grupy naukowe i niewielkie firmy.

W ramach doktoratu skupiono się na problemie doboru struktury sieci neuronowej. Podobnie jak w przypadku wielu rozwiązań raportowanych w literaturze, algorytmy doboru struktury opracowywane są zazwyczaj dla zadań dużej skali, gdzie dostępne są duże zbiory danych i znaczące zasoby sprzętowe. Celem prac w ramach doktoratu

było opracowanie metod dla problemów małej skali, z niewielką ilością danych. Wyniki badań opublikowano w [25], [26].

5.1.1 Przestrzeń poszukiwań, metoda doboru oraz metoda ewaluacji

Rozwiązania doboru struktury sieci neuronowych można scharakteryzować w trzech kategoriach: przestrzeń poszukiwań, metoda doboru oraz metoda ewaluacji (rys. 5.1).



Rysunek 5.1: Schemat działania systemu doboru struktury

Przestrzeń poszukiwań to zbiór możliwych architektur sieci neuronowych, które mogą zostać wygenerowane przez algorytm doboru struktury. Dobór przestrzeni poszukiwań ma wpływ na czas działania algorytmu i skuteczność wynikowej struktury. Często stosowaną praktyką jest definiowanie przestrzeni poszukiwań na podstawie sprawdzonych rozwiązań opracowanych ręcznie np. funkcja ReLU stosowana jest powszechnie jako funkcja aktywacji. Jednym z potencjalnych problemów związanych z definiowaniem przestrzeni poszukiwań jest ryzyko pojawienie się tendencji algorytmów do generowania architektur zbliżonych do tych, które zostały opracowane ręcznie, zamiast wprowadzać innowacyjne rozwiązania.

Przestrzeń poszukiwań może być scharakteryzowana według różnych kryteriów. Jednym z nich jest dozwolony schemat połączeń w sieci neuronowej. Można wyróżnić dwa sposoby połączeń: połączenie łańcuchowe oraz połączenia gałęziowe. Struktura łańcuchowa jest prostym sposobem połączeń gdzie, każda warstwa odbiera sygnał tylko od warstwy bezpośrednio ją poprzedzającej i przesyła wynik działania do warstwy kolejnej. Przestrzeń poszukiwań w tym przypadku jest ograniczona i obejmuje m.in. liczbę warstw, rodzaj warstw oraz hiperparametry opisujące warstwy. Z kolei, przestrzeń poszukiwań umożliwiająca generowanie sieci z połączeniami gałęziowymi jest znacznie bardziej złożona, ponieważ daje wiele możliwości budowania sieci i różnych schematów połączeń.

Innym sposobem definiowania przestrzeni poszukiwań jest podział na przestrzeń poszukiwań składającą się ze struktury całych sieci oraz przestrzeń poszukiwań składające się z modułów (ang. *cell, block*). W przypadku modułowej przestrzeni

poszukiwań najpierw dobierana jest struktura modułu, z którego następnie tworzona jest sieć, podobnie jak w sieci GoogleLeNet. Choć poszukiwania modułu jest znacznie szybsze niż dobór całej struktury, to prowadzi to do powstania kolejnego problemu optymalizacji, w którym należy dobrać liczbę modułów, w celu osiągnięcia optymalnych wyników pod kątem wybranych kryteriów.

Kolejnym sposobem kategoryzowania przestrzeni poszukiwań jest podział na przestrzeń poszukiwań dyskretną oraz ciągłą. Struktura sieci neuronowej oraz jej hiperparametry sprawiają, że naturalną formą przestrzeni poszukiwań jest forma dyskretna. Istnieją jednak metody relaksacji, które pozwalają przekształcenie przestrzeni poszukiwań z dyskretnej na ciągłą, umożliwiając zastosowanie gradientowych metod optymalizacji [20].

Drugim kryterium charakteryzującym algorytm doboru struktury jest metoda doboru. Najpopularniejsze metody obejmują metody ewolucyjne, uczenie przez wzmocnienie oraz zdobywające ostatnio coraz większą popularność metody gradientowe. Oprócz tego w doborze struktury stosuje się metody oparte o optymalizację Bayesowską [76], jednak rozwiązania te nie znalazły szerokiego zastosowania ze względu na niską efektywność w problemach wielowymiarowych. Metody Bayesowskie sprawdzają się jednak w doborze hiperparametrów, szczególnie gdy ich liczba nie jest duża. Inną kategorią są metody losowe, które najczęściej służą jako punkt odniesienia w ocenie innych rozwiązań. Metody te zostaną opisane bardziej szczegółowo w kolejnej sekcji.

Ostatnim kryterium jest sposób ewaluacji rozwiązań proponowanych przez algorytm. Informacja o skuteczności działania danej architektury wykorzystywana jest do kierowania procesem poszukiwania. Pełny trening każdej proponowanej architektury, a następnie ocena na zbiorze walidacyjnym, jest najprostszym, ale bardzo czasochłonnym rozwiązaniem. Aby przyspieszyć proces ewaluacji, stosuje się różne techniki takie jak niepełny trening, trening na niepełnym zbiorze danych, czy trening z wykorzystaniem obrazów o mniejszym rozmiarze [77]–[79]. Innym rozwiązaniem jest zastosowanie metod predykcji krzywej uczenia i przerywanie treningów tych modeli, których prognozowana dokładność jest niska [80], [81]. Kolejnym sposobem przyspieszenia ewaluacji jest wykorzystywanie parametrów z sieci generowanych w poprzednich etapach. Specjalnym przypadkiem rozwiązań są tzw. metody zachowujące funkcje, które umożliwiają rozbudowę struktury bez utraty zdobytej wiedzy [82], [83].

5.2. Przegląd literatury

5.2.1 Metody oparte o podejście ewolucyjne

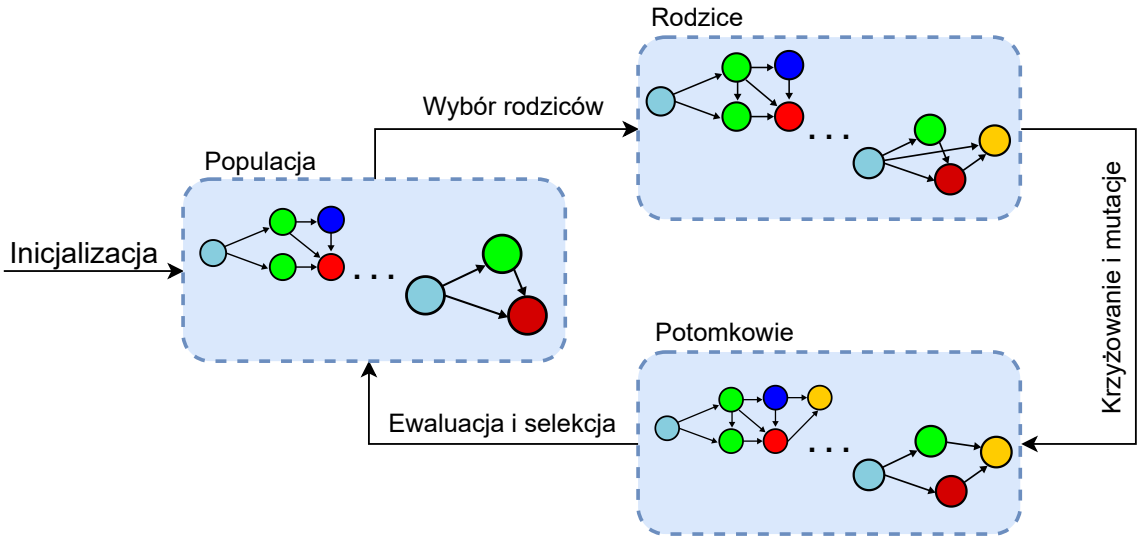
Metody ewolucyjne od dawna znajdują zastosowanie w dziedzinie automatycznego doboru struktury sieci neuronowej [74], [84]. Rozwiązania, w których metody ewolucyjne są wykorzystywane do doboru struktury określane są neuroewolucją (ang. *neuro-evolution*). Początkowo, metody te były stosowane do niewielkich struktur sieci neuronowych składających się z kilku warstw w pełni połączonych. Wraz z postępem w dziedzinie głębokiego uczenia, zastosowanie algorytmów ewolucyjnych do doboru ich struktury stwarza nowe wyzwania związane z wysokimi wymaganiami obliczeniowymi tych algorytmów, co przekłada się na wydłużony czas działania.

Algorytmy ewolucyjne są inspirowane procesem doboru naturalnego, który zachodzi wśród organizmów żywych. Działanie algorytmu rozpoczyna się od utworzenia populacji, składającej się z określonej liczby organizmów. Każdy z organizmów reprezentuje rozwiązanie pewnego problemu – w przypadku doboru struktury, organizm reprezentuje określoną strukturę sieci neuronowej. Sieci neuronowe, reprezentowane przez poszczególne organizmy, poddawane są treningowi, a następnie oceniane są pod kątem wybranej miary. Wynik ewaluacji jest wykorzystywany do doboru organizmów tworzących nową populację.

W trakcie działania algorytmu organizmy poddawane są modyfikacjom, nazywanymi mutacjami. W przypadku doboru struktury sieci neuronowej, mutacja może polegać na zmianie rodzaju warstwy lub modyfikacji hiperparametrów warstwy. Kolejną operacją jest krzyżowanie, które polega na tworzeniu nowego organizmu na podstawie dwóch dobranych rodziców. Dobór organizmów do krzyżowania dokonywany jest zgodnie z określoną metodą, najczęściej najlepsze organizmy mają większe szanse na generowanie potomków. Mutacja jest częściej stosowaną operacją w obszarze doboru struktury, ze względu na łatwiejszą implementację w porównaniu do operacji krzyżowania.

Cykl, który obejmuje powstanie nowej populacji, ewaluację organizmów, zastosowanie mutacji i krzyżowania, i generację kolejnej populacji nazywany jest iteracją, epoką lub generacją. W celu uniknięcia niejednoznaczności, w dalszej części rozdziału przyjęto, że termin iteracja odnosi się do algorytmu doboru struktury, natomiast termin epoka odnosi się do procesu uczenia sieci neuronowej. Algorytm

kończy swoje działanie, gdy spełnione zostaną określone warunki, takie jak osiągnięcie określonej liczby iteracji czy brak poprawy rozwiązania. Schemat działanie algorytmu przedstawiony jest na rys. 5.2



Rysunek 5.2: Algorytmy ewolucyjne w doborze struktury

Zastosowanie algorytmów ewolucyjnych do doboru struktury sieci neuronowych stanowi wyzwanie ze względu na kilka czynników m.in.: konieczność opracowania sposobu kodowania struktury sieci neuronowej, duże zapotrzebowanie obliczeniowe (każdy z organizmów w populacji musi zostać poddany treningowi), konieczność implementacji operacji mutacji i krzyżowania. Metody opisywane w literaturze różnią się pod względem sposobu doboru rodziców, realizacji mutacji i krzyżowania.

Ewolucja parametrów sieci neuronowej jest kosztownym procesem, dlatego stosowana była tylko w przypadku niewielkich sieci neuronowych [74]. Obecnie, metody ewolucyjne stosowane są tylko do doboru struktury, natomiast dobór parametrów dokonywany jest przy wykorzystaniu algorytmu propagacji wstecznej i algorytmów optymalizacji gradientowej.

Jednym z bardziej znanych przykładów zastosowania metod ewolucyjnych w doborze struktury głębokiej sieci neuronowej jest praca „Large-Scale Evolution of Image Classifiers" [85]. Rozwiązanie to charakteryzuje się dużą przestrzenią poszukiwań z niewielkimi ograniczeniami. Algorytm doboru struktury generuje grafy reprezentujące struktury sieci neuronowych. Jedną ze zmian w stosunku do poprzednich rozwiązań jest wprowadzenie warstwy jako głównej jednostki budującej

sieć zamiast neuronu. W metodzie zastosowano mechanizm mutacji, który polega na dodawaniu lub usuwaniu warstwy, a także na zmianie hiperparametrów istniejących warstw. Parametry proponowanych sieci dobierane są przez algorytm optymalizacji gradientowej, którego współczynnik uczenia również dobierany jest przez algorytm ewolucyjny.

Dobór organizmów przechodzących do kolejnej iteracji realizowany jest metodą turniejową, polegającą na losowym doborze pary organizmów. Lepszy organizm z pary przechodzi do kolejnej iteracji, a gorszy jest odrzucany. Dzięki doborowi turniejowemu możliwe jest równoległe prowadzenie obliczeń, ponieważ nie jest wymagane oczekiwania do końca treningu wszystkich struktur w celu zakończenia iteracji. Mutacje wybierane są z predefiniowanego zbioru operacji, który obejmuje m.in.: dodanie i usunięcie warstwy konwolucyjnej, dodanie lub usunięcie połączenia skracającego (ang. *shortcut connection*, zmianę kroku filtru (ang. *stride*), zmianę liczby filtrów w losowo dobranej warstwie. W algorytmie nie ustalono górnego limitu wartości hiperparametrów takich jak liczba warstw czy liczba filtrów w warstwie. Ewolucja przebiega w sposób przyrostowy, początkowa populacja zawiera struktury składające się z zaledwie jednej warstwy konwolucyjnej o różnych hiperparametrach, wraz z działaniem algorytmu wprowadza się kolejne warstwy. W celu skrócenia czasu działania algorytmu, w warstwach, w których jest to możliwe (np. w warstwach, które nie uległy mutacji) zastosowano dziedziczenie parametrów przez potomków. Proces doboru struktury został przeprowadzony z wykorzystaniem 250 jednostek GPU, rozmiar populacji wynosił 1000, a dobór ostatecznego modelu trwał 250 godzin dla zbioru CIFAR-10 [86]. Z tego wynika, że obliczenia prowadzone na jednej karcie trwałyby około siedmiu lat.

W artykule [77] zaprezentowano metody regularyzacji usprawniające proces doboru struktury. W ramach metody wykorzystano dobór turniejowy (ang. *tournament selection*), który znajduje zastosowanie w innych metodach doboru struktury. Wprowadzoną innowacją jest uzależnienie doboru osobników od ich wieku, mierzonego w iteracjach od powstania organizmu. Algorytm dobierając osobniki do mutacji oraz kolejnej iteracji preferował młodsze organizmy. Wprowadzona zmiana przyczyniała się do lepszej eksploracji przestrzeni poszukiwań oraz zapobiegła zbyt wczesnemu zawężeniu poszukiwań do określonych rozwiązań, co zwiększyło różnorodność proponowanych architektur. Dodatkowo wprowadzony mechanizm odrzuca z populacji osobniki, których wysoka skuteczność wynikała z przypadku. Algorytm dokonuje doboru struktury dwóch modułów: normalnego oraz

redukującego. Moduł normalny dokonuje transformacji sygnału wejściowego bez zmiany jego rozmiaru, natomiast moduł redukujący zmniejsza rozmiar sygnału wejściowego. Liczbę modułów w wynikowej strukturze sieci neuronowej oraz liczbę filtrów w warstwach dobiera się ręcznie. W algorytmie wykorzystuję się jedynie operację mutacji, pomijając krzyżowanie. Całkowity czas poszukiwania struktury wynosi 10000 godzin na jednej karcie graficznej (czyli ponad rok). Wynikiem działania algorytmu jest znana architektura AmoebaNet-A.

Algorytmy doboru struktury oparte o metody ewolucyjne znalazły wiele zastosowań w obszarze głębokiego uczenia. Metody ewolucyjne zostały zastosowane do doboru architektury sieci FCN (ang. *Fully Concolutional Network*) [87] oraz hiperparametrów uczenia m.in. współczynnika uczenia. Celem algorytmu była maksymalizacja miary określającej jakość segmentacji oraz minimalizacja liczby parametrów. Zaproponowane rozwiązanie zostało wykorzystane w segmentacji prostaty oraz detekcji chorób serca na obrazach rezonansu magnetycznego. Innym przykładem zastosowania metod ewolucyjnej jest opracowanie struktury sieci neuronowej do segmentacji żył dna oka [88]. Algorytm, optymalizując miarę F1, dobierał strukturę kilku modułów, z których tworzono wynikową sieć neuronową.

5.2.2 Metody oparte o uczenie przez wzmocnienie

Kolejną rodziną metod doboru struktury sieci neuronowej są techniki oparte o uczenie przez wzmocnienie, które polega na uczeniu agenta realizacji zadania w danym środowisku. Metoda ta przypomina proces prób i błędów, gdzie agent na podstawie nagród i kar uczy się skutecznie działać w danym środowisku.

Zadaniem agenta w obszarze doboru struktury jest proponowanie struktur sieci neuronowych, a maksymalizowaną nagrodą jest dokładność osiągana przez proponowane sieci. Znaną pracą w tym obszarze jest [89], w której zaproponowano system oparty o rekurencyjną sieć neuronową, której uczenie odbywało się metodą uczenia przez wzmocnienie. Sieć ta składa się z dwóch warstw zawierających 34 jednostki LSTM każda [90]. Zadaniem rekurencyjnej sieci było generowanie hiperparametrów kolejnych warstw, takich jak rozmiar filtru, krok filtru czy liczba filtrów w warstwie. Wartości hiperparametrów były generowane w sposób sekwencyjny, a wygenerowane hiperparametry trafiały z powrotem na wejście sieci rekurencyjnej. Działanie algorytmu zostaje zatrzymane po osiągnięciu założonej liczby warstw. Po wygenerowaniu struktury sieci poddawana jest ona uczeniu. Parametry rekurencyjnej sieci neuronowej dobierane są w celu maksymalizacji

dokładności uzyskiwanych przez generowane sieci. Dokładność jest sygnałem nagrody, ze wzwględu na brak możliwości wyznaczenia gradientu, w celu uczenia stosuje się metodę REINFORCE [91]. Do doboru struktury zastosowano 800 kart graficznych, autorzy nie podali jednak czasu działania algorytmu.

Kolejną znaną pracą, w której wykorzystano uczenie przez wzmocnienie, jest [82]. W rozwiązaniu tym wykorzystuje się transformacje zachowujące funkcję, które zostaną szerzej opisane w dalszej części pracy. Dzięki tym operacjom można dodawać nowe komponenty do struktury sieci neuronowej bez utraty już zdobytej wiedzy. Zastosowanie tych operacji umożliwiło znaczącą redukcję wymagań obliczeniowych. Obliczenia były prowadzone z wykorzystaniem pięciu kartach graficznych, co jest niewielką liczbą w porównaniu do wcześniejszych rozwiązań, w których stosowano setki kart graficznych. W przeciwieństwie do tych rozwiązań, gdzie każde nowopowstałe sieci uczone były od podstaw, w tym rozwiązaniu wykorzystuje się struktury z poprzednich iteracji, które są poddawane modyfikacjom w każdej kolejnej iteracji algorytmu. Za generowanie struktury sieci neuronowej odpowiada rekurencyjna sieć neuronowa, której parametry dobierane są metodą REINFORCE [91]. Ponieważ decyzje o kolejnych elementach struktury są podejmowane sekwencyjnie, przestrzeń poszukiwań jest przestrzenią warunkową. Koder sieci neuronowej, realizowany w postaci dwukierunkowej sieci LSTM, uczy się generowania reprezentacji aktualnej architektury. Moduł wykonawczy korzysta z reprezentacji i decyduje o zastosowaniu modyfikacji, a także określa parametry tych modyfikacji (np. liczbę filtrów dodawanej warstwy).

Innym podejściem, w którym wykorzystano uczenie przez wzmocnienie i algorytm REINFORCE jest praca [92], w której algorytm dokonuje doboru struktury modułu. Moduł może składać się z osiem typów operacji, obejmujących: różne rodzaje warstw konwolucyjnych o różnych rozmiarach filtrów, metody max i average pooling oraz operację tożsamościową. Sygnały wyjściowe warstw w module mogą być ze sobą łączone poprzez blok dodawania lub konkatenacji. Dobór architektury modułu przebiega od modułów najprostszych do najbardziej złożonych. W celu przyspieszenia ewaluacji modułów wprowadzono dodatkowy model, który dokonywał ich oceny bez konieczności przeprowadzania treningu. Model dobierający strukturę realizowany był w postaci rekurencyjnej sieci neuronowej opartej o bloki LSTM. W celu zwiększenia skuteczności generowanych modułów zastosowano technikę grupowania modeli złożoną z pięciu sieci rekurencyjnych.

Metody uczenia przez wzmocnienie znalazły zastosowanie w różnych aplikacjach. W pracy [93] autorzy zastosowali metody uczenia przez wzmocnienie do doboru hiperparametrów każdej warstwy sieci neuronowej. Sieć została wyposażona w funkcję aktywacji Swish oraz miała gęsty system połączeń, podobny do sieci DenseNet. Kolejnym przykładem zastosowania tych metod jest praca [94], gdzie kontroler poszukiwał struktury trzech komponentów, z których budowana była sieć do zadań segmentacji. Znaną architekturą, która powstała w wyniku wykorzystania uczenia przez wzmocnienie jest rodzina sieci EfficientNet [46]. Moduł sieci został dobrany za pomocą rekurencyjnej sieci neuronowej, która uczona była z zastosowaniem uczenia przez wzmocnienie. Jako cel obrano maksymalizację dokładności i minimalizację liczby obliczeń.

5.2.3 Rozwiązania oparte o metody gradientowe

Gradientowe metody optymalizacji są powszechnie stosowane w doborze parametrów głębokich sieci neuronowych. Ich zastosowanie jest możliwe dzięki temu, że modele oraz funkcje celu są różniczkowalne, co pozwala na wyznaczenie gradientu względem parametrów sieci. Jednak dyskretna przestrzeń poszukiwań struktur sieci neuronowych uniemożliwia bezpośrednie zastosowanie tych metod do doboru struktury. Aby tego dokonać, konieczne jest przekształcenie przestrzeni poszukiwań z dyskretnej w ciągłą. W ostatnich latach dokonał się znaczący postęp w obszarze gradientowych metodach doboru struktury. Ich zaletą jest duża efektywność – dobór struktury w zadaniu klasyfikacji ImageNet zajmuje zaledwie kilka dni przy wykorzystaniu jednej karty graficznej.

Gradientowe metody doboru struktury opierają się na relaksacji, która przekształca przestrzeń poszukiwań z dyskretnej na ciągłą. Metoda relaksacji wprowadza dodatkowe, dobieralne w trakcie uczenia parametry odpowiadające za kształt struktury. Dzięki temu, że można wyznaczyć gradient funkcji celu względem tych parametrów, możliwe jest zastosowanie gradientowych metod optymalizacji do doboru struktury.

Przełomowym rozwiązaniem w tym obszarze jest metoda DARTS (ang. *Differentiable ARchiTecture Search*) [20], która stała się podstawą dla innych rozwiązań. Metoda charakteryzuje się uniwersalnością i może być zastosowana do doboru struktury różnego typu sieci neuronowych, takich jak rekurencyjne czy konwolucyjne.

Metoda DARTS służy do doboru architektury modułu, który będzie wykorzystany w tworzeniu finalnej sieci. Każdy moduł w strukturze ma dwa wejścia i jedno wyjście. Struktura modułu może być interpretowana jako skierowany, acykliczny graf składający się z N wierzchołków, z których każdy odpowiada reprezentacji $x^{(i)}$ wewnątrz struktury modułu. Każda krawędź grafu (i, j) reprezentuje pewną transformację $o^{(i,j)}$, która przetwarza reprezentację $x^{(i)}$. W procesie optymalizacji dobierane są operacje, które realizowane są przez krawędzie grafu. Zbiór możliwych operacji jest ustalany przed rozpoczęciem działania algorytmu.

Aby przekształcić dyskretny problem optymalizacji w problem ciągły, stosuje się relaksację. Relaksacja polega na zastosowaniu ważonej sumy operacji $o \in \mathcal{O}$ zgodnie ze wzorem:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x) = \sum_{o \in \mathcal{O}} p_o^{(i,j)} o(x) \tag{5.1}$$

$$\sum_{o \in \mathcal{O}} p_o^{(i,j)} = 1 \tag{5.2}$$

gdzie:

- x – wejście,
- $\bar{o}^{(i,j)}(x)$ – wyjście krawędzi (i, j),
- $\mathcal{O}$ – zbiór operacji,
- $\alpha^{(i,j)}$ – wektor parametrów, dla krawędzi (i, j) o rozmiarze $|\mathcal{O}|$
- $p^{(i,j)}$ – wektor wag dla krawędzi (i, j)

Reprezentacja $x^{(j)}$ wierzchołka j wyznaczana jest poprzez sumę dochodzących krawędzi:

$$x^{(j)} = \sum_{i<j} o^{i,j}(x^{(i)}) \tag{5.3}$$

Dobór struktury sieci sprowadza się do dobrania parametrów $\alpha^{(i,j)}$, które odpowiadają za dobór operacji realizowanych przez krawędzie. Oprócz tego dobierane są parametry θ, które wpływają na działanie operacji (np. mogą być to parametry warstwy konwolucyjnej). Parametry te dobierane są w dwupoziomowej optymalizacji. Parametry θ dobierane są tak, aby minimalizować funkcję celu $\mathcal{L}_{train}$ szacowaną na zbiorze treningowym. Natomiast parametry $\alpha^{(i,j)}$ dobierane są tak, aby minimalizować funkcję $\mathcal{L}_{val}$ szacowaną na zbiorze walidacyjnym. Po zakończeniu optymalizacji, struktura sieci dobierana jest poprzez wybór jednej operacji dla każdej

krawędzi (i, j). Dobierane są operacje, odpowiadające elementom o najwyższej wartości wektorów $\alpha^{(i,j)}$. Metoda DARTS wyróżnia się dużą efektywnością – w przypadku klasyfikacji zbioru ImageNet, dobór struktury modułu zajmuje około czterech dni przy wykorzystaniu jednej karty graficznej.

Metoda ProxyLessNAS [95] ma za zadanie minimalizować czas odpowiedzi przez sieć neuronową, jednocześnie maksymalizując jej dokładność. Jednym z ograniczeń metody DARTS jest konieczność przechowania w pamięci danych potrzebnych do wykonania wszystkich operacji $o \in \mathcal{O}$ dla każdej krawędzi (i, j), co stanowiło istotne ograniczenie. Dlatego metoda DARTS stosowana była tylko do doboru struktury pojedynczego modułu, zamiast całej sieci neuronowej. W ProxyLessNAS zaproponowano rozwiązania, które umożliwiają dobór struktury całej sieci neuronowej. Podczas generowania odpowiedzi, algorytm przechowuje w pamięci wynik obliczeń tylko jednej wybranej operacji dla każdej krawędzi, a nie wszystkich, jak w metodzie DARTS. Operacja dobierana jest losowo, zgodnie z następującym wzorem:

$$\bar{o}^{(i,j)}(x) = \begin{cases} o_1(x) & \text{z prawdopodobieństwem } p_1 \\ \cdots & \\ o_N(x) & \text{z prawdopodobieństwem } p_N \end{cases} \tag{5.4}$$

gdzie $N = |\mathcal{O}|$, a prawdopodobieństwa p_i otrzymywane są z operacji softmax.

W inny sposób niż w metodzie DARTS dobierane są parametry $\alpha^{(i,j)}$, które mają bezpośredni wpływ na wartości prawdopodobieństw wybrania operacji. W metodzie ProxyLessNAS, aby zaktualizować parametry $\alpha^{(i,j)}$, dla każdej krawędzi losowo dobiera się dwie operacje i dokonuje się aktualizacji tylko parametrów odpowiadającym wybranym operacjom. Ostateczna architektura sieci wybierana jest poprzez wybór operacji, które odpowiadają najwyższym elementom wektorów $\alpha^{i,j}$, podobnie jak w metodzie DARTS.

Kolejną próbą usprawnienia metody DARTS była metoda Partially-Connected Darts (PC DARTS) [96]. W rozwiązaniu zastosowano częściowe połączenia kanałów (ang. *partial channel connections*) poprzez zastosowanie operacji $o \in \mathcal{O}$ tylko dla wybranych kanałów $x^{(i)}$. Do wyboru kanałów stosuje się maskę $S_{i,j}$ zawierającą wartości zero lub jeden, gdzie wartość jeden oznacza, że odpowiadający kanał podlega modyfikacji. Wartość generowana przez krawędź wynosi zatem:

$$\bar{o}(x, S) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o)}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'})} o(S * x) + (1 - S) * x \tag{5.5}$$

gdzie operacja $S * x$ oznacza wybór kanałów. Maska S jest dobierana losowo, a wybór liczby kanałów uwzględnianych w obliczeniach wiąże się z kompromisem pomiędzy dokładnością otrzymanej struktury a wymaganiami obliczeniowymi i pamięciowymi. Zmniejszenie wymagań obliczeniowych i pamięciowych umożliwia zastosowanie większego rozmiaru wsadu, co prowadzi do stabilniejszego procesu doboru struktury.

Losowy dobór kanałów powoduje dużą zmienność w proponowanych strukturach. W metodzie DARTS wyjścia krawędzi są dodawane w celu otrzymania wartości wierzchołka, natomiast w metodzie PC-DARTS stosuje się ważoną sumę, gdzie Wagi zależne są od parametrów β. Wprowadzenie normalizacji krawędzi doprowadziło również do poprawy działania oryginalnej metody DARTS. Metoda umożliwiła poprawę dokładności i szybkości działania w stosunku do metody DARTS.

Metody gradientowe za sprawą swej efektywności stały się popularnym wyborem wśród badaczy. W pracy [97], zastosowano metodę DARTS w doborze struktury sieci, która miała za zadanie wykrywanie przerzutów nowotworu. W zaproponowanym rozwiązaniu zrealizowano fuzję informacji z dwóch modalności – tomografu komputerowego oraz pozytonowo-emisyjnej tomografii komputerowej. W innej pracy [98] autorzy zaproponowali rozwiązanie oparte o metody gradientowe w celu poszukiwania kodera i dekodera sieci typu Unet. Algorytm dobierał strukturę dwóch modułów DownSC i UpSc, które odpowiadały za redukcję rozmiaru map cech (DownSC) oraz zwiększanie rozmiaru map cech (UpSC). W kolejnej pracy [99] również wykorzystano metody gradientowe w obszarze medycyny. Zaproponowany algorytm dobierał architekturę łączącą elementy struktur klasy 2D, 3D i Pseudo3D.

5.3. Proponowane rozwiązanie

W ramach doktoratu skoncentrowano się na metodach automatycznego doboru struktury w problemie klasyfikacji znamion skórnych. Zdecydowano się na wykorzystanie transformacji zachowujących funkcję, czyli zbioru modyfikacji sieci neuronowych, które umożliwiają ich rozbudowę bez utraty zdobytej wiedzy i bez pogorszenia skuteczności. Metody te umożliwiają relatywnie szybkie i wydajne uzyskiwanie kolejnych, lepszych i bardziej rozbudowanych wersji sieci bez znaczącego

nakładu obliczeniowego. Do nadzoru procesu doboru struktury wybrano algorytm wspinaczkowy [100], który charakteryzuje się niskim zapotrzebowaniem obliczeniowym. Proces składa się z trzech następujących po sobie etapów: trening sieci początkowej, dobór struktury za pomocą algorytmu wspinaczkowego, trening finalnej sieci. W kolejnych sekcja przedstawiono szczegółowy opis zaproponowanego rozwiązania.

W celu oceny skuteczności zaproponowanej metody porównano sieci neuronowe zaprojektowane przez algorytm z sieciami opisywanymi w rozdziale 4. Zaproponowane rozwiązanie najbardziej zbliżone jest do pracy [83].

5.3.1 Transformacje zachowujące funkcję

Transformacje zachowujące funkcję (ang. *function preserving transformation* to rodzina metod, które umożliwiają rozbudowę struktury sieci neuronowej, zachowując przy tym wcześniej zdobytą wiedzę. Metody te mogą być wykorzystane w algorytmach doboru struktury w celu przyspieszenia przeszukiwania przestrzeni dopuszczalnych rozwiązań.

Modyfikacja sieci neuronowej $f(\cdot)$ polega na zastosowaniu transformacji zachowującej funkcję, czego wynikiem jest nowa sieć $g(\cdot)$. Przy tym zachodzi zależność:

$$\forall x, f(x) = g(x) \tag{5.6}$$

gdzie x, jest wejściem sieci neuronowej.

Zastosowanie metody polega na znalezieniu parametrów θ' sieci $g(\cdot)$, aby spełnione zostało równanie:

$$\forall x, f(x, \theta) = g(x, \theta') \tag{5.7}$$

W praktyce, dobierane są tylko parametry komponentów dodanych do sieci (np. parametry dodanej warstwy).

Sieć $g(\cdot)$ generuje takie same wyjścia jak sieć $f(\cdot)$, co oznacza, że ich skuteczność jest taka sama. Jednakże, ze względu na bardziej złożoną strukturę sieci $g(\cdot)$, ma ona większą pojemność, co pozwala na zwiększenie jej skuteczności poprzez dodatkowy trening.

W ramach proponowanego rozwiązania zastosowano: operacje dodawania nowych warstw, operacje powiększania istniejących warstw, operacje dodawania struktury agregacji metodą dodawania, operacje dodawania struktury agregacji metodą

konkatenacji oraz operacje dodawania połączenia skracających. Taki wybór operacji umożliwia generację złożonych struktur sieci neuronowych zdolnych do odwzorowania skomplikowanych zależności w danych. Do realizacji tego celu wykorzystano metody Net2WiderNet oraz Net2Deeper, szczegółowo opisane w [101]. Metody tworzące połączenia wielogałęziowe oparto na tych modyfikacjach.

Operacja Net2WiderNet umożliwia zwiększenie rozmiaru istniejącej warstwy poprzez dodanie nowych neuronów lub filtrów. Aby rozszerzyć warstwę i, należy zmodyfikować jej macierz parametrów $\boldsymbol{W}^{(i)}$ oraz macierz parametrów kolejnej warstwy $\boldsymbol{W}^{(i+1)}$. Jeśli warstwa i ma m wejść i n wyjść, a warstwa $i+1$ ma p wyjść, to $\boldsymbol{W}^{(i)} \in \mathbb{R}^{m \times n}$ i $\boldsymbol{W}^{(i+1)} \in \mathbb{R}^{n \times p}$. Metoda rozszerzania warstw umożliwia zwiększenie liczby wyjść w warstwie i na q, gdzie $q > n$. Nie jest jednak możliwe zmniejszenie liczby elementów w warstwie.

Zwiększanie rozmiaru warstwy przebiega następująco. Pierwszym krokiem jest wprowadzenie funkcji losowego przypisania $r(j)$:

$$r(j) = \begin{cases} j & j \leqslant n \\ \text{losowa liczba ze zbioru } \{1, 2, \cdots, n\} & j > n \end{cases} \tag{5.8}$$

Losowy wybór liczb dokonywany jest ze zwracaniem. Wykorzystując funkcję $r(j)$ można dokonać zwiększenia rozmiaru warstwy poprzez wprowadzenie nowych macierzy parametrów $\boldsymbol{U}^{(i)}$ oraz $\boldsymbol{U}^{(i+1)}$ zgodnie z poniższym równaniem.

$$\boldsymbol{U}^{(i)}_{k,j} = \boldsymbol{W}^{(i)}_{k,r(j)}, \quad \boldsymbol{U}^{(i+1)}_{k,j} = \frac{1}{|\{x|r(x) = r(j)\}|} \boldsymbol{W}^{(i+1)}_{r(j),h} \tag{5.9}$$

Pierwsze n kolumn macierzy $\boldsymbol{W}^{(i)}$ kopiowane jest do macierzy $\boldsymbol{U}^{(i)}$. Następnie, kolumny od $n+1$ do q tworzone są poprzez losowy dobór ze zwracaniem kolumn z macierzy $\boldsymbol{W}^{(i)}$. Selekcja ze zwracaniem pozwala na wielokrotne użycie tych samych kolumn. Wiersze macierzy $\boldsymbol{W}^{(i+1)}$ są również powielane i mnożone przez współczynnik skalowania $\frac{1}{|\{x|r(x)=r(j)\}|}$. Współczynnik skalowania stosuje się, aby uwzględnić wzrost liczby wyjść poprzedniej warstwy.

Metoda może zostać wykorzystana do powiększenia warstw konwolucyjnych. W przedstawionym przykładzie warstwa była powiększana poprzez dodawanie kolejnych neuronów oraz kopiowanie kolumn macierzy, które reprezentują parametry

określonych neuronów. Rozszerzanie warstwy konwolucyjnej działa w analogiczny sposób, z tą różnicą, że kopiowane są parametry filtrów.

Elementy warstwy (filtry lub neurony) powielone poprzez kopiowanie parametrów mogą realizować tę samą funkcję co inne jednostki, co może pogarszać proces uczenia. Aby ograniczyć ten problem, można zastosować metody losowe np. wprowadzając metodę dropout lub poprzez dodanie szumu o niewielkiej wariancji do parametrów dodawanych jednostek.

Operacja Net2Deeper stosowana jest w celu dodania warstwy w strukturze sieci neuronowej. W ramach tej operacji parametry nowej warstwy inicjalizowane są tak, aby warstwa realizowała funkcję tożsamościową. W odróżnieniu od operacji Net2WiderNet, zastosowanie operacji Net2Deeper nie wymaga modyfikacji parametrów innych warstw. Liczba jednostek w nowej warstwie powinna być taka sama jak w poprzedniej warstwie.

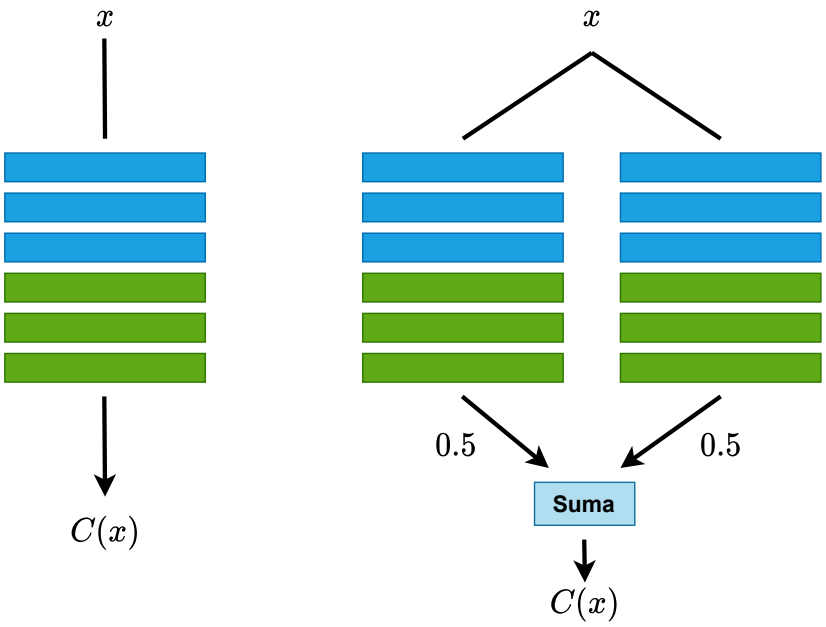
W pracy zastosowano transformacje umożliwiające tworzenie gałęzi, które agregowane są przy pomocy operacji sumowania i konkatenacji. Oprócz tego zastosowano metodę wprowadzania połączeń skracających.

W celu utworzenia dwóch gałęzi agregowanych metodą sumowania (rys. 5.3), do istniejącej warstwy dodawana jest druga warstwa równoległa o identycznych parametrach. Aby zachować funkcję realizowaną przez warstwę, wyjścia obu warstw są sumowane przy zastosowaniu wag o wartości 0,5:

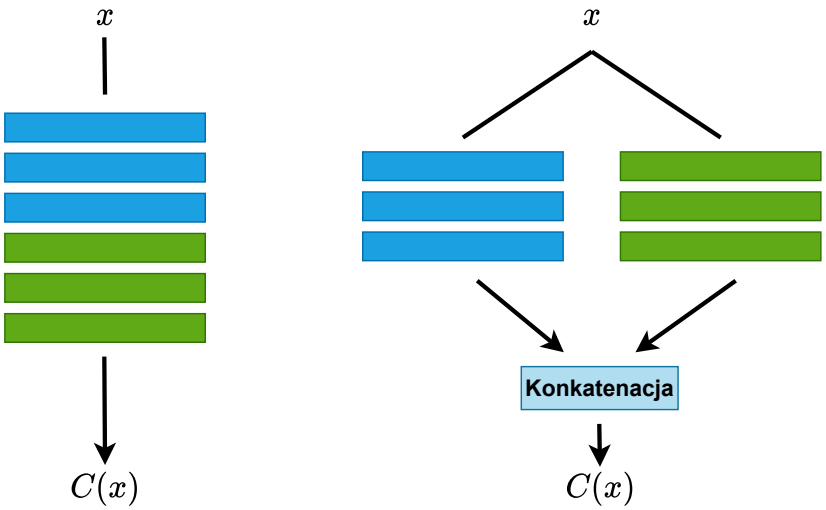
$$C(x) = 0,5 \cdot C(x) + 0,5 \cdot C(x) \tag{5.10}$$

Innym rodzajem połączeń równoległych są połączenia agregowane metodą konkatenacji (rys. 5.4). Wprowadzenie takiego połączenia polega na podziale jednostek warstwy (neuronów lub filtrów) na dwie warstwy działające równolegle, z których każda powinna zawierać połowę elementów pierwotnej warstwy. Wyjścia obu warstw poddane są operacji konkatenacji, dzięki czemu struktura działa w identyczny sposób jak pojedyncza warstwa przed podziałem.

W zaproponowanym rozwiązaniu wprowadzono także możliwość dodania połączeń skracających, znanych z rodziny architektur ResNet. Wprowadzenie schematu polega na dodaniu nowej warstwy realizującej funkcję tożsamościową oraz równoległego do niej połączenia skracającego. Wyjścia połączenia skracającego i nowej warstwy są następnie sumowane z wagą równą 0,5.



Rysunek 5.3: Tworzenie połączeń dwugałęziowych agregowanych metodą sumowania

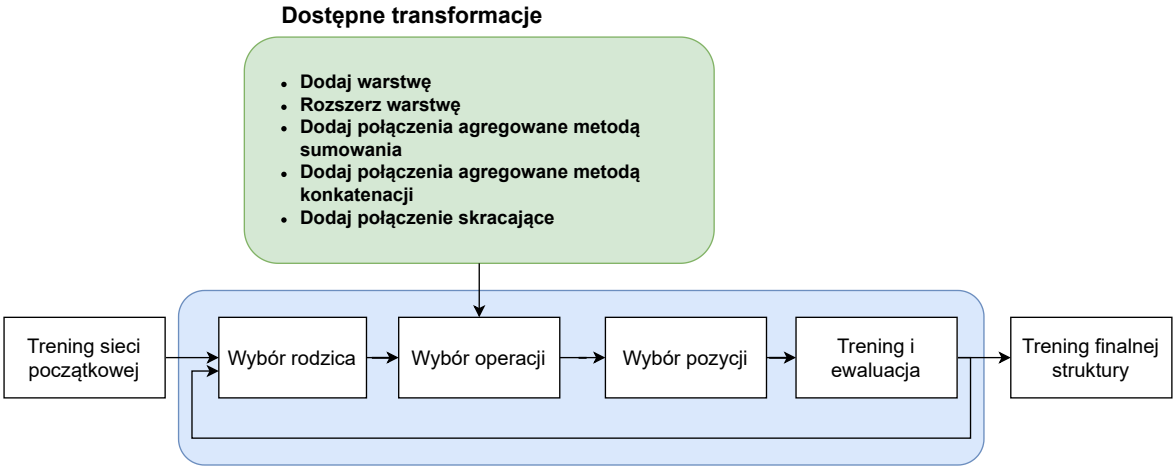


Rysunek 5.4: Tworzenie połączeń dwugałęziowych agregowanych metodą konkatenacji

Przedstawione modyfikacje wprowadzające wiele gałęzi początkowo działają jak pojedyncza warstwa. Dopiero po dalszym uczeniu sieci możliwe jest wykorzystanie pełni możliwości tych schematów. Ponadto wprowadzone schematy wielogałęziowe mogą być poddane dalszym modyfikacjom, tworząc złożone struktury.

5.3.2 Algorytm wspinaczkowy

W celu nadzorowania procesu doboru struktury zastosowano algorytm wspinaczkowy [100]. Algorytm ten może być interpretowany jako uproszczona wersja algorytmu genetycznego bez operacji krzyżowania i z jednym organizmem przechodzącym do następnej populacji. Schemat zastosowanego algorytmu widoczny jest na rys. 5.5.



Rysunek 5.5: Schemat proponowanego rozwiązania

Zastosowanie algorytmu wiąże się z koniecznością wyboru początkowej struktury sieci, która następnie będzie podlegała modyfikacjom w każdej iteracji algorytmu. W każdej iteracji generowane są nowe organizmy, których struktura jest rozwinięciem struktury rodzica. Następnie, każdy z organizmów (sieci) jest trenowany przez określoną liczbę epok. Dzięki zastosowaniu transformacji zachowujących funkcje nie ma potrzeby uczenia generowanych sieci od podstaw. Na przykład, jeśli rodzic zakończył trening z dokładnością 70%, każdy z wygenerowanych potomków na początku uczenia osiąga tę dokładność. Dodatkowy trening umożliwia wykorzystanie pojemności modeli, która została zwiększona dzięki wprowadzeniu nowych elementów struktury.

W trakcie działania algorytmu sieć może zostać poddana następującym modyfikacjom:

- dodanie warstwy,
- dodanie filtrów do istniejącej warstwy,
- dodanie połączenia gałęziowego agregowanego metodą sumowania,
- dodanie połączenia gałęziowego agregowanego metodą konkatenacji,
- dodanie połączenia skracającego.

Każdy nowy potomek tworzony jest poprzez losowy wybór dwóch operacji. Dla każdej operacji losowo dobierane jest miejsce zastosowania modyfikacji w strukturze sieci neuronowej (np. warstwa, która powinna być poddana powiększeniu). W przypadku, gdy wylosowana modyfikacja nie jest możliwa do zrealizowania (np. zwiększenie rozmiaru warstwy jednej z gałęzi, które są łączone za pomocą metody konkatenacji), modyfikacja i jej parametry losowane są ponownie.

5.4. Opis przeprowadzonych eksperymentów

Przedstawiony algorytm doboru struktury wykorzystany został w celu doboru struktury sieci neuronowej realizującej zadania klasyfikacji znamion skórnych.

5.4.1 Przetwarzanie danych

W celu porównania wyników struktur dobranych przez algorytm z tymi, które zostały dobrane ręcznie (opisane w poprzednim rozdziale), zastosowano identyczny zbiór danych oraz metody jego przetwarzania (opisany w sekcji 4.3.1). Znamiona zostały wycięte przy użyciu dostarczonych masek, następnie dokonano normalizacji obrazów poprzez sprowadzenie średniej wartości pikseli do zera a odchylenia standardowego do jeden. Rozmiar wejściowy obrazów wynosił $224 \times 224 \times 3$ piksele.

Zastosowano ten sam podział na zbiór treningowy, walidacyjny i testowy. Zbiór testowy i walidacyjny składały się z 200 obrazów każdy (100 obrazów na klasę), a reszta zbioru została wykorzystana jako zbiór treningowy. W celu zrównoważenia liczebności klas, zastosowano powielenie obrazów mniej licznej klasy (znamiona ze zmianami złośliwymi).

Zastosowano takie same metody agumentacji danych jak w eksperymentach opisanych w rozdziale 4 tj. obroty, rozciągania, odwrócenie i powiększenie.

Rozszerzanie danych zostało przeprowadzone online podczas treningu, przed podaniem obrazów na wejście sieci neuronowej.

5.4.2 Wartość odniesienia

W celu oceny struktur dobranych przez algorytm postanowiono porównać je z rozwiązaniami testowanymi w poprzednim rozdziale. Z powodu dużego obciążenia obliczeniowego nie zastosowano metody k-krotnej walidacji krzyżowej. Rezultaty sieci otrzymanych w wyniku działania algorytmu porównano z sieciami z rodziny VGG (VGG8, VGG11 oraz VGG16) przestawionymi w tabeli 5.1.

Tabela 5.1: Lista przeprowadzonych eksperymentów – wartość odniesienia

Eksperyment	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Architektura	VGG8	VGG8	VGG8	VGG8	VGG11	VGG16	VGG16
Funkcja aktywacji	ReLU	Leaky ReLu	ReLU	ReLU	ReLU	ReLU	ReLU
Batch normalization		✓	✓				
Dropout				✓	✓	✓	✓
Transfer learning							✓

5.4.3 Proces doboru struktury

Przeprowadzone eksperymenty składały się z trzech etapów: trening sieci początkowej, dobór struktury oraz trening wynikowej struktury. Wszystkie treningi zostały przeprowadzone przy użyciu optymalizatora SGDR, którego działania polega na stopniowym zmniejszaniu współczynnika uczenia, podążając za funkcją kosinus. Współczynnik uczenia jest cyklicznie resetowany po osiągnięciu ustalonej minimalnej wartości. Opis optymalizatora znajduje się w sekcji 3.3. Z uwagi na ograniczenia pamięci karty graficznej, we wszystkich etapach algorytmu, rozmiar wsadu wynosił 8.

Aby zredukować wymagania obliczeniowe algorytmu doboru struktury, w trakcie działania algorytmu wykorzystano 33% zbioru uczącego (obrazy zostały wybrane losowo). Ostateczny trening wybranej architektury został przeprowadzony przy użyciu pełnego zbioru danych.

Architektura sieci początkowej

W eksperymentach jako sieć początkową przyjęto prostą, jednokierunkową sieć o wejściu o rozmiarze $224 \times 224 \times 3$. Struktura składa się z sześciu warstw

konwolucyjnych, przy czym po każdej z nich umiejscowiono warstwę batch normalization oraz funkcję aktywacji ReLU. Każda z warstw konwolucyjnych składa się z filtrów o rozmiarze 3×3 o kroku równym 1. Ponadto zastosowano warstwy MaxPooling z oknem o rozmiarze 2×2 oraz krokiem równym 2. Zastosowano regularyzację L_2 ze współczynnikiem λ równym 0,0005. Struktura sieci początkowej przedstawiona jest w tabeli 5.2.

Jako funkcję celu zastosowano binarną krosentropię krzyżową. Trening trwał 60 epok, a wartość współczynnika uczenia optymalizatora była stopniowo zmniejszana w sposób kosinusoidalny od 0,01 do 0, z restartem co 10 epok.

Tabela 5.2: Struktura sieci początkowej

Sieć początkowa
conv128
MaxPooling
conv128
MaxPooling
conv128
MaxPooling
conv128
MaxPooling
conv128 conv128
Neuron sigmoidalny

Dobór struktury

Po przeprowadzeniu treningu struktura początkowa służy jako punkt wyjściowy dla algorytmu doboru struktury. W każdej iteracji generowanych jest pięć potomków, poprzez zastosowanie dwóch losowo wybranych transformacji najlepszej sieci z poprzedniej iteracji algorytmu ewolucyjnego. W celu zachowania potencjału najlepszej sieci z danej iteracji, przechodzi ona niezmieniona do kolejnej iteracji jako szósty organizm.

Wszystkie stosowane transformacje działają na filtrach konwolucyjnych o rozmiarze 3×3, w każdej warstwie stosowana jest regularyzacja L_2 ze współczynnikiem λ wynoszącym 0,0005. Każdy nowopowstały organizm trenowany jest przez osiem epok, z wykorzystaniem algorytmu SGDR ze stopniową redukcją współczynnika uczenia od 0,005 do 0. Wybrana liczba epok jest kompromisem pomiędzy czasem działania algorytmu a dokładnością procesu doboru struktury. W pracy [83] wykazano, że wyniki uzyskane przez sieci poddane krótkotrwałemu treningowi z wykorzystaniem algorytmu SGDR, są bardziej skorelowane z finalną końcową skutecznością mierzoną na zbiorze testowym, w porównaniu do algorytmu SGD. Właściwość ta umożliwia lepsze kierowanie procesem doboru struktury. Po każdej iteracji wszystkie organizmy są testowane, a najlepszy organizm zostaje rodzicem w kolejnej iteracji. W przypadku, gdy żaden z potomków nie osiągnie lepszej dokładności niż rodzic, rodzic pozostaje niezmieniony.

Algorytm doboru struktury kończy działanie po upłynięciu 15 iteracji. Liczba iteracji została dobrana jako kompromis pomiędzy dokładnością sieci, jej wynikowym rozmiarem i czasem poszukiwania. Konsekwencją zbyt długotrwałego doboru struktury są nadmiernie rozbudowane struktury. Przykładowo, po 10 iteracjach algorytmu powstała sieć zawierająca 1,5 mln parametrów, a po 25 iteracjach – sieć z 5,2 mln parametrów.

Trening finalnej struktury

Najlepszy model w ostatniej iteracji algorytmu wspinaczkowego poddawany jest dłuższemu treningowi. Trening przeprowadzony jest przez 200 epok, wykorzystując algorytm SGDR ze współczynnikiem uczenia stopniowo zmniejszającym się od 0,005 do 0. Współczynnik uczenia resetowany jest co 25 epok. Stosowane są takie same metody rozszerzania danych jak w uczeniu sieci początkowej oraz poszukiwaniu struktury.

5.5. Wyniki

W celu pokazania efektywności zaproponowanego rozwiązania zaplanowano i przeprowadzono serię eksperymentów. Wszystkie raportowane wyniki zostały uzyskane na zbiorze testowym. W celu zminimalizowania niepewności proces doboru

struktury został powtórzony czterokrotnie, a następnie wyznaczono średnią stosowanych miar. Rezultaty zaprezentowane są w tabeli 5.3.

Tabela 5.3: Dobór struktury z wykorzystaniem dokładności

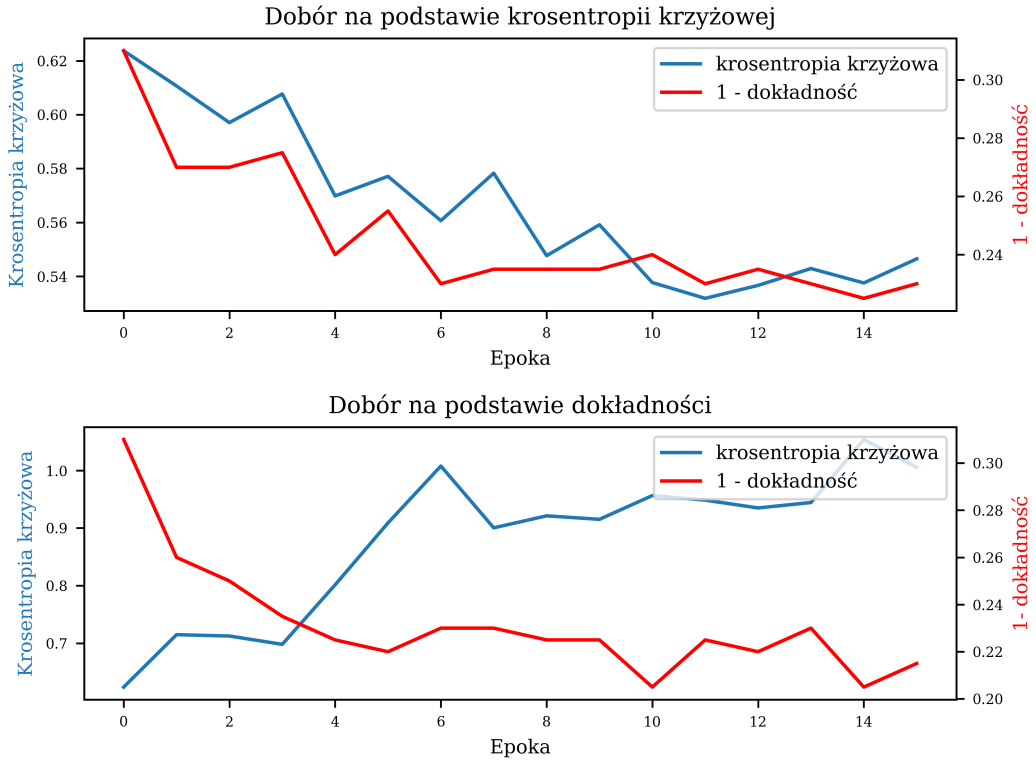
Numer architektury	**Dokładność**	**ROC AUC**	**Liczba parametrów**
1	72,0	0,792	1 527 873
2	70,0	0,779	2 228 609
3	72,5	0,805	2 083 841
4	74,5	0,827	1 342 593
Średnia	72,25	0,801	1 795 729

W trakcie eksperymentów obserwowano sytuacje, kiedy więcej niż jeden organizm osiągał taką samą dokładność, co utrudniało wybór najlepszego organizmu, który stawał się rodzicem w kolejnej iteracji. Ponadto, często występowała sytuacja, kiedy model osiągający najniższą wartość funkcji celu mierzonej na zbiorze walidacyjnym, nie był wybierany jako rodzic, z powodu osiągnięcia niższej dokładności.

W rozwiązaniach opisywanych w literaturze wybór modelu często dokonywany jest na podstawie dokładności mierzonej na zbiorze walidacyjnym. Jednakże stosowanie dokładności mierzonej na niewielkim zbiorze walidacyjnym, może prowadzić do niewiarygodnych wyników. Dokładność może przyjąć skończoną liczbę wartości równą liczbie przykładów w zbiorze walidacyjnym. Dlatego mały rozmiar zbioru walidacyjnego zwiększa prawdopodobieństwo, że modele mogą osiągać taką samą dokładność, utrudniając wybór rodzica. Z tego powodu, zdecydowano się dokonywać ewaluacji organizmów, stosując miarę binarnej krosentropii krzyżowej, która jest metryką ciągłą i niesie więcej informacji na temat poprawności działania modelu, uwzględniając pewność modelu co do generowanej odpowiedzi. Porównanie doboru struktury z wykorzystaniem dwóch sposobów ewaluacji widoczne jest na rys. 5.6.

W trakcie eksperymentów zauważono, że stosowanie binarnej krosentropii krzyżowej pozwala na stabilny spadek tej miary przy jednoczesnym wzroście dokładności. W przypadku doboru najlepszego modelu na podstawie dokładności, dokładność zazwyczaj spada, a wartości entropii krzyżowej rośnie. Duża wartość miary entropii krzyżowej może prowadzić do pogorszenia wartości ROC AUC.

Na podstawie tych spostrzeżeń przeprowadzono dobór struktury z wykorzystaniem entropii krzyżowej jako metryki stosowanej do selekcji potomków. To podejście doprowadziło do widocznej poprawy wyników, co można zaobserwować w tabeli 5.4.



Rysunek 5.6: Porównanie doboru struktury przy użyciu różnych miar. Zastosowanie dokładności jako miary nie prowadzi do spadku krosentropii krzyżowej

Tabela 5.4: Dobór struktury z wykorzystaniem binarnej krosentropii krzyżowej

Numer architektury	**Dokładność**	**ROC AUC**	**Liczba parametrów**
1	76,5	0,825	1 198 593
2	74,0	0,812	1 785 601
3	77,0	0,823	1 054 593
4	78,0	0,845	750 849
Średnia	76,38	0,826	1 197 409

W tabeli 5.5 przedstawiono podsumowanie wyników oraz porównanie struktur dobranych ręcznie i za pomocą algorytmu. Struktury wygenerowane przez algorytm osiągają zbliżone wyniki do ręcznie opracowanych struktury z rodziny VGG, które były powszechnie stosowane w czasie prowadzenia eksperymentów. Najlepsze wyniki zostały osiągnięte przez najmniejszą sieć (nr 4). Sieć ta osiąga lepszą dokładność (78%) oraz minimalnie gorszą wartość ROC AUC (0,845) w porównaniu do sieci VGG16 wstępnie przeuczonej na zadaniu ImageNet, gdzie dokładność wynosiła 75,75% a ROC AUC 0,847. Należy podkreślić, że różnica pomiędzy wartościami ROC AUC była pomijalnie niska i wynosiła zaledwie 0,002. Pomimo osiągnięcia porównywalnych wyników, warto zauważyć znacząca redukcję parametrów o ponad 98% z 41456499 do 750849.

Tabela 5.5: Podsumowanie wyników i porównanie ze strukturami dobranymi ręcznie. Rozwinięcia skrótów: LReLU – Leaky ReLU, BN – batch normalization, DO – dropout, TF – transfer learning

Eksperyment	**Dokładność**	**ROC AUC**	**Liczba parametrów**
A VGG8 ReLU	70,58	0,803	30 652 545
B VGG8 LReLU BN	72,33	0,811	30 652 545
C VGG8 ReLU BN	75,08	0,841	30 659 587
D VGG8 ReLU DO	74,75	0,827	30 652 545
E VGG11 ReLU DO	69,42	0,780	35 971 843
F VGG16 ReLU DO	67,83	0,748	41 456 449
G VGG16 ReLU DO TF	75,75	0,847	41 456 449
Dobór – krosentropia krzyżowa	76,38	0,826	1 197 409
Dobór – dokładność	72,63	0,810	1 795 729
Dobór – krosentropia – najlepszy model	78,0	0,845	750 849
Dobór – krosentropia – grupowanie 4 modeli	77,00	0,843	7 182 916

Czas doboru struktury wynosił 18 godzin przy użyciu jednej karty graficznej (GeForce 1080Ti). Krótki czas doboru struktury osiągnięto dzięki zastosowaniu transformacji zachowujących funkcje oraz dzięki ograniczeniu rozmiaru zbioru uczącego w trakcie doboru. Dobór struktury przeprowadzony na pełnym zbiorze danych prowadzi do takich samych wyników jak w przypadku zastosowania jednej trzeciej zbioru. Należy podkreślić, że czas doboru jest zbliżony do czasu uczenia

architektury, który wynosi od kilku do kilkunastu godzin, w zależności od dobranych hiperparametrów i architektury.

Na rys. 5.7 i 5.8 przedstawiono strukturę sieci nr 4 (pozostałe struktury zamieszczono w Dodatku A), a w tabeli 5.6 przedstawiono parametry warstw. Spośród czterech sieci otrzymanych w wyniku działania algorytmu, sieć nr 4 uzyskała najlepszy wynik, a zarazem jest najmniejsza. Struktura ma cechy struktury ResNet50 w postaci połączenia skracającego (ang. *shortcut connection*) oraz struktury Inception w postaci modułu złożonego z równoległych warstw, których wyjścia łączone są metodami sumowania i konkatenacji.

Tabela 5.6: Struktura otrzymana w wyniku działania algorytmu – sieć 4

Warstwa	**Liczba filtrów**
conv2d	128
conv2d_1	128
conv2d_2	128
conv2d_3	128
conv2d_6	128
conv2d_8	64
conv2d_9	64
conv2d_10	256
conv2d_11	128

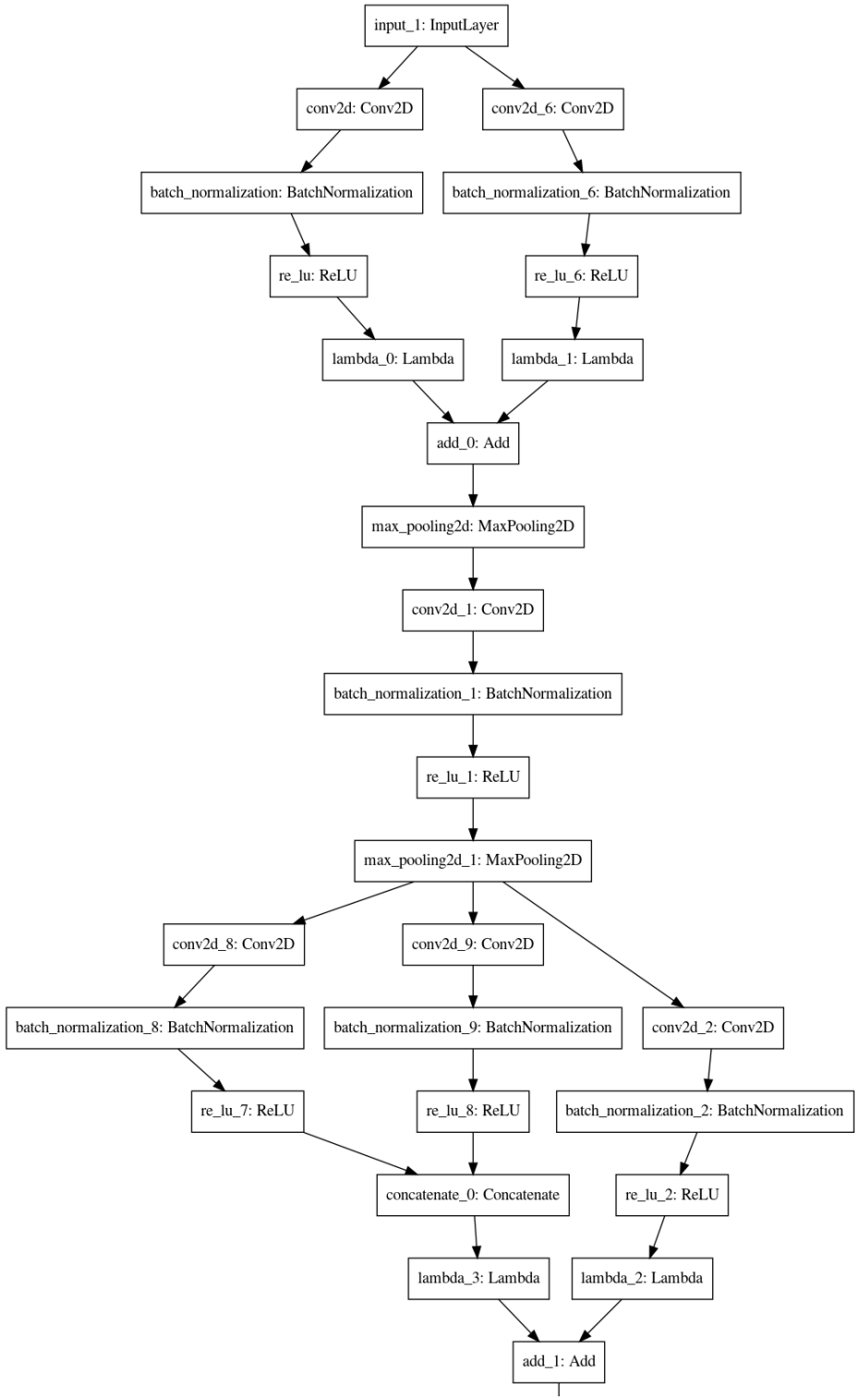
Proces doboru struktury został powtórzony cztery razy, co skutkowało powstaniem czterech różnych struktur. W celu poprawy dokładności postanowiono wykorzystać metodę grupowania modeli. Zastosowanie tej metody umożliwiło uzyskanie wyników wyższych niż średnia wartość wyników uzyskanych przez cztery modele. Niemniej jednak, grupowanie modeli nie przyniosło lepszych wyników niż najlepszy model wchodzący w skład grupy.

5.6. Podsumowanie

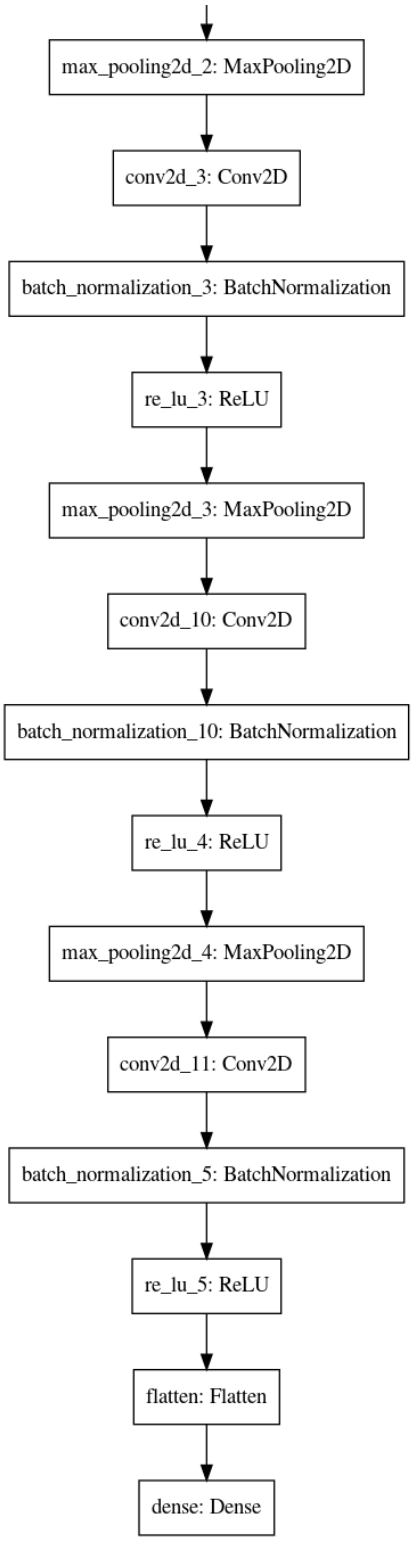
W badaniach zaproponowano system doboru struktury sieci neuronowej oparty o modyfikacje zachowujące funkcję i algorytm wspinaczkowy. Ponowne zastosowanie parametrów doprowadziło do dużej redukcji kosztu obliczeniowego. Proponowane rozwiązanie rozwija sieć poprzez dodawanie kolejnych jej elementów, zaczynając od

niewielkiej struktury. Otrzymane struktury posiadały 98% mniej parametrów niż sieci przygotowane ręcznie, uzyskując przy tym zbliżoną dokładność.

Przyspieszenie doboru struktury możliwe było dzięki zastosowaniu mniejszego zbioru danych, który stanowił tylko 33% zastosowanego zbioru. Czas doboru struktury wynosił około 18 godzin przy użyciu jednej karty graficznej. Wadą proponowanego rozwiązania jest brak możliwości usuwania lub zmniejszania rozmiaru warstw.



Rysunek 5.7: Schemat struktury otrzymanej w wyniku działania algorytmu: sieć 4 – część 1



Rysunek 5.8: Schemat struktury otrzymanej w wyniku działania algorytmu: sieć 4 – część 2

6. Dwuetapowy proces uczenia

6.1. Wprowadzenie

Jednym z głównych ograniczeń w stosowaniu sieci neuronowych jest znaczne zapotrzebowanie na dane uczące. Przeprowadzone eksperymenty, opisane w rozdziale 4, wykazały, że trening sieci neuronowych z losową inicjalizacją parametrów na małym zbiorze uczącym prowadzi do słabych wyników. W celu zwiększenia skuteczności sieci uczonych na małych zbiorach danych, szeroko stosuje się technikę transfer learning.

Metoda transfer learning polega na inicjalizacji parametrów sieci neuronowej, parametrami sieci poddanej wcześniej wstępnemu uczeniu na innym zadaniu, nazywanym dalej zadaniem pomocniczym. Głęboka sieć, która jest w stanie realizować określone zadanie, zazwyczaj może również realizować inne, podobne zadania po dostrojeniu parametrów. Jest to możliwe dzięki wykorzystaniu wielu warstw, które odpowiadają za ekstrakcję cech. W głębokich sieciach neuronowych stosowanych w analizie obrazu, za ekstrakcję cech odpowiadają warstwy konwolucyjne, które uczą się rozpoznawać cechy powszechne w większości analizowanych problemów.

Zadania analizy obrazu wiążą się z wyodrębnieniem informacji dotyczących krawędzi, kolorów, kształtów, efektów związanych z oświetleniem i cieniem. Wyuczony ekstraktor cech, który jest w stanie wykrywać te cechy może zostać użyty w szerokiej gamie innych zastosowań. Dlatego, zamiast uczyć sieć na zadaniu docelowym od podstaw, można douczyć wstępnie przetrenowaną sieć neuronową (ang. *fine-tuning*). Podczas douczania, nie jest konieczna nauka wszystkich umiejętności przez sieć, ponieważ wiele z nich zostało nabyte podczas wstępnego uczenia na innym zadaniu. W ten sposób wiedza zdobyta w trakcie treningu na zadaniu pomocniczym

może być wykorzystana do realizacji zadania docelowego. Metoda transfer learning okazuje się najbardziej efektywna, gdy zadanie pomocnicze i docelowe są do siebie zbliżone pod względem cech, np. rozpoznaje się podobne typy obiektów [102].

Dominującą praktyką jest stosowanie zadania pomocniczego w postaci uczenia nadzorowanego. Do zadań klasyfikacji stosuje się zazwyczaj struktury sieci neuronowych poddane uczeniu na zadaniu klasyfikacji zbioru ImageNet. Technika stała się na tyle popularna, że biblioteki programistyczne stosowane do tworzenia sieci neuronowych, takie jak Tensorflow, Pytorch udostępniają gotowe modele sieci neuronowych przeuczone na tym zadaniu. W praktyce, wykorzystanie metody transfer learning jest stosunkowo proste w przypadku popularnych architektur sieci neuronowych, ponieważ wymaga to jedynie pobrania parametrów sieci z publicznego repozytorium. W przypadku nietypowych architektur, których wstępnie przeuczonych wersji nie można znaleźć w repozytoriach, konieczne jest przeprowadzenie wstępnego uczenia. Należy jednak mieć na uwadze, że taki proces może być czasochłonny.

W obszarze medycyny, zastosowanie zadania pomocniczego w postaci uczenia nadzorowanego jest trudne ze względu na ograniczoną liczbę dużych, oznaczonych zbiorów danych. Ponadto, w obrazowaniu medycznym używa się różnych modalności (np. różne sekwencje rezonansu magnetycznego, tomografia komputerowa, rentgen) oraz obrazów różnych organów, co wymaga zastosowania różnych zadań pomocniczych. Dzięki postępującej cyfryzacji ośrodków medycznych gromadzone są duże ilości danych, jednak często są one nieoznaczone. Przykładem takich zbiorów są biobanki, które poza materiałem biologicznym gromadzą również dane. Przykładowo, UK Biobank [103] zawiera około 50 tys. obrazów MRI mózgu, jednak obrazy nie są oznaczone, co uniemożliwia przeprowadzenia wstępnego treningu w sposób nadzorowany. Z tego powodu, wykorzystanie tych zbiorów do wstępnego treningu w sposób samonadzorowany (ang. *self-supervised learning*) stało się atrakcyjnym i skutecznym podejściem jako alternatywa dla wstępnego uczenia w sposób nadzorowany.

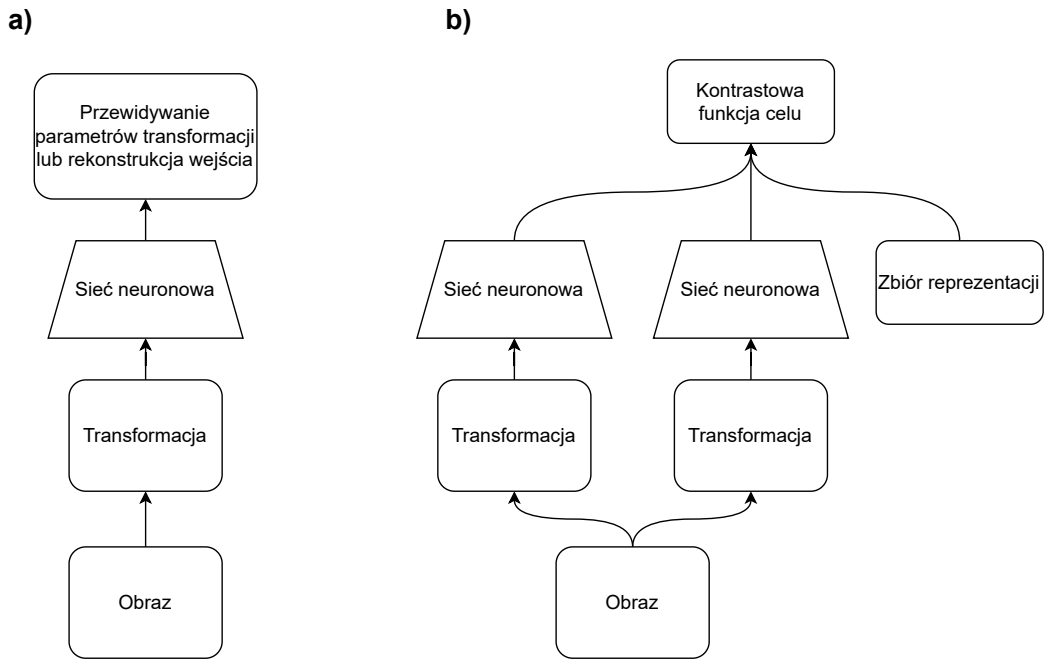
Wyniki opisanych w rozdziale badań opublikowano w [24].

6.2. Przegląd rozwiązań

Spośród metod uczenia samonadzorowanego stosowanych w analizie obrazu najpopularniejszymi są metody oparte o zadanie pretekstowe oraz metody oparte

o kontrastową funkcję celu. W metodach opartych na zadaniu pretekstowym, automatycznie generuje się pary uczące (przykład uczący i oczekiwana odpowiedź) bazując na cechach analizowanego zbioru danych. Wygenerowany w ten sposób zbiór umożliwia przeprowadzenie wstępnego uczenia w sposób nadzorowany. Większość zadań pretekstowych opiera się na odtworzeniu brakującej części obrazu lub zastosowaniu transformacji obrazu, a następnie przewidywaniu przez sieć parametrów zastosowanej transformacji. Przykładowo, obrazy można obracać o losowo wybrany kąt, a następnie nauczyć sieć przewidywać zastosowany kąt obrotu. Zbiór uczący w tym przypadku można wygenerować w sposób automatyczny przy pomocy skryptu.

Obecnie dominującym podejściem są metody oparte o tzw. kontrastową funkcję celu (ang. *contrastive loss function*). Główną ideą w tej klasie metod jest minimalizacja odległości pomiędzy reprezentacjami różnych modyfikacji tego samego obrazu, przy równoczesnym oddalaniu reprezentacji różnych obrazów. Podejścia zostały zilustrowane na diagramie na rys. 6.1.

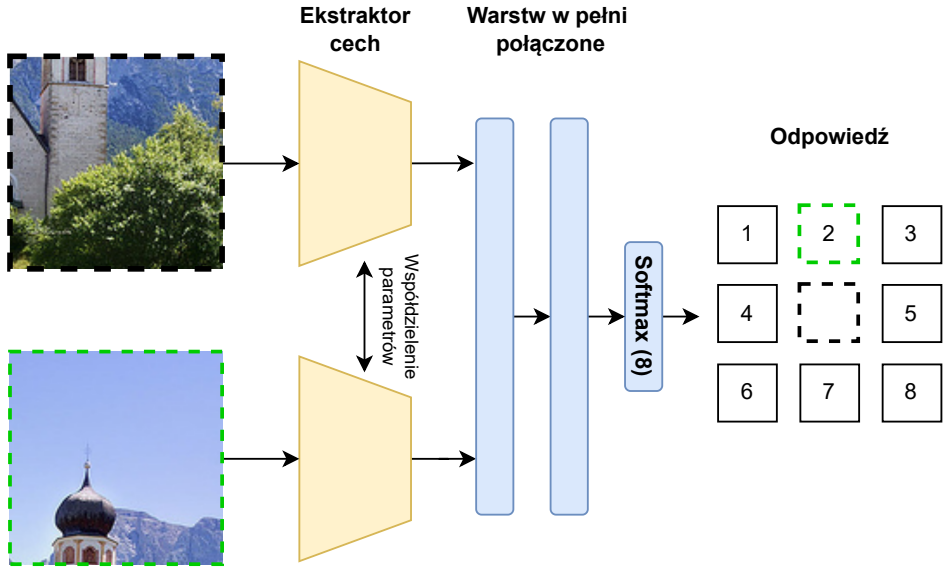


Rysunek 6.1: Uczenie samonadzorowane: a) uczenie z zadaniem pretekstowym, b) uczenie z kontrastową funkcją celu

6.2.1 Metody oparte o zadanie pretekstowe

Jedną z pierwszych prac nad zastosowaniem zadań pretekstowych w obszarze przetwarzania obrazu jest [104]. Zadaniem sieci jest określenie relacji pomiędzy

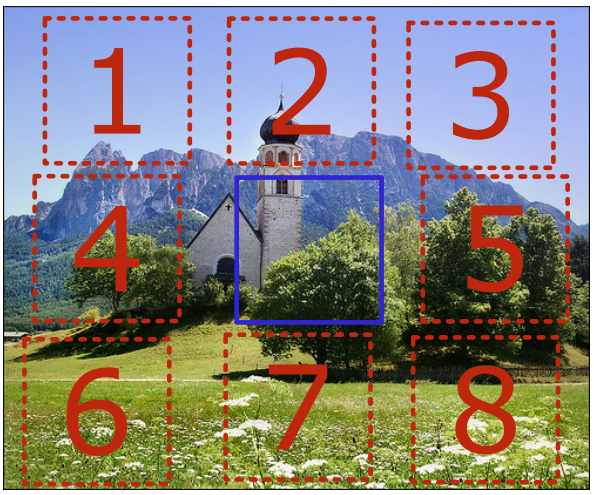
dwoma fragmentami obrazu (rys. 6.3). W rozwiązaniu zastosowano dwie sieci neuronowe, współdzielące wagi, których wejście stanowią dwa fragmenty obrazu. Reprezentacje wygenerowane przez te sieci stanowią wejście kolejnych trzech warstw w pełni połączonych, które generują odpowiedź określającą relatywne położenie fragmentów. Schemat rozwiązania przedstawiony jest na rys. 6.2. Realizacja tego zadania wymaga umiejętności analizy sceny i obiektów na obrazie przez sieć neuronową, co jest również istotne w realizacji zadań docelowych.



Rysunek 6.2: Zadanie kontekstowe polegające na przewidywaniu relacji pomiędzy fragmentami obrazu

Częstym problemem w uczeniu samonadzorowanym jest pojawianie się tzw. trywialnych rozwiązań (ang. *trivial solutions*), w których sieć dokonuje tzw. „pójścia na skróty" (ang. *shortcut solution*). Przykładem takiego rozwiązania w opisanej metodzie może być rozwiązanie oparte na analizie ciągłości wartości pikseli sąsiadujących fragmentów. Aby temu zapobiec, pomiędzy wybieranymi fragmentami zachowano pewną odległość jak na rys. 6.3. Innym przykładem trywialnego rozwiązania jest przewidywanie relatywnego położenia fragmentów obrazu na podstawie zjawiska aberracji chromatycznej, które jest wynikiem różnic w sposobie załamywania przez obiektyw światła o różnych długościach fali. Efekt ten jest najczęściej obserwowany na zdjęciach wykonanych urządzeniem niskiej klasy i objawia się w kanale koloru zielonego. Zjawisko wpływa na wartości pikseli, tworząc pewien wzorzec zależny od położenia na obrazie. Wzorzec ten może być w prosty sposób wykryty przez sieć, co pozwala na określenie globalnej lokalizacji fragmentu

obrazu, bez konieczności analizy informacji kontekstowej. Aby zapobiec temu trywialnemu rozwiązaniu, autorzy zastosowali specjalne metody przetwarzania obrazów m.in.: losowe usuwanie kanałów koloru.



Rysunek 6.3: Metoda przewidywania relacji pomiędzy fragmentami obrazu

Innym przykładem zadania pomocniczego jest przewidywanie kąta obrotu obrazu [105]. W tym celu wykorzystywana jest sieć, której wejście stanowią obrazy, które mogą być obrócone o 0, 90, 180, lub 270 stopni. Wybór innych kątów mógłby powodować powstawanie artefaktów obrazu, na podstawie których sieć mogłaby w prosty sposób realizować zadanie. Obrót obrazu jest prostą transformacją geometryczną, dlatego wygenerowanie par obraz – odpowiedź oczekiwana, nie wymaga oznaczenia przez człowieka. Aby ocenić kąt obrotu, sieć musi nauczyć się lokalizacji obiektów na obrazie, rozpoznawania orientacji obrazu, typu obiektu i zestawić to z dominującą orientacją, w jakiej obiekt jest przedstawiany na obrazie. Zaletą metody jest brak potrzeby stosowania metod, które miałyby zapobiec trywialnym rozwiązaniom.

Przewidywanie kąta obrotu powoduje uczenie się przez sieć informacji kontekstowej, co powoduje, że sieć zwraca uwagę na kształt obiektów, jednak nie uwzględnia tekstury. Na przykład w zadaniu rozpoznawania tekstur [106] (przykłady tekstur wykorzystanych w pracy przedstawiono na rys. 6.4), sieć wstępnie przetrenowana na zadaniu przewidywania kąta obrotu nie uzyskała lepszych wyników niż sieć zainicjalizowana losowymi parametrami. W generowaniu odpowiedzi przez sieć neuronową używana jest również informacja o teksturze, dlatego rozwiązanie zadania pomocniczego powinno być również oparte o analizę tekstury. W celu

generowania reprezentacji, w której włączona jest informacja o teksturze, autorzy [107] rozszerzyli metodę przewidywania kąta obrotu o przewidywanie stopnia jasności, kontrastu, saturacji, wyostrzenia oraz solaryzacji. Sieć realizowała zadanie klasyfikacji, ponieważ stopnie intensywności każdej modyfikacji zostały pogrupowane w zakresy. Spośród wymienionych modyfikacji najlepsze wyniki przyniosło zadanie oparte o estymację stopnia solaryzacji.

Rysunek 6.4: Przykłady tekstur wykorzystanych w zadaniu rozpoznawania tektur [106] – tkanina, bąbelki, paski, pęknięcia, plaster miodu, szachownica

Autorzy [108] wprowadzili inny sposób nauki reprezentacji z kontekstu. Podobnie jak w metodzie przedstawionej na rys. 6.3 obraz dzielony jest na dziewięć niezachodzących na siebie fragmentów. Następnie, każdy z fragmentów przetwarzany jest przez sieć neuronową w celu generacji wektora reprezentacji. Wygenerowane wektory są łączone w losowy sposób, tworząc wektor stanowiący wejście sieci sieć w pełni połączonej, która przewiduje zastosowaną permutację. Z uwagi na to, że liczba możliwych permutacji dla dziewięciu fragmentów wynosi $9! = 362880$, co wymagałoby zastosowania warstwy wyjściowej z funkcją softmax o rozmiarze 362880, autorzy zredukowali zadanie do 1000 permutacji. W celu zapewnienia jak największej różnorodności permutacji maksymalizowano dystans Hamminga pomiędzy nimi. Dodatkowo, w celu utrudnienia działania sieci i uniknięcia wystąpienia trywialnych rozwiązań, fragmenty oddalano od siebie o średnio dziewięć pikseli. Zastosowano również inne rozwiązania mające na celu uniknięcie trywialnych rozwiązań opartych

na ciągłości koloru. Do każdego fragmentu niezależnie zastosowano augmentację danych.

Inną klasą rozwiązań zadań pretekstowych są algorytmy oparte o generację treści. Jednym z takich zadań jest wypełnienie brakujących fragmentów obrazu [109]. Do realizacji tego zadania zastosowano autoenkoder (sekcja 6.2.3), którego wejściem jest obraz z usuniętym fragmentem, a wyjściem obraz ze zrekonstruowanym fragmentem. W rozwiązaniu zastosowano funkcje celu MSE, odpowiadającą za generowanie przybliżonego zarysu obiektów. Jednakże, zastosowanie tej funkcji powodowało, że otrzymywane obrazy były rozmyte. Aby temu zapobiec, wprowadzono dodatkową funkcję celu typu *adversarial*. Dodanie tej funkcji celu spowodowało, że otrzymywane obrazy były bardziej realistyczne i ostre (rys. 6.5). Aby zastosować funkcję celu typu *adversarial*, do systemu należało dodać dodatkową sieć neuronową zwaną dyskryminatorem, która miała za zadanie rozpoznawać czy wypełnienie pochodzi z rzeczywistego obrazu, czy zostało wygenerowane przez sieć. Część autoenkodera zwana enkoderem została wykorzystana jako ekstraktor cech do innych zadań przetwarzania obrazu takich jak klasyfikacja, segmentacja i detekcja.

Rysunek 6.5: Zadanie rekonstrukcji. Od lewej: obraz poddany rekonstrukcji, rekonstrukcja z wykorzystaniem funkcji celu L_2, rekonstrukcja z wykorzystaniem funkcji celu typu *adversarial*, rekonstrukcja z wykorzystaniem funkcji celu L_2 oraz *adversarial*[109]

Autorzy [110] jako zadanie pomocnicze zaproponowali kolorowanie czarno-białych obrazów. Sieć, aby pokolorować obraz, musi znać treść obrazu oraz umieć lokalizować obiekty. Wygenerowanie danych uczących jest możliwe poprzez prostą operację zamiany obrazów kolorowych na czarno-białe. W proponowanym rozwiązaniu sieć neuronowa dla każdej pozycji na obrazie generuje histogram kolorów. Umożliwia to reprezentowanie wielu kolorów obiektów występujących w różnych kolorach (rys. 6.6).

Inną pracą, w której zastosowano zadanie kolorowania, jest [111]. Zadaniem sieci było generowanie koloru w przestrzeni barw CIELab. Cechą widoczną w zbiorach

Rysunek 6.6: Wykorzystanie histogramów do kolorowania obrazów [110]

danych jest częstsze występowanie kolorów nienasyconych (np. nieba) niż nasyconych. Prowadzi to do problemu niejednorodnego rozkładu kolorów, co utrudnia zadanie kolorowania. W celu redukcji tego problemu wprowadzono ważoną funkcję celu, która przypisywała wyższe wagi rzadziej występującym kolorom .

6.2.2 Metody oparte o kontrastową funkcję celu

Metody kontrastowe opierają się na następującej zasadzie: minimalizowany jest dystans pomiędzy reprezentacjami różnych wersji tego samego obrazu (nazywanymi pozytywnymi parami) oraz maksymalizowany jest dystans pomiędzy reprezentacjami różnych obrazów ze zbioru danych (nazywanymi parami negatywnymi). Pary pozytywne generowane są poprzez zastosowanie metod rozszerzania danych. W rozwiązaniach stosowana jest następująca funkcja celu (lub jej warianty):

$$\mathcal{L}_{NCE} = -\log\left(\frac{\exp(s(\mathbf{v_I}, \mathbf{v_{I^t}})/\tau)}{\exp(s(\mathbf{v_I}, \mathbf{v_{I^t}})/\tau) + \sum_{I' \in \mathcal{D}_N} \exp(s(\mathbf{v_{I^t}}, \mathbf{v_{I'}})/\tau)}\right) \tag{6.1}$$

gdzie:

- $\mathbf{v_I}, \mathbf{v_{I^t}}$ – reprezentacja obrazu i transformacji obrazu,
- $s(\cdot)$ – miara podobieństwa reprezentacji,
- τ – współczynnik temperatury,
- I' – zbiór przykładów negatywnych wylosowany ze zbioru danych,
- $\mathbf{v_{I'}}$ – reprezentacja przykładów negatywnych,
- D_N – zbiór danych.

Minimalizacja funkcji powoduje, że model ignoruje nieistotne informacje w obrazie związane m.in.: z kolorem i pozycją obiektu, a zwraca uwagę między innymi na kontekst czy teksturę [112].

Liczba przykładów negatywny wykorzystywanych w opisanej funkcji celu ma istotny wpływ na jakość generowanych reprezentacji. W literaturze opisywane są

różne metody generowania i przechowywania reprezentacji przykładów negatywnych. Jednym z rozwiązań jest stosowanie obrazów negatywnych z aktualnego wsadu (ang. *batch*) [21]. Jednak, w tej metodzie rozmiar wsadu ogranicza rozmiar dostępnej pamięci GPU. Innym rozwiązaniem jest zastosowanie pamięci, przechowującej reprezentacje wszystkich obrazów w zbiorze uczącym [22], [113]. Oprócz tych rozwiązań stosowane są również inne metody przechowywania i generowania reprezentacji przykładów negatywnych [114], [115].

W metodzie MOCO (ang. *MOmentum COntrast*) [114] zastosowano dynamiczną pamięć przechowującą reprezentacje obrazów. Zastosowanie pamięci umożliwia uniezależnienie działania metody od rozmiaru wsadu. Obrazy z aktualnego wsadu są kodowane przez dwie sieci f_q oraz f_k. Sieć f_q trenowana jest przy użyciu mechanizmu propagacji wstecznej, a parametry sieci f_k wyznaczane są na podstawie parametrów f_q:

$$\boldsymbol{\theta}_k = m\boldsymbol{\theta}_k + (1-m)\boldsymbol{\theta}_q \tag{6.2}$$

gdzie: $\boldsymbol{\theta}_q$ i $\boldsymbol{\theta}_k$ to parametry sieci f_q i f_k, a $m \in [0, 1)$ to współczynnik aktualizacji. Sieć f_k służy do generowania reprezentacji, które zapisywane są w kolejce FIFO (ang. *First In First Out*), przechowującej przykłady negatywne. Generowanie par pozytywnych odbywa się poprzez zastosowanie następujących metod: losowe wycinanie, losowe zaburzenie koloru (ang. *color jittering*), losowe odbicia w pionie i poziomie, losowa zamiana przestrzeni koloru na przestrzeń czarno-białą.

Autorzy metody SimCLR [21] odkryli, że w metodach kontrastowych należy stosować rozszerzanie danych o większej intensywności niż w przypadku uczenia nadzorowanego. Zauważyli również, że wprowadzenie nieliniowych warstw transformujących reprezentacje generowane przez sieć, do reprezentacji przyjmowanej przez funkcję celu powoduje poprawę wyników. Autorzy przeanalizowali również relacje pomiędzy jakością uzyskiwanych reprezentacji a rozmiarem wsadu, odkrywając, że większy rozmiar wsadu prowadzi do lepszych wyników. Przy analizie stosowanych metod augmentacji danych odkryto, że jeden sposób modyfikacji obrazów jest niewystarczający do nauki dobrych reprezentacji. Na przykład użycie metody wycinania powodowało, że rozwiązanie opierało się o trywialne rozwiązania polegające na porównaniu kolorów wyciętych fragmentów. Po testach i analizach zastosowano trzy typy rozszerzania danych: losowe wycinanie fragmentu obrazu, losowe zaburzenie koloru oraz losowe rozmycie Gaussa.

Metoda MOCO [114] została ulepszona poprzez wykorzystanie rozwiązań zastosowanych w SimCLR [21]. Zaproponowane rozwiązanie nosi nazwę MOCOv2 [115]. Podobnie jak w metodzie SimCLR wprowadzono dwuwarstwową sieć neuronową o nieliniowej funkcji aktywacji ReLU, transformującą reprezentację do przestrzeni funkcji celu. Zastosowano także takie same metody rozszerzania danych: losowe wycinanie fragmentów, losowe zaburzenia koloru oraz rozmycie Gaussa. W trakcie treningu zastosowano kosinusową zmianę współczynnika uczenia.

Autorzy metody [22] zastosowali pamięć przechowującą eksponencjalną średnią kroczącą generowanych reprezentacji. W stosowanej funkcji celu przykłady negatywne pochodziły z zastosowanej pamięci. Szczegółowy opis metody znajduje się w sekcji 6.3.

W [113] odkryto, że zwiększenie liczby przykładów pozytywnych korzystnie wpływa na jakość generowanych reprezentacji. Zbadano dwie funkcje celu uwzględniające większą liczbę przykładów pozytywnych. Jedna z funkcji celu porównywała wszystkie reprezentacje z_j do reprezentacji z_1:

$$\mathcal{L} = \sum_{j=2}^{M} \mathcal{L}_{NCE}(z_1, z_j) \tag{6.3}$$

Drugą metodą, bardziej kosztowną obliczeniowo, było porównywanie reprezentacji sposobem „każdy z każdym”:

$$\mathcal{L} = \sum_{1 \leqslant i \leqslant j \leq M} \mathcal{L}_{NCE}(z_i, z_j) \tag{6.4}$$

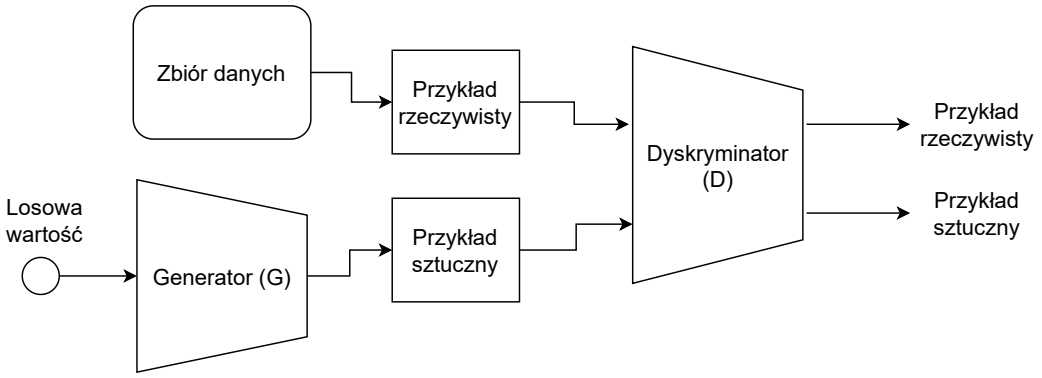
W eksperymentach nie wykryto znacznych różnic pomiędzy dwoma sposobami.

6.2.3 Inne metody stosowane w celu uczenia reprezentacji

W uczeniu reprezentacji stosuje się również inne metody [116]. Jednym z podejść jest zastosowanie autoenkoderów, które znalazły zastosowanie m.in. w zadaniach redukcji wymiarowości. Autoenkoder składa się z dwóch części: kodera, który transformuje dane wejściowe x w określoną reprezentację $z = f(x)$ oraz dekodera, który stara się odtworzyć x na podstawie reprezentacji z tj. $x^{'} = g(z)$. Funkcję realizowaną przez autoenkoder można zatem zapisać w postaci $g(f(x))$. W metodach przetwarzania obrazów, uczenie autoenkodera polega na optymalizacji funkcji celu MSE, która reprezentuje podobieństwo pomiędzy obrazem wejściowym x a jego

rekonstrukcją $x^{'}$. Rozmiar wektora z, który jest dużo mniejszy niż rozmiar wejścia x, powoduje, że autoenkoder powinien uwzględnić w reprezentacji z informacje istotne w odtworzeniu $x^{'}$. Koder $f(\cdot)$ uczy się użytecznych reprezentacji, dzięki czemu może być później wykorzystany jako ekstraktor cech w zadaniach docelowych.

Innym sposobem uczenia reprezentacji bez stosowania danych oznaczonych jest zastosowanie i trening sieci GAN (ang. *Generative Adversarial Network*) [117]–[119]. Zadaniem jest generowanie danych, które powinny przypominać dane rzeczywiste (np. obrazy). Trening składa się z uczenia dwóch sieci: generatora G generującego dane oraz dyskryminatora D, którego zadaniem jest określenie prawdopodobieństwa, że próbka pochodzi od dyskryminatora D (rys. 6.7). Funkcja celu składa się z dwóch członów: pierwszy odpowiada za trening sieci G w taki sposób, aby generowała jak najbardziej realistyczne obrazy i wprowadzała tym samym dyskryminator D w błąd. Drugi człon odpowiada za naukę dyskryminatora D, który ma za zadanie odróżnianie przykładów rzeczywistych od tych wygenerowanych przez generator. Dyskryminator D aby odróżnić dane sztuczne od rzeczywistych musi mieć umiejętność ekstrakcji cech. Dlatego, dyskryminator można zastosować jako ekstraktor cech w zadaniach docelowych.



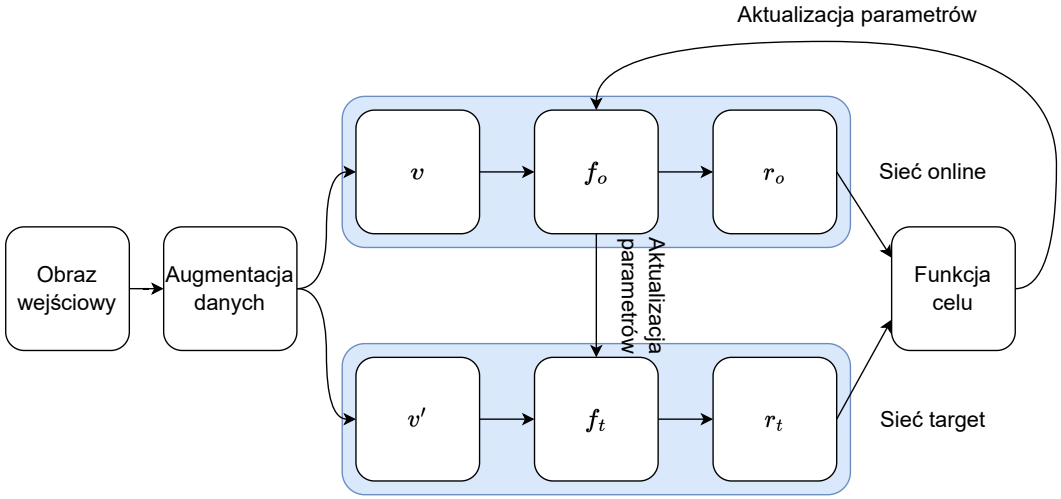
Rysunek 6.7: Schemat działania sieci GAN

Metoda BYOL [23] (ang. *Bootstrap Your Own Latent*) polega na treningu dwóch sieci *online* oraz *target*. Wejście sieci stanowią dwa obrazy poddane różnym operacjom augmentacji danych. Celem sieci *online* jest wygenerowanie reprezentacji, która jest identyczna z reprezentacją wygenerowaną przez sieć *target*. Wyjście sieci *target* można traktować jako oznaczenie obrazu, które musi zostać odtworzone przez sieć *online* (rys. 6.8). Parametry $\boldsymbol{\theta}_o$ sieci *online* dobierane są w standardowy sposób w procesie optymalizacji gradientowej. Natomiast parametry $\boldsymbol{\theta}_t$ sieci *target* są

eksponencjalną średnią kroczącą parametrów $\boldsymbol{\theta}_o$, zgodnie ze wzorem:

$$\boldsymbol{\theta}_{t,n+1} = a\boldsymbol{\theta}_{t,n} + (1-a)\boldsymbol{\theta}_{o,n} \tag{6.5}$$

W metodzie zastosowano te same metody augmentacji danych, co w pracy [21].



Rysunek 6.8: Schemat działania metody BYOL

6.3. Uczenie dwuetapowe w zadaniu klasyfikacji znamion skórnych

Wnioski wyciągnięte z opisanych wcześniej badań zrealizowanych w ramach doktoratu oraz analiza obecnie stosowanych metod uczenia skłoniły autora do podjęcia badań nad zastosowaniem podejścia wykorzystującego uczenie samonadzorowane do zadania klasyfikacji znamion skórnych.

Wstępne analizy i przegląd literatury wykazały, że najlepszą metodą generowania reprezentacji będzie metoda PIRL (*Pretext Invariant Representation Learning*) [22]. W czasie prowadzenia eksperymentów, była to jedna z nowocześniejszych metod uczenia nadzorowanego, która prowadziła do dobrych wyników na zadaniach docelowych (m.in. ImageNet). Metoda PIRL przetestowana została na dużych zbiorach danych, jednak autorzy metody nie raportowali skuteczności przy zastosowaniu małych zbiorów danych. W badaniach prowadzonych w trakcie doktoratu postanowiono przetestować działanie metody do wstępnego uczenia sieci

z wykorzystaniem niewielkiego zbioru danych. Metodę PIRL wybrano ze względu na jej zalety, m.in. bank pamięci, który uniezależnia skuteczność od rozmiaru pamięci karty graficznej, co obniża koszty i sprawdza się w zadaniach małej skali.

Podczas wstępnego treningu sieci neuronowej, metoda PIRL ma na celu osiągnięcie niezależności reprezentacji obrazów od zastosowanych transformacji obrazu, co jest przedstawione na rys. 6.9. W celu zbliżenia podobieństwa reprezentacji tego samego obrazu poddanego różnym modyfikacjom podczas treningu wykorzystano kontrastową funkcję celu.

W przeprowadzonych eksperymentach wykorzystano zbiór znamion skórnych ISIC2017, który jest znacznie mniejszy niż zbiór użyty w eksperymentach opisanych w rozdziałach 4 i 5. Zbiór ISIC2017 wykorzystywany jest przez społeczność naukową, co umożliwia porównywanie rozwiązań z innymi pracami. Zbiór zawiera 2000 obrazów treningowych, 150 obrazów walidacyjnych oraz 600 obrazów testowych, a każdy zbiór zawiera dwie klasy – zmiana nowotworowa i zmiana łagodna. Rozkład etykiet w każdym zbiorze jest zbliżony – 80% stanowią obrazy ze zmianami łagodnymi, a 20% ze zmianami nowotworowymi.

6.3.1 Funkcja celu

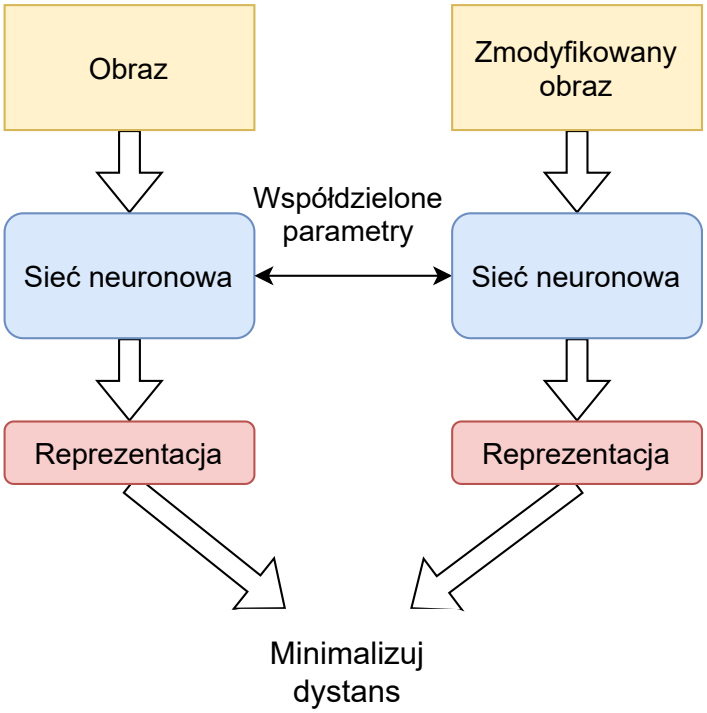
Przyjęta funkcja celu uwzględnia podobieństwo pomiędzy reprezentacją obrazu a reprezentacją jego modyfikacji (przykłady pozytywne reprezentowane w liczniku) oraz podobieństwo pomiędzy reprezentacją obrazu a reprezentacjami innych obrazów występujących w zbiorze (przykłady negatywne reprezentowane w mianowniku). W metodzie PIRL podobieństwo reprezentacji wyrażone jest poprzez dystans kosinusowy. Funkcja celu użyta w eksperymentach przyjmuje następującą postać:

$$\mathcal{L}_{NCE} = -\log\left(\frac{\exp(s(\mathbf{v}_{\mathbf{I}}, \mathbf{v}_{\mathbf{I}^t})/\tau)}{\exp(s(\mathbf{v}_{\mathbf{I}}, \mathbf{v}_{\mathbf{I}^t})/\tau) + \sum_{I' \in \mathcal{D}_N} \exp(s(\mathbf{v}_{\mathbf{I}^t}, \mathbf{v}_{\mathbf{I}'})/\tau)}\right) \tag{6.6}$$

gdzie:

- $\mathbf{v}_{\mathbf{I}}, \mathbf{v}_{\mathbf{I}^t}$ – reprezentacja obrazu i transformacji obrazu
- $s(\cdot)$ – podobieństwo kosinusowe,
- τ – współczynnik temperatury,
- I' – zbiór przykładów negatywnych wylosowany ze zbioru danych,
- $\mathbf{v}_{\mathbf{I}'}$ – reprezentacja przykładów negatywnych,
- D_N – zbiór danych.

Minimalizacja funkcji celu prowadzi do zwiększania podobieństwa reprezentacji obrazów pozytywnych (licznik) i zmniejszania podobieństwa pomiędzy reprezentacją obrazu a reprezentacją innych obrazów w zbiorze (mianownik). Powoduje to, że sieć neuronowa mapuje reprezentacje tych samych obrazów do zbliżonych obszarów w przestrzeni.



Rysunek 6.9: Schemat działania metody PIRL [22]

Wcześniejsze badania dotyczące uczenia kontrastowego wykazały, że liczba przykładów negatywnych w zbiorze I' ma wpływ na jakość otrzymywanych reprezentacji. Przeprowadzone eksperymenty pokazały, że zwiększenie liczby przykładów negatywnych prowadzi do wzrostu dokładności mierzonej na zadaniu docelowym [120], [121]. W metodzie PIRL autorzy zaproponowali utworzenie pamięci reprezentacji, przechowującej reprezentacje każdego obrazu w zbiorze danych, co zmniejszyło wymagania obliczeniowe i zapotrzebowanie na pamięć. Reprezentacje przechowywane w pamięci są eksponencjalną średnią kroczącą reprezentacji generowanych w kolejnych epokach.

Dzięki wprowadzeniu pamięci reprezentacji możliwe było zwiększenie liczby porównań reprezentacji. Między innymi możliwe się stało porównanie reprezentacji

obrazu do reprezentacji tego obrazu przechowywanej w pamięci. W konsekwencji funkcja celu przybrała następującą postać:

$$\mathcal{L}(I, I^t) = \lambda \mathcal{L}_{NCE}(\mathbf{m_I}, \mathbf{v_{I^t}}) + (1 - \lambda)\mathcal{L}_{NCE}(\mathbf{m_I}, \mathbf{v_I}) \tag{6.7}$$

gdzie:

I, I^t – obraz i transformacja obrazu,
$\mathbf{v_I}, \mathbf{v_{I^t}}$ – reprezentacja obrazu i transformacji obrazu,
$\mathbf{m_I}$ – reprezentacja przechowywana w pamięci,
λ – współczynnik funkcji celu.

Przedstawiona funkcja celu stanowi wypukłą kombinację miar podobieństwa. Pierwszy człon funkcji odpowiada za podobieństwo reprezentacji przechowywanej w pamięci oraz reprezentacji obrazu zmodyfikowanego, uwzględniając również podobieństwo do innych obrazów z pamięci. Człon jest analogiczny do oryginalnej funkcji celu, z tą różnicą, że zamiast reprezentacji oryginalnego obrazu generowanej przez sieć neuronową wykorzystuje się reprezentację przechowywaną w pamięci.

Drugi człon funkcji celu porównuje reprezentację oryginalnego obrazu przechowywanego w pamięci z reprezentacją wygenerowaną przez sieć w aktualnej iteracji. Człon stabilizuje trening i zapobiega zbyt szybkim zmianom parametrów w sieci neuronowej.

Współczynnik λ kontroluje wpływ obu członów wartość funkcji celu. W eksperymentach przeprowadzonych w [22] wykazano, że najlepsze wyniki uzyskuje się, gdy wartość współczynnika λ wynosi 0,5.

6.3.2 Szczegóły zaproponowanego podejścia

W rozwiązaniu zrealizowanym w ramach doktoratu zastosowano dwa typy transformacji: obroty [105] oraz puzzle [108], opisane w artykule przedstawiającym metodę PIRL [22]. Każda z tych transformacji wymaga przygotowania obrazu I oraz obrazu zmodyfikowanego I'. Szczegółowe opisy tych transformacji przedstawione zostaną w kolejnych sekcjach pracy.

W celu ewaluacji opisywanego rozwiązania, jako ekstraktor cech zastosowano architekturę ResNet50 [13]. Ta architektura wykazała się skutecznością w wielu zastosowaniach, w tym w zastosowaniach medycznych. Ekstraktor cech dla każdego obrazu generuje reprezentację w postaci 2048-wymiarowego wektora. Następnie,

poprzez warstwy w pełni połączone, rozmiar wektora jest redukowany do 128 elementów. Sposób redukcji rozmiaru różni się w zależności od wybranego zadania pomocniczego.

6.3.3 Wstępne przetwarzanie danych

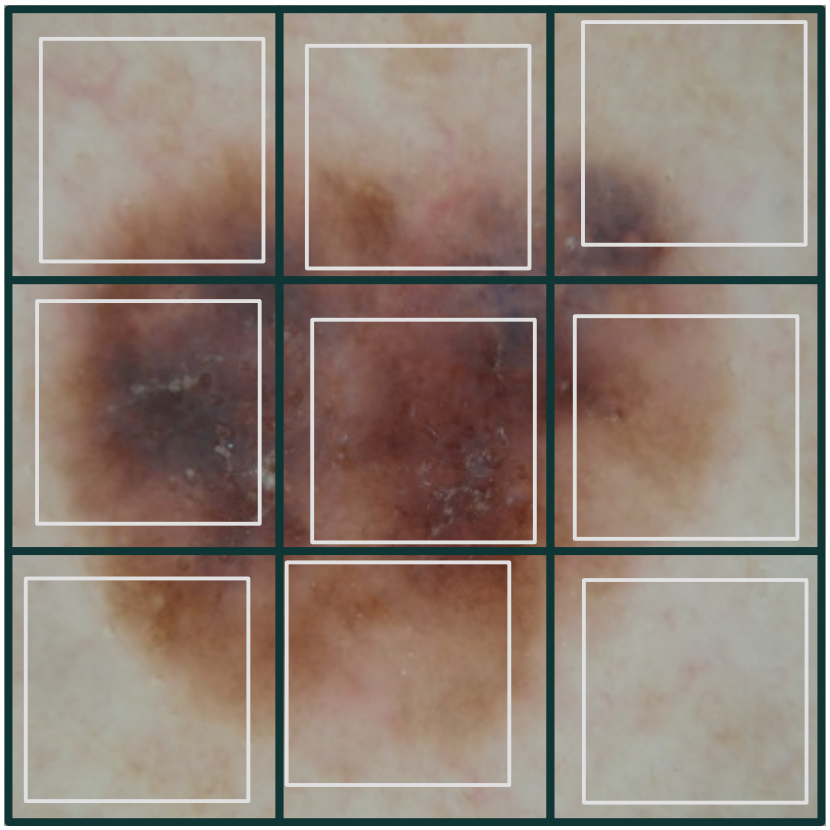
Obrazy w zbiorze danych różnią się rozmiarem oraz proporcjami krawędzi. W celu wycięcia znamion ze zdjęcia zastosowano maski dostępne w zbiorze danych. Sposób wycinania jest taki sam jak w poprzednich rozdziałach, z tą różnicą, że wycięty fragment zostaje przeskalowany do rozmiaru 400×400 pikseli. Dzięki temu zachowywane są proporcje znamion, co może mieć wpływ na wyniki klasyfikacji, ponieważ proporcja znamion jest jednym ze znaczących kryteriów oceny złośliwości.

6.3.4 Zadanie Puzzle

Zadanie Puzzle polega na wygenerowaniu reprezentacji oryginalnego obrazu I oraz reprezentacji I' powstałej z połączenia reprezentacji dziewięciu fragmentów oryginalnego obrazu. Wynik generowania dziewięciu fragmentów przedstawiony jest na rys. 6.10 i został dokonany w następujący sposób. Pierwszym krokiem jest zmniejszenie obrazu z 400×400 do 255×255 pikseli. Następnie obraz dzielony jest w siatkę 3×3, zawierającą dziewięć pól o rozmiarze 85×85 pikseli. Z każdego z dziewięciu pól losowo wycinany jest fragment o rozmiarze 64×64 pikseli. Każdy z dziewięciu fragmentów poddawany jest metodom rozszerzania danych, wykorzystując różne parametry modyfikacji. Do stosowanych metod należą: zaburzenie koloru oraz odbicie pionowe i poziome. Na końcu każdy fragment poddawany jest normalizacji sprowadzającej średnią wartości pikseli obrazu do zera, a odchylenie standardowe do jeden.

Obraz oryginalny generowany jest w sposób następujący. Najpierw rozmiar przetworzonego obrazu zmniejszany jest z 400×400 pikseli do 224×224 pikseli. Następnie stosowane są metody rozszerzania danych tj. odbicia w pionie i poziomie oraz losowe zaburzenie koloru. Na końcu wynikowy obraz poddawany jest normalizacji.

Reprezentacja obrazu I generowana jest poprzez warstwę w pełni połączoną, zawierającą 128 neuronów realizujących funkcję ReLU. Wejściem tej warstwy jest 2048-elementowy wektor wygenerowany przez ekstraktor cech sieci ResNet50. Generacja reprezentacji fragmentów I^t jest bardziej złożona. W celu generowania reprezentacji dziewięciu fragmentów stosowana jest sieć neuronowa zakończona

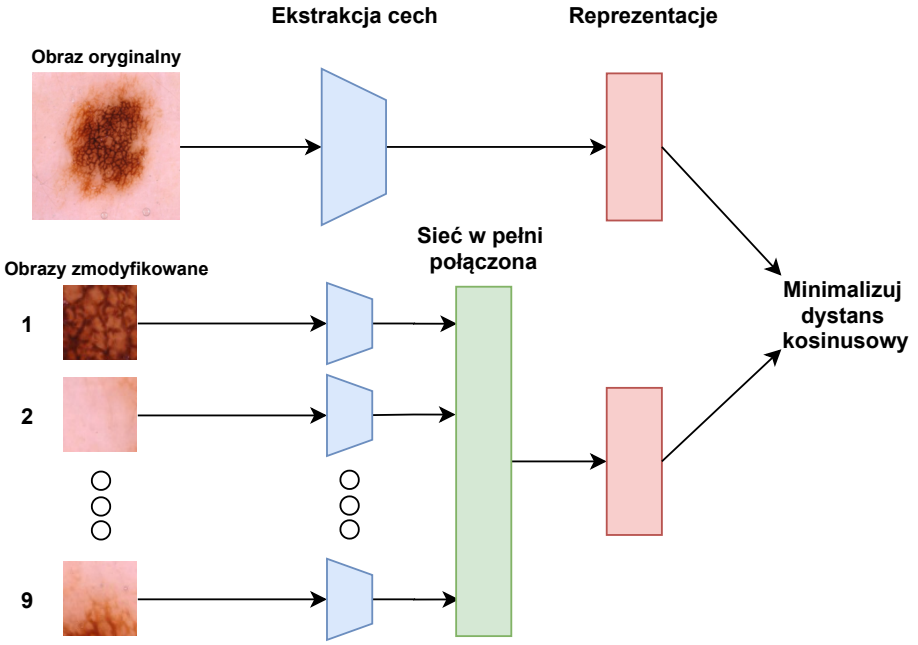


Rysunek 6.10: Sposób generowania dziewięciu fragmentów

warstwą w pełni połączoną, która generuje 128-elementowy dla każdego fragmentu. Następnie wektory są ze sobą łączone w losowej kolejności, tworząc wektor zawierający 1152 elementy. Wektor ten stanowi wejście sieci w pełni połączonej o funkcji aktywacji ReLU, generującej 128-elementowego wektor, będący reprezentacją I' stosowaną w funkcji celu. Sposób generowania reprezentacji I oraz I' został przedstawiony na rys. 6.11.

6.3.5 Zadanie Obroty

Generacja reprezentacji obrazu I oraz reprezentacji transformacji obrazu I' jest mniej skomplikowana niż w zadaniu Puzzle. Wstępnie przetworzony obraz zmniejszany jest do rozmiaru 300×300 pikseli. Następnie, z obrazu losowo wycinane są dwa fragmenty o rozmiarze 224×224 pikseli. Do każdego obrazu zastosowano losowe zaburzenia koloru oraz normalizację. Następnie, obraz I' obracany jest o losowo wybrany kąt (0, 90, 180 lub 270 stopni). Obrazy te stanowią wejście sieci ResNet50, której wyjście stanowi 2048-elementowa reprezentacja. Aby uzyskać 128-elementową reprezentację, stosowana jest warstwa w pełni połączona, której



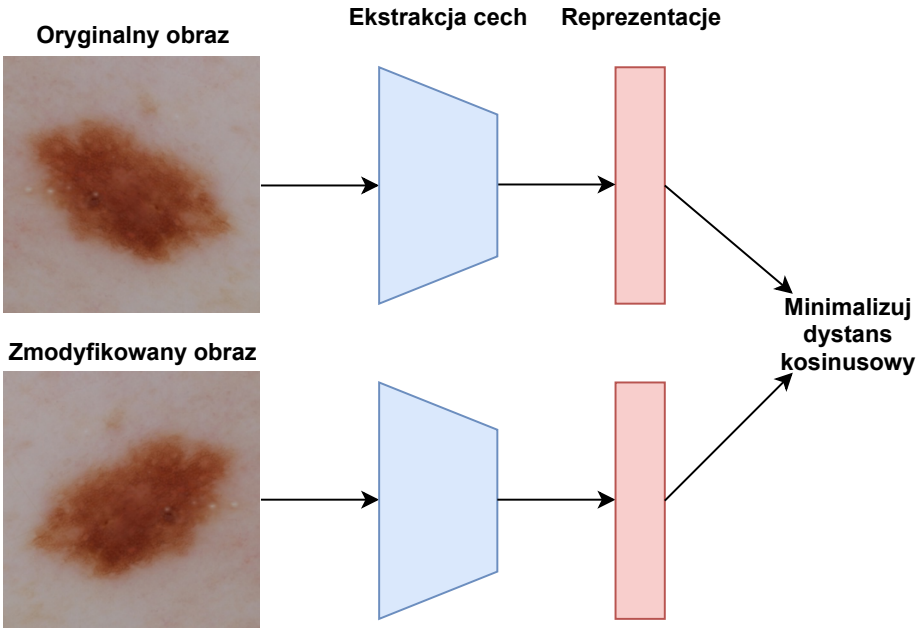
Rysunek 6.11: Generowanie 128-elementowych wektorów metodą Puzzle

wejście stanowi reprezentacja generowana przez sieć ResNet50. Schemat działania zadania Obroty przedstawiony jest na rys. 6.12.

6.3.6 Wstępne uczenie

W procesie wstępnego uczenia zastosowano kontrastową funkcję celu przedstawioną w równaniu 6.7. Użyto optymalizatora SGD z momentem Nesterova, przyjęto współczynnik uczenia równy 0,001, współczynnik momentum 0,9, a także zastosowano wsad o rozmiarze 32 obrazów. Do każdego obrazu losowano 1000 przykładów negatywnych pochodzących z pamięci reprezentacji. Na podstawie analizy literatury, przyjęto współczynnik temperatury w funkcji celu wynoszący 0,07, co jest wartością często stosowaną w innych rozwiązaniach m.in. PIRL [22], SimCLR [21]. Zastosowano współczynnik funkcji celu λ równy 0,5. Zgodnie z wynikami raportowanymi w literaturze, zwiększenie długości treningu prowadzi do uzyskania lepszej jakości reprezentacji, dlatego zdecydowano się przeprowadzić trening trwający 1000 epok.

Reprezentacje obrazów przechowywane są w pamięci w postaci eksponencjalnej średniej kroczącej, w której współczynnik aktualizacji przyjęto na poziomie 0,5. Dzięki temu, reprezentacje uzyskane w ostatnich epokach mają silniejszy wpływ na aktualną wartość reprezentacji przechowywanej w pamięci. Przed rozpoczęciem



Rysunek 6.12: Generowanie reprezentacji metodą Obroty

treningu, w pamięci zapisuje się reprezentacje wygenerowane przez sieć z losowo zainicjalizowanymi parametrami.

6.3.7 Zadanie docelowe

W treningu zadania docelowego jako punkt startowy zastosowano parametry sieci uzyskane podczas wstępnego treningu. Warstwy służące do generowania 128-elementowych reprezentacji usunięto ze struktury. Następnie, na końcu sieci neuronowej umieszczono neuron sigmoidalny, którego wejściem jest 2048-elementowy wektor reprezentacji generowany przez sieć ResNet. W treningu wykorzystano binarną krosentropię krzyżową jako funkcję celu. Zastosowano optymalizator SGD z momentem Nesterova, gdzie przyjęto współczynnik uczenia 0,001, współczynnik momentu 0,9, a rozmiar wsadu 32. Zastosowano metodę wczesnego zatrzymania w celu ograniczenia nadmiernego dopasowania. W eksperymentach nie stosowano metod augmentacji danych.

6.4. Eksperymenty i wyniki

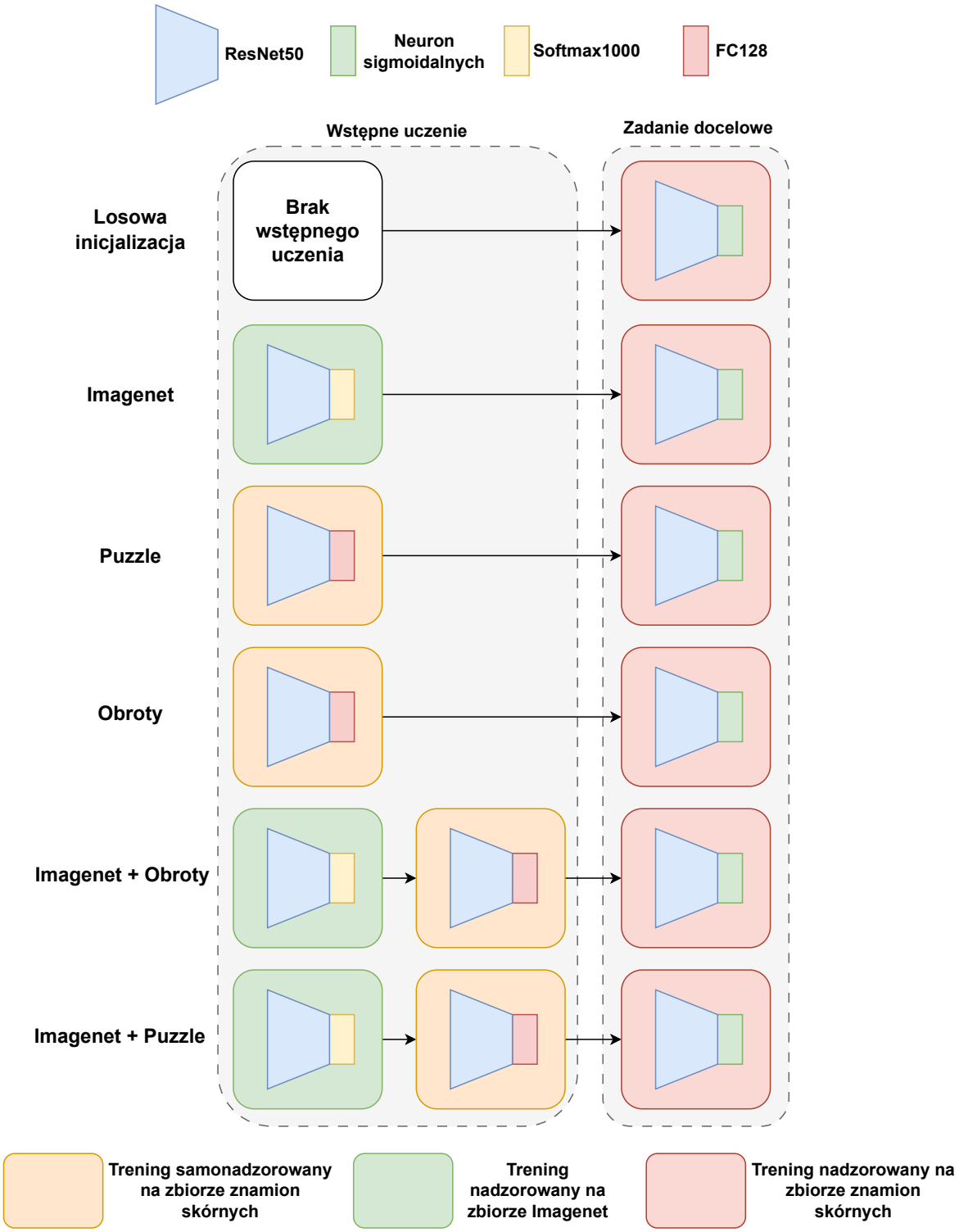
W celu ewaluacji zaproponowanej metody przeprowadzono serię eksperymentów, w których porównano sześć sposobów wstępnego uczenia. Jako sieć odniesienia

przyjęto sieć z losowo zainicjalizowanymi parametrami oraz sieć wstępnie przeuczoną na zadaniu ImageNet. Następnie, przetestowano dwie metody wstępnego uczenia tj. Puzzle oraz Obroty przy zastosowaniu zbioru treningowego ISIC2017. Dodatkowo zaproponowano sekwencyjne wstępne uczenie, w którym sieć była wstępnie uczona na zbiorze ImageNet, a następnie z zastosowaniem uczenia samonadzorowanego na zbiorze ISIC2017. Przeprowadzono analizę dwóch konfiguracji oznaczonych w tabelach jako ImageNet + Puzzle oraz ImageNet + Obroty. Schemat eksperymentów przedstawiono na rys. 6.13.

W celu oceny zapotrzebowania sieci na dane uczące zaplanowano eksperymenty, w których przeprowadzono uczenie na zadaniu docelowym stosując ograniczony zbiór uczący. Trening przeprowadzono na pełnym zbiorze treningowym (2000 obrazów), 10% zbioru treningowego (200 obrazów) oraz na 1% zbioru treningowego (20 obrazów). W celu dodatkowej oceny jakości reprezentacji dokonano ewaluacji w trybie liniowego protokołu ewaluacji (ang. *linear evaluation protocol*), który polega na treningu modelu regresji logistycznej na wyuczonych reprezentacjach. Oprócz tego testom poddano trzy wybrane klasyfikatory: k najbliższych sąsiadów (kNN), las losowy oraz SVM. Końcowym etapem eksperymentów był trening z wykorzystaniem augmentacji danych.

W celu kompleksowej walidacji wyników każdy trening na zadaniu docelowym powtórzono pięć razy, a następnie wyniki uśredniono. Jako główną miarę przyjęto powierzchnię pod krzywą ROC, stosowaną w zadaniu ISIC 2017. Oprócz tego, w tabelach przedstawiono wartości dokładności oraz miary PR AUC (ang. *(Precision-Recall Area Under Curve)*.

Wyniki treningu na pełnym zbiorze treningowym przedstawione są w tabeli 6.1. Jak się spodziewano, najgorsze wyniki uzyskała sieć z losowo zainicjalizowanymi parametrami. Zastosowanie wstępnego uczenia na zbiorze ImageNet w sposób nadzorowany umożliwia znaczną poprawę wszystkich raportowanych miar skuteczności, względem losowo zainicjalizowanej sieci. Jest to zgodne z innymi wynikami raportowanymi w literaturze, gdzie zazwyczaj metoda transfer learning umożliwia uzyskanie lepszych rezultatów. Zastosowanie wstępnego uczenia w sposób nienadzorowany (oznaczone w tabeli jako Puzzle oraz Obroty) również prowadzi do lepszych wyników niż w przypadku losowej inicjalizacji parametrów, jednak wyniki są gorsze od wstępnego treningu w sposób nadzorowany na zbiorze ImageNet. Wśród wszystkich rozpatrywanych metod, najlepsze rezultaty przynosi użycie sekwencyjnego



Rysunek 6.13: Schemat analizowanych podejść wstępnego uczenia

wstępnego uczenia, najpierw w sposób nadzorowany (ImageNet), a następnie samonadzorowany (Puzzle oraz Obroty). Najlepszy wynik, wśród wszystkich przeprowadzonych eksperymentów udało się osiągnąć poprzez zastosowanie wstępnego uczenia nadzorowanego (ImageNet), a następnie wstępne uczenia na zadaniu Puzzle. Osiągnięto wartość AUC ROC równą 0,83 dla zadania docelowego, co jest wynikiem wyższym niż wynik osiągnięty przy użyciu sieci wstępnie uczonej tylko na zadaniu nadzorowanym (0,791).

Tabela 6.1: Wyniki treningu na pełnym zbiorze treningowym bez użycia augmentacji danych

Metoda wstępnego uczenia	**ROC AUC**	**PR AUC**	**ACC**
Losowa inicjalizacja	0,598	0,257	79,67
ImageNet	0,791	0,525	83,03
Puzzle	0,664	0,321	79,80
Obroty	0,732	0,370	80,30
ImageNet + Puzzle	**0,830**	**0,595**	**84,47**
ImageNet + Obroty	0,826	0,591	84,00

Aby ocenić zapotrzebowanie na dane uczące w zadaniu docelowym zaplanowano eksperymenty z zastosowaniem tylko części zbioru treningowego. W tym celu przeprowadzono uczenie na 1% (20 obrazów) i 10% obrazów (200 obrazów), które zostały dobrane losowo. Trening z 20 obrazami został przeprowadzony ze wsadem (ang. *batch size*) o rozmiarze 4. Rozmiar zbioru walidacyjnego i testowego pozostał taki sam jak w eksperymentach przeprowadzonych na pełnym zbiorze. Rezultaty przeprowadzonych eksperymentów przedstawione są w tabelach 6.2 i 6.3.

We wszystkich przeprowadzonych eksperymentach najlepsze rezultaty osiągnęła metoda sekwencyjnego wstępnego treningu. Jednak tym razem, lepsza okazała się metoda ImageNet + Obroty. Na uwagę zasługuje fakt, że trening sieci na 10% zbioru treningowego, w której zastosowano wstępny trening w sposób sekwencyjny, daje porównywalne wyniki do sieci uczonej na 100% danych, wstępnie przeuczonej w sposób nadzorowany. Świadczy to o tym, że dodatkowa faza wstępnego uczenia polegająca na uczeniu w sposób nadzorowany jest szczególnie istotna w zadaniach, w których dostępny zbiór danych jest niewielki.

Tabela 6.2: Wyniki treningu na 20 obrazach (1% zbioru uczącego).

Metoda wstępnego uczenia	**ROC AUC**	**PR AUC**	**ACC**
Losowa inicjalizacja	0,555	0,230	79,30
ImageNet	0,519	0,201	80,70
Puzzle	0,599	0,252	79,33
Obroty	0,658	0,310	79,53
ImageNet + Puzzle	0,666	0,327	80,53
ImageNet + Obroty	**0,755**	**0,395**	**81,50**

Tabela 6.3: Wyniki treningu na 200 obrazach (10% zbioru uczącego)

Metoda wstępnego uczenia	**ROC AUC**	**PR AUC**	**ACC**
Losowa inicjalizacja	0,493	0,201	79,30
ImageNet	0,734	0,405	80,70
Puzzle	0,638	0,265	79,33
Obroty	0,698	0,361	79,53
ImageNet + Puzzle	0,753	0,416	80,53
ImageNet + Obroty	**0,780**	**0,458**	**81,50**

W celu dodatkowej oceny generowanych reprezentacji zastosowano liniowy protokół ewaluacji (ang. *linear evaluation protocol*). W tym celu przeprowadzono trening 4 klasyfikatorów, stosując reprezentacje wygenerowane przez sieci neuronowe, poddane wstępnemu uczeniu stosując 5 opisanych wcześniej metod oraz przez sieć zainicjalizowaną losowo. Należy zwrócić uwagę na to, że w przeciwieństwie do poprzednich eksperymentów, parametry ekstraktora cech nie są dobierane w trakcie uczenia. W ramach eksperymentów wykorzystano 2048-elementowe reprezentacje wygenerowane przez sieć neuronową ResNet50. Analizie poddano działanie czterech szeroko stosowanych klasyfikatorów: maszyn wektorów nośnych (SVM) z radialną funkcją bazową, k-najbliższych sąsiadów (k-NN) z czterema sąsiadami, las losowy z maksymalną liczbą drzew wynoszącą 50 oraz algorytm regresji logistycznej. Wyniki ewaluacji przedstawione są w tabeli 6.4.

Otrzymane wyniki świadczą o dużej jakości generowanych reprezentacji. Można zauważyć, że każda metoda wstępnego uczenia prowadzi do poprawy rezultatów.

Tabela 6.4: Ocena jakości reprezentacji – liniowy protokół ewaluacji

Klasyfikator	**Metoda wstępnego uczenia**	**ROC AUC**	**PR AUC**	**ACC**
k-NN	Losowa inicjalizacja	0,582	0,244	0,785
	Imagenet	0,700	0,388	0,778
	Puzzle	0,600	0,243	0,752
	Obroty	0,676	0,314	0,768
	Imagenet + Puzzle	0,680	0,420	0,808
	Imagenet + Obroty	**0,761**	**0,482**	**0,815**
Regresja logistyczna	Losowa inicjalizacja	0,597	0,247	0,778
	Imagenet	0,748	0,434	0,793
	Puzzle	0,678	0,318	0,765
	Obroty	0,704	0,359	0,792
	Imagenet + Puzzle	0,758	0,446	0,800
	Imagenet + Obroty	**0,797**	**0,552**	**0,820**
Las losowy	Losowa inicjalizacja	0,572	0,243	0,801
	Imagenet	0,727	0,368	0,803
	Puzzle	0,682	0,320	0,789
	Obroty	0,680	0,331	0,803
	Imagenet + Puzzle	0,708	0,390	0,812
	Imagenet + Obroty	**0,748**	**0,451**	**0,815**
SVM	Losowa inicjalizacja	0,599	0,260	0,805
	Imagenet	0,796	0,461	0,815
	Puzzle	0,675	0,313	0,801
	Obroty	0,740	0,391	0,800
	Imagenet + Puzzle	0,776	0,489	0,829
	Imagenet + Obroty	**0,814**	**0,581**	**0,851**

Świadczy to o tym, że reprezentacje niosą istotne informacje wykorzystywane w zadaniu klasyfikacji. Przeprowadzone eksperymenty wykazały, że połączenie wstępnego uczenia w sposób nadzorowany i samonadzorowany prowadzi do lepszych wyników we wszystkich badanych klasyfikatorach. W szczególności klasyfikator SVM poddany uczeniu na reprezentacjach generowanych przez sieć poddaną wstępnemu uczeniu na zbiorze ImageNet, a następnie metodą obroty uzyskał najlepszy wynik wśród trzech raportowanych metryk (ROC AUC, PR AUC oraz dokładność). Na uwagę zasługuje również wysoki wynik przy użyciu metody k najbliższych sąsiadów (k-NN), wykorzystujący informację o odległościach pomiędzy przykładami wejściowymi, dla reprezentacji uzyskanych z sieci wstępnie przeuczonej na zbiorze ImageNet i uczeniu samonadzorowanym. Wynik sugeruje, że reprezentacje są odpowiednio rozłożone w przestrzeni reprezentacji.

W opisanych eksperymentach nie zastosowano augmentacji danych. Dlatego zdecydowano się przeprowadzić eksperymenty, w których zastosowano bardziej intensywną augmentację danych składającą się z losowego wycinania, losowych odbić w pionie i poziomie, oraz losowych zaburzeń koloru. Rezultaty tych eksperymentów przedstawiono w tabeli 6.5, a macierze błędów w tabelach 6.6, 6.7 i 6.8.

Tabela 6.5: Trening na 2000 obrazach (cały zbiór danych) z wykorzystaniem augmentacji danych

Metoda	**ROC AUC**	**PR AUC**	**ACC**
ImageNet	0,825	0,587	84,10
ImageNet + Puzzle	0,835	0,607	83,33
ImageNet + Obroty	**0,842**	**0,613**	**85,17**

Tabela 6.6: Macierz błędów dla wstępnego uczenia z wykorzystaniem wstępnego uczenia na zbiorze ImageNet z wykorzystaniem augmentacji danych w zadaniu docelowym

		Klasa wygenerowana	
		Negatywna	Pozytywna
Klasa rzeczywista	Negatywna	446	37
	Pozytywna	59	59

Tabela 6.7: Macierz błędów dla wstępnego uczenia z wykorzystaniem wstępnego uczenia na zbiorze ImageNet oraz wstępnego uczenia metodą Puzzle z wykorzystaniem augmentacji danych w zadaniu docelowym

		Klasa wygenerowana	
		Negatywna	Pozytywna
Klasa rzeczywista	Negatywna	439	45
	Pozytywna	55	62

Tabela 6.8: Macierz błędów dla wstępnego uczenia z wykorzystaniem wstępnego uczenia na zbiorze ImageNet oraz wstępnego uczenia metodą Obroty z wykorzystaniem augmentacji danych w zadaniu docelowym

		Klasa wygenerowana	
		Negatywna	Pozytywna
Klasa rzeczywista	Negatywna	453	30
	Pozytywna	59	58

Tak jak oczekiwano, wyniki eksperymentów wykazują, że zastosowanie augmentacji danych o większym stopniu intensywności i większej liczbie operacji przyczyniło się do poprawy wyników. Ponownie, wykorzystanie sieci wstępnie uczonych w sposób nadzorowany, a następnie samonadzorowany prowadziło do uzyskania najlepszych wyników spośród wszystkich badanych rozwiązań. Warto jednak zauważyć, że różnica pomiędzy rozwiązaniem opartym o wstępny trening w sposób nadzorowany a rozwiązaniem opartym o trening w sposób nadzorowany i samonadzorowany jest mniejsza, w porównaniu do eksperymentów bez augmentacji danych. Wynika to z faktu, że rozszerzanie danych powoduje zwiększenie ilości danych, a uczenia samonadzorowane przynosi największe korzyści, gdy zbiór danych jest niewielki.

Najlepsze z zaproponowanych podejść umożliwiło uzyskanie wyników zbliżonych do najlepszych wyników raportowanych w konkursie ISIC2017. Trzy najlepsze rozwiązania w tym konkursie uzyskały wyniki ROC AUC kolejno 0,868, 0,856 oraz 0,874, podczas gdy prezentowane rozwiązanie przyniosło wynik 0,842. Porównanie uzyskanych rezultatów do innych rozwiązań z literatury przedstawione jest w tabeli 6.9.

Tabela 6.9: Porównanie proponowanego rozwiązania z innymi rozwiązaniami testowanymi na zbiorze ISIC2017

Sieć	**ROC AUC**
VGG16 [122]	0,766
VGG16 [73]	0,800
ResNet50 [122]	0,757
ResNet50 [73]	0,775
ResNet50 [123]	0,868
ResNet50 [124]	0,870
DenseNet161 [125]	0,818
DenseNet161 [73]	0,800
ResNet50 – ImageNet + Puzzle	0,835
ResNet50 – ImageNet + Obroty	0,842
ResNet50 – grupowanie modeli	0,856

Mimo że uzyskane wyniki nie są najlepsze, należy zwrócić uwagę na fakt, że zostały one uzyskane tylko przy użyciu sieci ResNet50 bez dodatkowych rozwiązań architektonicznych (np. modułu atencji), skomplikowanych metod rozszerzania danych, grupowania modeli czy augmentacji danych w trybie testowym, a takie mechanizmy stosowane są w najlepszych rozwiązaniach. Ponadto nie zastosowano dodatkowych danych treningowych, jak miało to miejsce w innych rozwiązaniach [50]. Przedstawione wyniki wykazują, że wysoką skuteczność działania można osiągnąć poprzez zastosowanie odpowiednich podejść uczenia.

W trakcie treningu wszystkie eksperymenty powtórzono pięć razy dla każdej metody w celu uzyskania bardziej wiarygodnych rezultatów. Podobnie jak w poprzednich rozdziałach zdecydowano się zastosować podejście grupowania modeli poprzez uśrednianie wyników sieci. Dzięki temu uzyskano wzrostu dokładności i powierzchni pod krzywą ROC odpowiednio do 86,16% i 0,856.

6.5. Integracja systemu poszukiwania struktury i uczenia dwuetapowego

Końcowym etapem badań było zastosowanie zaproponowanych metod wstępnego uczenia z architekturami głębokich sieci neuronowych uzyskanych przy wykorzystaniu

zaproponowanej metody automatycznego doboru struktury, opisanej w rozdziale 5. W przeprowadzonych eksperymentach wykazano, że najlepszą metodą wstępnego uczenia jest uczenie w sposób nadzorowany na zbiorze ImageNet, a następnie w sposób samonadzorowany na zbiorze znamion skórnych, dlatego tę metodę wybrano w eksperymentach ze strukturami dobranymi w wyniki działania algorytmu.

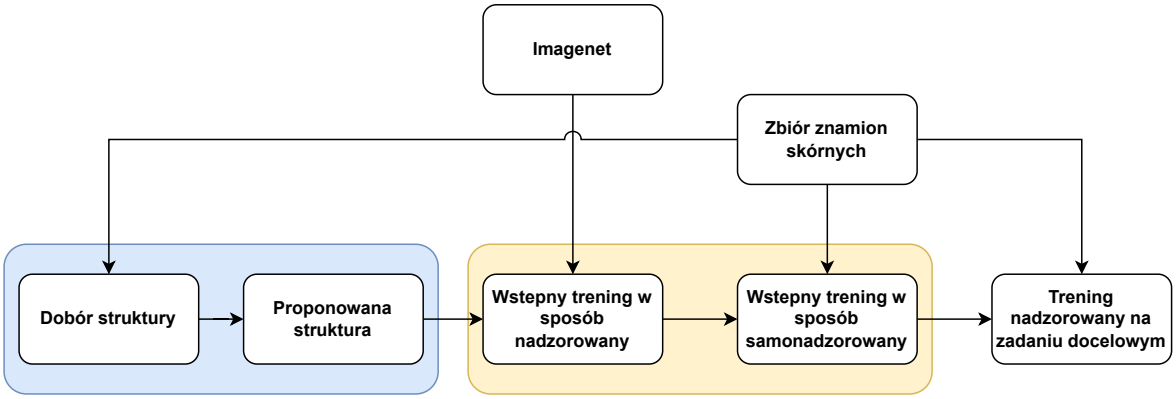
Aby wykazać efektywność metody, porównano cztery struktury sieci neuronowej wygenerowane w procesie doboru struktury, uzyskane w eksperymentach opisanych w rozdziale 5. Sieci zostały zainicjalizowane losowymi parametrami, a następnie przeszły wstępne uczenie w sposób nadzorowany i samonadzorowany.

Wstępne uczenie w sposób nadzorowany przeprowadzono z wykorzystaniem zbioru ImageNet. Zastosowano rozmiar wsadu 24, a trening trwał 50 epok. Większa liczba epok nie prowadziła do poprawy minimalizowanej miary, jaką była krosentropia. Nie wystąpiło również zjawisko nadmiernego dopasowania, objawiające się wzrostem minimalizowanej miary wyznaczanej na zbiorze walidacyjnym, co prawdopodobnie spowodowane było zbyt małym modelem w stosunku do rozmiaru stosowanego zbioru danych (1 mln obrazów). Liczba epok została dobrana na podstawie analizy wartości funkcji celu mierzonej na zbiorze walidacyjnym. Zastosowano optymalizator SGDR [16], z restartem po 25 epoce, współczynnik uczenia, dobrany na bazie obserwacji krzywej uczenia, przyjęto 0,001, a momentum 0,9. Wykorzystano następujące metody augmentacji danych: losowe wycinanie (ang. *random crop*), losowe zaburzenie koloru (ang. *color jittering*), losowe odbicia w pionie i poziome. Ostatnim krokiem była normalizacja każdego z obrazów.

Następnie przeprowadzono wstępne uczenie w sposób samonadzorowany. W trakcie treningu wykorzystano zbiór danych opisany w rozdziale 5 zawierający około 13 tys. obrazów znamion skórnych. Uczenie przeprowadzono z wykorzystaniem metody Obroty oraz Puzzle. W optymalizacji zastosowano algorytm SGD, na bazie obserwacji krzywej uczenia dobrano współczynnik uczenia o wartości 0,001, momentum 0,9, rozmiar wsadu 16. Ze względu na zwiększony zbiór danych zdecydowano się na zmniejszenie liczby epok wstępnego uczenia z 1000 do 300. Zastosowano metody augmentacji danych przedstawione w sekcjach opisujących zadania Obroty oraz Puzzle.

Po wstępnym uczeniu przeprowadzono trening na zadaniu docelowym, wykorzystując zbiór danych opisany w rozdziale 5. Zastosowano metodą SGDR, początkowy współczynnik uczenia wynosił 0,001, współczynnik momentum 0,9,

a rozmiar wsadu 24. Współczynnik uczenia resetowany był co 50 epok. Schemat proponowanego podejścia przedstawiony jest na rys. 6.14.



Rysunek 6.14: Schemat integracji metod doboru struktury ze wstępnym uczeniem

Wyniki eksperymentów przedstawiono w tabeli 6.10.

Tabela 6.10: Wyniki eksperymentów z połączenia NAS i wstępnego uczenia

Numer struktury	**Eksperyment**	**ROC AUC**	**Liczba parametrów**
1	NAS	0,825	1 198 593
	ImageNet + Puzzle	0,838	
	ImageNet + Obroty	**0,867**	
2	NAS	0,812	1 785 601
	ImageNet + Puzzle	**0,836**	
	ImageNet + Obroty	0,824	
3	NAS	0,823	1 054 593
	ImageNet + Puzzle	**0,863**	
	ImageNet + Obroty	0,851	
4	NAS	0,845	750 849
	ImageNet + Puzzle	0,858	
	ImageNet + Obroty	**0,872**	

Tabela przedstawia wyniki eksperymentów, które łączą dwie metody: NAS (ang. Neural Architecture Search) oraz wstępne uczenie. W eksperymentach zastosowano cztery różne struktury, które powstały w wyniku zastosowania algorytmu doboru struktury opisanego w rozdziale 5. Eksperymenty z oznaczeniem NAS zostały opisane w rozdziale 5. Eksperymenty oznaczone jako ImageNet + Puzzle oraz ImageNet +

Obroty polegały na zastosowaniu wstępnego uczenia do sieci otrzymanych w wyniki działania algorytmu doboru struktury.

Zastosowanie wstępnego uczenia prowadzi do uzyskania lepszych wyników każdej sieci w porównaniu z zastosowaniem tylko doboru struktury. W przypadku dwóch sieci (1 oraz 4) najlepsze wyniki osiągnięto stosując metodę ImageNet + Obroty, a w dwóch pozostałych (2 i 3) poprzez zastosowanie metody ImageNet + Puzzle. Przedstawione wyniki świadczą o efektywność wstępnego uczenia. Jednak, spośród dwóch zaproponowanych metod wstępnego uczenia nie można jednoznacznie wyznaczyć lepszej metody.

6.6. Podsumowanie

Wstępne uczenie jest często stosowane w sytuacjach, kiedy nie ma dostępnego dużego zbioru danych, który pozwoliłby na uzyskanie zadowalających wyników w treningu na zadaniu docelowym. W przypadku, gdy zbiór danych jest zbyt mały, uczenie sieci neuronowej z losowo zainicjalizowanymi parametrami może okazać się nieskuteczne. Często stosowaną praktyką jest wykorzystanie parametrów uzyskanych w trakcie uczenia na innym zadaniu w sposób nadzorowany. Jednakże, taka metoda może mieć mniejszą skuteczność, gdy zadanie pomocnicze różni się od docelowego.

Innym sposobem wstępnego uczenia jest stosowanie uczenie samonadzorowanego, co pozwala na wykorzystanie we wstępnym treningu danych nieoznaczonych. W literaturze przedstawiono wiele rozwiązań uczenia samonadzorowanego, ale większość testowanych rozwiązań jest sprawdzanych na licznych zbiorach danych zawierających setki tysięcy obrazów. W przeprowadzonych eksperymentach sprawdzono działanie uczenia samonadzorowanego w zadaniach małej skali, gdzie dostępny zbiór danych był niewielki.

W ramach badań przetestowano różne sposoby wstępnego uczenia. Porównano działanie sieci neuronowych zainicjalizowanych losowo, z sieciami wstępnie przeuczonymi w sposób nadzorowany oraz samonadzorowany. Do wstępnego uczenia w sposób nadzorowany zastosowano zbiór danych ImageNet, natomiast do wstępnego uczenia w sposób nadzorowany zbiór znamion skórnych, który był taki sam jak w zadaniu docelowym. Najlepsze wyniki przyniosło wstępne uczenie na zbiorze ImageNet, następnie wstępne uczenie samonadzorowane, i ostatecznie trening na zadaniu docelowym. Przeanalizowano działanie badanych metody przy użyciu

różnych rozmiarów zbiorów danych w treningu docelowym. Wstępne uczenie okazało się szczególnie przydatne w przypadku gdy zbiór danych był niewielki (20, 200 obrazów).

W badaniach zastosowano znaną architekturę ResNet50. Oprócz tego postanowiono zastosować struktury otrzymane wyniku automatycznego doboru struktury sieci neuronowej. W przeprowadzonych eksperymentach otrzymano analogiczne wyniki do tych uzyskanych poprzez zastosowanie sieci ResNet50. Ponownie, najlepsze wyniki przyniosło wstępne uczenie nadzorowane na zbiorze ImageNet, następnie wstępne uczenie samonadzorowane, a ostatecznie trening na zadaniu docelowym.

Połączenie automatycznego doboru struktury oraz uczenia samonadzorowanego niesie szereg korzyści. Automatyczny dobór struktury umożliwia opracowanie struktury, która zapewni mniejsze zużycie pamięci, szybsze generowanie odpowiedzi, a odpowiednia technika wstępnego uczenia może prowadzić do mniejszego zapotrzebowanie na dane oznaczone.

7. Podsumowanie rozprawy

W pracy doktorskiej podjęto się zbadania zagadnienia optymalnego doboru architektury głębokich sieci neuronowych oraz ich uczenia, w warunkach deficytu danych uczących. Algorytmy uczenia maszynowego proponowane w literaturze często tworzone są i testowane pod kątem realizacji zadań dużej skali, w których stosuje się duże modele trenowane przy użyciu licznych zbiorów danych i znaczącej mocy obliczeniowej. Jednak w wielu praktycznych realizacjach dostęp do zasobów obliczeniowych i do licznych zbiorów danych oznaczonych jest ograniczony, głównie ze względu na koszty. Zmotywowało to do podjęcia badań nad zadaniami małej skali przy relatywnie niewielkich zasobach obliczeniowych oraz przy ograniczonym rozmiarze danych uczących.

W pracy postawiono hipotezę, że zastosowanie podejścia optymalizującego do projektowania architektury oraz uczenia dwuetapowego, prowadzi do uzyskania mniejszych struktur, dopasowanych do danego problemu, przy jednoczesnym zachowaniu dokładności klasyfikacji. W celu potwierdzenia tej hipotezy przeprowadzono obszerny przegląd literatury, zaproponowano rozwiązania, oraz przeprowadzono liczne eksperymenty. Jako zadanie testowe przyjęto zadanie klasyfikacji znamion skórnych na zmiany łagodne i złośliwe.

Na wstępie dokonano generalnego opisu metod głębokiego uczenia w kontekście szerszej grupy podejść, czyli uczenia maszynowego. W tym celu przedstawiono podstawowe koncepcje związane z uczeniem maszynowym, przedstawiono podział algorytmów oraz opisano zadania realizowane przez te algorytmy. Następnie przedstawiono opis klasycznych, płytkich sieci neuronowych. Kolejnym krokiem było zaprezentowanie podstawowej struktury głębokiej sieci neuronowej stosowanej w analizie obrazu.

Kolejny rozdział obejmował zagadnienia, metody oraz techniki związane z głębokim uczeniem, głównie koncentrujące się na zagadnieniu analizy obrazu.

Dokonano w nim szczegółowego opisu najczęściej stosowanych technik. Przedstawiono warstwy powszechnie stosowane w konwolucyjnych sieciach neuronowych i omówiono ich funkcje. Następnie scharakteryzowano popularne architektury sieci neuronowych, z uwzględnieniem różnych podejść do projektowania ich struktur. Przedstawiono struktury o budowie liniowej, modułowej oraz z połączeniami skracającymi. Następnie przybliżono najważniejsze metody uczenia oraz inne techniki wspomagające uczenie głębokich sieci neuronowych. Na końcu przedstawiono metryki oraz metody ewaluacji, które mają zastosowanie w testowaniu rozwiązań proponowanych w ramach doktoratu.

W kolejnym rozdziale zaproponowano szereg eksperymentów, w których kolejno przeanalizowano wpływ architektury, zastosowanych hiperparametrów oraz różnych podejść do uczenia, na skuteczność działania modeli. W rozdziale tym przedstawiono również problematykę klasyfikacji znamion skórnych. Zaprezentowano różnego typu podejścia do automatycznej diagnozy znamion, zarówno oparte o klasyczne algorytmy przetwarzania obrazu, jak i te oparte o głębokie uczenie. W badaniach zdecydowano się zastosować głębokie sieci z rodziny VGG o liniowej strukturze. Wśród badanych czynników, najistotniejszy wpływ na wyniki miało zastosowanie techniki transfer learning oraz dobór odpowiedniej liczby warstw. Oprócz tego porównano metody regularyzacji oraz funkcje aktywacji, których wpływ na wyniki nie był aż tak istotny.

W kolejnym rozdziale podjęto tematykę metod automatycznego doboru struktury sieci neuronowej. Dokonano przeglądu metod z uwzględnieniem metod opartych o algorytmy ewolucyjne, uczenie ze wzmocnieniem oraz metody gradientowe. Zaproponowano rozwiązanie, w którym zastosowano algorytm wspinaczkowy, do nadzorowania procesu doboru struktury. Zgodnie z przyjętym algorytmem, sieć w kolejnych iteracjach ewoluowała od prostej kilkuwarstwowej struktury, aż po złożoną strukturę, dopasowaną do danego problemu. W celu rozbudowy sieci zastosowano transformacje zachowujące funkcje, które umożliwiają zwiększenie sieci bez utraty wcześniej nabytej wiedzy. Zastosowanie tego rozwiązania umożliwiło uzyskanie struktur, których skuteczność jest porównywalna do sieci VGG16, ale liczba parametrów stanowi tylko 2% liczby parametrów sieci VGG.

W kolejnym rozdziale przeanalizowano metody wstępnego uczenia sieci neuronowych, będące kolejnym z rozwiązań problemu deficytu danych oznaczonych. Dokonano przeglądu literatury, w którym skupiono się na dwóch dominujących podejściach: uczeniu samonadzorowanym z tzw. zadaniem pretekstowym oraz uczeniu

samonadzorowanym opartym o kontrastową funkcję celu. W rozprawie zaproponowano wykorzystanie metody PIRL (ang. *Pretext Invariant Representation Learning*). W eksperymentach porównano powszechnie stosowane wstępne uczenie w sposób nadzorowany na zbiorze ImageNet ze wstępnym uczeniem w sposób samonadzorowany na zbiorze znamion skórnych. Najlepszą skuteczność przyniosło sekwencyjne zastosowanie dwóch metod – najpierw wstępnego uczenia w sposób nadzorowany, a następnie wstępnego uczenia w sposób samonadzorowany. Końcowym etapem eksperymentów była integracja metod doboru struktury z metodami wstępnego uczenia. W tym celu struktury uzyskane w wyniku działania algorytmu poddane zostały zaproponowanemu uczeniu dwuetapowemu. I tym razem najlepsze wyniki przyniosło podejście sekwencyjne, a wprowadzenie wstępnego uczenia umożliwiło uzyskać lepsze wyniki, niż w przypadku zastosowania samego algorytmu doboru struktury.

Podsumowując, jako najważniejsze osiągnięcia przedstawione w pracy, uważam:

- dokonanie kompleksowego przeglądu rozwiązań stosowanych w głębokich sieciach neuronowych w zadaniach analizy obrazu, obejmującego architektury, warstwy, oraz metody wspomagające proces uczenia,
- przeprowadzenie eksperymentów i analizy wyników dotyczących wpływu architektury oraz metod modeli na osiągane wyniki klasyfikacji,
- dokonanie przeglądu metod automatycznego doboru struktury głębokich sieci neuronowych,
- propozycję algorytmu automatycznego doboru struktury sieci neuronowej działającego w warunkach deficytu danych uczących,
- dokonanie analizy metod wstępnego uczenia głębokich sieci neuronowych, ze szczególnym naciskiem na metody samonadzorowane,
- propozycję dwuetapowego procesu uczenia, wykorzystującego wstępne uczenie polegające na zastosowaniu sekwencyjnego wstępnego uczenia w sposób nadzorowany i samonadzorowany
- propozycję oraz analizę eksperymentalną i porównawczą systemu wspomagania decyzji, złożonego z wygenerowanej w sposób automatyczny architektury głębokiej sieci neuronowej optymalnie dostosowanej do analizowanego

problemu, uczonego w sposób dwuetapowy, zgodnie z zaproponowanym w pracy algorytmem.

Przedstawione badania oraz wnioski z nich płynące, pozwalają mi stwierdzić, że zastosowanie podejścia optymalizacyjnego przy projektowaniu architektury sieci dla danego problemu, a także dwuetapowego uczenia, umożliwia uzyskanie modeli neuronowych o znacznie mniejszej liczbie parametrów niż powszechnie wykorzystywane modele, nie tracąc przy tym ich zdolności do klasyfikacji, szczególnie w warunkach deficytu danych uczących.

Powyższe w mojej opinii potwierdza postawioną w pracy tezę.

Spis rysunków

Spis tabel

Bibliografia

[1] *Introducing ChatGPT*, https://openai.com/blog/chatgpt. (term. wiz. 26. 05. 2023).

[2] A. Krizhevsky, I. Sutskever i G. E. Hinton, „ImageNet Classification with Deep Convolutional Neural Networks," w *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a Meeting Held December 3-6, 2012, Lake Tahoe, Nevada, United States*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges i in., red., 2012, s. 1106–1114.

[3] A. Esteva, B. Kuprel, R. A. Novoa i in., „Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, t. 542, nr. 7639, s. 115–118, 2017, ISSN: 1476-4687. DOI: 10.1038/nature21056.

[4] E. F. Morales, R. Murrieta-Cid, I. Becerra i in., „A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning," *Intelligent Service Robotics*, t. 14, nr. 5, s. 773–805, 2021, ISSN: 1861-2784. DOI: 10.1007/s11370-021-00398-z.

[5] M. R. Bachute i J. M. Subhedar, „Autonomous Driving Architectures: Insights of Machine Learning and Deep Learning Algorithms," *Machine Learning with Applications*, t. 6, s. 100 164, 2021, ISSN: 2666-8270. DOI: 10.1016/j.mlwa.2021.100164.

[6] J. Jumper, R. Evans, A. Pritzel i in., „Highly accurate protein structure prediction with AlphaFold," *Nature*, t. 596, nr. 7873, s. 583–589, 2021, ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2.

[7] M. L. Bileschi, D. Belanger, D. H. Bryant i in., „Using deep learning to annotate the protein universe," *Nature Biotechnology*, t. 40, nr. 6, s. 932–937, 2022, ISSN: 1546-1696. DOI: 10.1038/s41587-021-01179-w.

[8] A. Kamilaris i F. X. Prenafeta-Boldú, „Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, t. 147, s. 70–90, 2018, ISSN: 0168-1699. DOI: 10.1016/j.compag.2018.02.016.

[9] A. Ramesh, M. Pavlov, G. Goh i in., *Zero-Shot Text-to-Image Generation*, 2021. DOI: 10.48550/arXiv.2102.12092. arXiv: 2102.12092 [cs].

[10] A. Vaswani, N. Shazeer, N. Parmar i in., „Attention Is All You Need," *Advances in neural information processing systems*, t. 30, 2017.

[11] A. Ramesh, P. Dhariwal, A. Nichol i in., „Hierarchical Text-Conditional Image Generation with Clip Latents," *arXiv preprint arXiv:2204.06125*, 2022. arXiv: 2204.06125.

[12] C. Szegedy, W. Liu, Y. Jia i in., „Going Deeper with Convolutions," w *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, s. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[13] K. He, X. Zhang, S. Ren i in., „Deep Residual Learning for Image Recognition," w *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, s. 770–778. DOI: 10.1109/CVPR.2016.90.

[14] K. Simonyan i A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition," w *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio i Y. LeCun, red., 2015.

[15] D. P. Kingma i J. Ba, „Adam: A Method for Stochastic Optimization," w *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio i Y. LeCun, red., 2015.

[16] I. Loshchilov i F. Hutter, „SGDR: Stochastic Gradient Descent with Warm Restarts," w *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[17] N. Srivastava, G. Hinton, A. Krizhevsky i in., „Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, t. 15, nr. 56, s. 1929–1958, 2014, ISSN: 1533-7928.

[18] S. Ioffe i C. Szegedy, „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," w *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, 2015, s. 448–456.

[19] G. Huang, Y. Sun, Z. Liu i in., *Deep Networks with Stochastic Depth*, 2016.

[20] H. Liu, K. Simonyan i Y. Yang, „DARTS: Differentiable Architecture Search," w *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[21] T. Chen, S. Kornblith, M. Norouzi i in., „A Simple Framework for Contrastive Learning of Visual Representations," w *Proceedings of the 37th International Conference on Machine Learning*, PMLR, 2020, s. 1597–1607.

[22] I. Misra i L. van der Maaten, „Self-Supervised Learning of Pretext-Invariant Representations," w *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, s. 6707–6717.

[23] J.-B. Grill, F. Strub, F. Altché i in., „Bootstrap Your Own Latent a New Approach to Self-Supervised Learning," w *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20, Red Hook, NY, USA: Curran Associates Inc., 2020, s. 21 271–21 284, ISBN: 978-1-71382-954-6.

[24] A. Kwasigroch, M. Grochowski i A. Mikołajczyk, „Self-Supervised Learning to Increase the Performance of Skin Lesion Classification," *Electronics*, t. 9, nr. 11, s. 1930, 2020, ISSN: 2079-9292. DOI: 10.3390/electronics9111930.

[25] A. Kwasigroch, M. Grochowski i A. Mikołajczyk, „Neural Architecture Search for Skin Lesion Classification," *IEEE Access*, t. 8, s. 9061–9071, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2964424.

[26] A. Kwasigroch, M. Grochowski i M. Mikolajczyk, „Deep Neural Network Architecture Search Using Network Morphism," w *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2019, s. 30–35. DOI: 10.1109/MMAR.2019.8864624.

[27] M. Grochowski, A. Kwasigroch i A. Mikołajczyk, „Selected Technical Issues of Deep Neural Networks for Image Classification Purposes," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, t. 67, nr. No. 2, s. 363–376, 2019. DOI: 10.24425/bpas.2019.128485.

[28] D. G. Lowe, „Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, t. 60, nr. 2, s. 91–110, 2004, ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94.

[29] P. Viola i M. Jones, „Rapid Object Detection Using a Boosted Cascade of Simple Features," w *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, t. 1, 2001, s. I–I. DOI: 10.1109/CVPR.2001.990517.

[30] N. Dalal i B. Triggs, „Histograms of Oriented Gradients for Human Detection," w *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, t. 1, 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[31] D. Mahajan, R. Girshick, V. Ramanathan i in., „Exploring the Limits of Weakly Supervised Pretraining," w *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu i in., red., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, s. 185–201, ISBN: 978-3-030-01216-8. DOI: 10.1007/978-3-030-01216-8_12.

[32] L. van der Maaten i G. Hinton, „Visualizing Data Using T-SNE," *Journal of Machine Learning Research*, t. 9, s. 2579–2605, 2008.

[33] Y. Matsubara, R. Yang, M. Levorato i in., „Supervised Compression for Resource-Constrained Edge Computing Systems," w *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, s. 923–933. DOI: 10.1109/WACV51458.2022.00100.

[34] I. Goodfellow, Y. Bengio i A. Courville, *Deep Learning*. MIT Press, 2016.

[35] A. L. Maas, A. Y. Hannun i A. Y. Ng, „Rectifier Nonlinearities Improve Neural Network Acoustic Models," w *In ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[36] K. He, X. Zhang, S. Ren i in., „Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," w *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, s. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[37] X. Glorot i Y. Bengio, „Understanding the Difficulty of Training Deep Feedforward Neural Networks," w *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh i M. Titterington, red., ser. Proceedings of Machine Learning Research, t. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, s. 249–256.

[38] L. J. Ba, J. R. Kiros i G. E. Hinton, „Layer Normalization," *CoRR*, t. abs/1607.06450, 2016. arXiv: 1607.06450.

[39] K. He i J. Sun, „Convolutional Neural Networks at Constrained Time Cost," w *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, s. 5353–5360.

[40] R. K. Srivastava, K. Greff i J. Schmidhuber, *Highway Networks*, 2015.

[41] J. Hu, L. Shen i G. Sun, „Squeeze-and-Excitation Networks," w *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, s. 7132–7141. DOI: 10.1109/CVPR.2018.00745.

[42] S. Xie, R. Girshick, P. Dollár i in., „Aggregated Residual Transformations for Deep Neural Networks," w *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, s. 5987–5995. DOI: 10.1109/CVPR.2017.634.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe i in., „Rethinking the Inception Architecture for Computer Vision," w *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, s. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[44] C. Szegedy, S. Ioffe, V. Vanhoucke i in., „Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," w *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17, San Francisco, California, USA: AAAI Press, 2017, s. 4278–4284.

[45] G. Huang, Z. Liu, L. Van Der Maaten i in., „Densely Connected Convolutional Networks," w *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, s. 2261–2269. DOI: 10.1109/CVPR.2017.243.

[46] M. Tan i Q. Le, „EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," w *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri i R. Salakhutdinov, red., ser. Proceedings of Machine Learning Research, t. 97, PMLR, 2019, s. 6105–6114.

[47] D. E. Rumelhart, G. E. Hinton i R. J. Williams, „Learning representations by back-propagating errors," *Nature*, t. 323, nr. 6088, s. 533–536, 1986, ISSN: 1476-4687. DOI: 10.1038/323533a0.

[48] J. Duchi, E. Hazan i Y. Singer, „Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *The Journal of Machine Learning Research*, t. 12, nr. null, s. 2121–2159, 2011, ISSN: 1532-4435.

[49] G. Hinton, *Neural Networks for Machine Learning*, Coursera, Video Lectures.

[50] *ISIC Challenge*, https://challenge.isic-archive.com/landing/2017. (term. wiz. 14. 07. 2020).

[51] S. J. Pan i Q. Yang, „A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, t. 22, nr. 10, s. 1345–1359, 2010, ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191.

[52] M. Oquab, L. Bottou, I. Laptev i in., „Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks," w *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, s. 1717–1724. DOI: 10.1109/CVPR.2014.222.

[53] J. Deng, W. Dong, R. Socher i in., „Imagenet: A Large-Scale Hierarchical Image Database," w *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Ieee, 2009, s. 248–255.

[54] T.-Y. Lin, M. Maire, S. Belongie i in., „Microsoft COCO: Common Objects in Context," w *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele i in., red., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2014, s. 740–755, ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48.

[55] M. Everingham, L. Gool, C. K. Williams i in., „The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, t. 88, nr. 2, s. 303–338, 2010, ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4.

[56] X. Li, S. Chen, X. Hu i in., „Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift," w *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, s. 2677–2685. DOI: 10.1109/CVPR.2019.00279.

[57] R. R. Selvaraju, M. Cogswell, A. Das i in., „Grad-Cam: Visual Explanations from Deep Networks via Gradient-Based Localization," w *Proceedings of the IEEE International Conference on Computer Vision*, 2017, s. 618–626.

[58] G. Montavon, A. Binder, S. Lapuschkin i in., „Layer-Wise Relevance Propagation: An Overview," w *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, ser. Lecture Notes in Computer Science, W. Samek, G. Montavon, A. Vedaldi i in., red., Cham: Springer International Publishing, 2019, s. 193–209, ISBN: 978-3-030-28954-6. DOI: 10.1007/978-3-030-28954-6_10.

[59] M. T. Ribeiro, S. Singh i C. Guestrin, „"Why Should i Trust You?"Explaining the Predictions of Any Classifier," w *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, s. 1135–1144.

[60] S. M. Lundberg i S.-I. Lee, „A Unified Approach to Interpreting Model Predictions," *Advances in neural information processing systems*, t. 30, 2017.

[61] O. Russakovsky, J. Deng, H. Su i in., „ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, t. 115, nr. 3, s. 211–252, 2015, ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y.

[62] J. Ferlay, M. Colombet, I. Soerjomataram i in., „Cancer statistics for the year 2020: An overview," *International Journal of Cancer*, t. 149, nr. 4, s. 778–789, 2021, ISSN: 1097-0215. DOI: 10.1002/ijc.33588.

[63] B.-A. C, D. Sw, A. Al i in., „Differences between polarized light dermoscopy and immersion contact dermoscopy for the evaluation of skin lesions," *Archives of dermatology*, t. 143, nr. 3, 2007, ISSN: 0003-987X. DOI: 10.1001/archderm.143.3.329.

[64] R. H. Johr, „Dermoscopy: Alternative melanocytic algorithms—the ABCD rule of dermatoscopy, menzies scoring method, and 7-point checklist," *Clinics in Dermatology*, t. 20, nr. 3, s. 240–247, 2002, ISSN: 0738-081X, 1879-1131. DOI: 10.1016/S0738-081X(02)00236-5.

[65] A. Miko\lajczyk, S. Majchrowska i S. Carrasco Limeros, „The (de) Biasing Effect of GAN-based Augmentation Methods on Skin Lesion Images," w *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VIII*, Springer, 2022, s. 437–447.

[66] M. E. Celebi, H. A. Kingravi, B. Uddin i in., „A methodological approach to the classification of dermoscopy images," *Computerized Medical Imaging and Graphics*, t. 31, nr. 6, s. 362–373, 2007, ISSN: 0895-6111. DOI: 10.1016/j.compmedimag.2007.01.003.

[67] D. A. Clausi, „An Analysis of Co-Occurrence Texture Statistics as a Function of Grey Level Quantization," *Canadian Journal of Remote Sensing*, t. 28, nr. 1, s. 45–62, 2002, ISSN: 0703-8992. DOI: 10.5589/m02-004.

[68] D. S.Gopinathan i S. N. A. Rani, „The Melanoma Skin Cancer Detection and Feature Extraction through Image Processing Techniques," *International Journal of Application or Innovation in Engineering & Management*, t. Volume 5, Issue 4, July - August 2016, 2016.

[69] M. H. Jafari, S. Samavi, N. Karimi i in., „Automatic detection of melanoma using broad extraction of features from digital images," *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference*, t. 2016, s. 1357–1360, 2016, ISSN: 2694-0604. DOI: 10.1109/EMBC.2016.7590959.

[70] Q. Abbas, M. Emre Celebi, I. F. Garcia i in., „Melanoma recognition framework based on expert definition of ABCD for dermoscopic images," *Skin Research and Technology*, t. 19, nr. 1, e93–e102, 2013, ISSN: 1600-0846. DOI: 10.1111/j.1600-0846.2012.00614.x.

[71] Y. Fujisawa, Y. Otomo, Y. Ogata i in., „Deep-learning-based, computer-aided classifier developed with a small dataset of clinical images surpasses board-certified dermatologists in skin tumour diagnosis," *British Journal of Dermatology*, t. 180, nr. 2, s. 373–381, 2019, ISSN: 1365-2133. DOI: 10.1111/bjd.16924.

[72] J. Zhang, Y. Xie, Y. Xia i in., „Attention Residual Learning for Skin Lesion Classification," *IEEE Transactions on Medical Imaging*, t. 38, nr. 9, s. 2092–2103, 2019, ISSN: 1558-254X. DOI: 10.1109/TMI.2019.2893944.

[73] C. Barata, M. E. Celebi i J. S. Marques, „Explainable Skin Lesion Diagnosis Using Taxonomies," *Pattern Recognition*, s. 107 413, 2020. DOI: 10.1016/j.patcog.2020.107413.

[74] K. O. Stanley i R. Miikkulainen, „Evolving Neural Networks through Augmenting Topologies," *Evolutionary Computation*, t. 10, nr. 2, s. 99–127, 2002, ISSN: 1063-6560. DOI: 10.1162/106365602320169811.

[75] E. D. Cubuk, B. Zoph, D. Mane i in., *AutoAugment: Learning Augmentation Policies from Data*, 2019. DOI: 10.48550/arXiv.1805.09501. arXiv: 1805.09501 [cs, stat].

[76] J. Bergstra, D. Yamins i D. D. Cox, „Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures," w *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, ser. JMLR Workshop and Conference Proceedings, t. 28, JMLR.org, 2013, s. 115–123.

[77] E. Real, A. Aggarwal, Y. Huang i in., „Regularized Evolution for Image Classifier Architecture Search," w *Proceedings of the AAAI Conference on Artificial Intelligence*, t. 33, 2019, s. 4780–4789.

[78] F. Runge, D. Stoll, S. Falkner i in., *Learning to Design RNA*, 2019. DOI: 10.48550/arXiv.1812.11951. arXiv: 1812.11951 [cs, q-bio, stat].

[79] S. Falkner, A. Klein i F. Hutter, „BOHB: Robust and Efficient Hyperparameter Optimization at Scale," w *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, J. G. Dy i A. Krause, red., ser. Proceedings of Machine Learning Research, t. 80, PMLR, 2018, s. 1436–1445.

[80] B. Baker, O. Gupta, R. Raskar i in., „Accelerating Neural Architecture Search Using Performance Prediction," w *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, OpenReview.net, 2018.

[81] A. Klein, S. Falkner, J. T. Springenberg i in., „Learning Curve Prediction with Bayesian Neural Networks," w *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[82] H. Cai, T. Chen, W. Zhang i in., „Efficient Architecture Search by Network Transformation," w *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial*

Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, ser. AAAI'18/IAAI'18/EAAI'18, New Orleans, Louisiana, USA: AAAI Press, 2018, s. 2787–2794, ISBN: 978-1-57735-800-8.

[83] T. Elsken, J.-H. Metzen i F. Hutter, *Simple And Efficient Architecture Search for Convolutional Neural Networks*, 2017. DOI: 10.48550/arXiv.1711.04528. arXiv: 1711.04528 [cs, stat].

[84] G. F. Miller, P. M. Todd i S. U. Hegde, „Designing Neural Networks Using Genetic Algorithms.," w *ICGA*, t. 89, 1989, s. 379–384.

[85] E. Real, S. Moore, A. Selle i in., „Large-Scale Evolution of Image Classifiers," w *Proceedings of the 34th International Conference on Machine Learning*, PMLR, 2017, s. 2902–2911.

[86] A. Krizhevsky, „Learning Multiple Layers of Features from Tiny Images," s. 32–33, 2009.

[87] M. Baldeon Calisto i S. K. Lai-Yuen, „AdaEn-Net: An ensemble of adaptive 2D–3D Fully Convolutional Networks for medical image segmentation," *Neural Networks*, t. 126, s. 76–94, 2020, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.03.007.

[88] Z. Fan, J. Wei, G. Zhu i in., *Evolutionary Neural Architecture Search for Retinal Vessel Segmentation*, 2020. DOI: 10.48550/arXiv.2001.06678. arXiv: 2001.06678 [cs, eess].

[89] B. Zoph i Q. V. Le, „Neural Architecture Search with Reinforcement Learning," w *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[90] S. Hochreiter i J. Schmidhuber, „Long Short-Term Memory," *Neural Comput.*, t. 9, nr. 8, s. 1735–1780, 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[91] R. J. Williams, „Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, t. 8, nr. 3, s. 229–256, 1992, ISSN: 1573-0565. DOI: 10.1007/BF00992696.

[92] C. Liu, B. Zoph, M. Neumann i in., „Progressive Neural Architecture Search," w *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu i in., red., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, s. 19–35, ISBN: 978-3-030-01246-5. DOI: 10.1007/978-3-030-01246-5_2.

[93] A. Mortazi i U. Bagci, „Automatically Designing CNN Architectures for Medical Image Segmentation," w *Machine Learning in Medical Imaging*, Y. Shi, H.-I. Suk i M. Liu, red., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, s. 98–106, ISBN: 978-3-030-00919-9. DOI: 10.1007/978-3-030-00919-9_12.

[94] Z. Xu, S. Zuo, E. Y. Lam i in., „AutoSegNet: An Automated Neural Network for Image Segmentation," *IEEE Access*, t. 8, s. 92 452–92 461, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2995367.

[95] H. Cai, L. Zhu i S. Han, „ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware," w *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[96] Y. Xu, L. Xie, X. Zhang i in., „PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search," w *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[97] Y. Peng, L. Bi, M. Fulham i in., „Multi-modality Information Fusion for Radiomics-Based Neural Architecture Search," w *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, s. 763–771, ISBN: 978-3-030-59728-3. DOI: 10.1007/978-3-030-59728-3_74.

[98] Y. Weng, T. Zhou, Y. Li i in., „NAS-Unet: Neural Architecture Search for Medical Image Segmentation," *IEEE Access*, t. 7, s. 44 247–44 257, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2908991.

[99] Z. Zhu, C. Liu, D. Yang i in., *V-NAS: Neural Architecture Search for Volumetric Medical Image Segmentation*, 2019. DOI: 10.48550/arXiv.1906.02817. arXiv: 1906.02817 [cs, eess].

[100] S. Russell i P. Norvig, *Artificial Intelligence: A Modern Approach*, Third. Prentice Hall, 2010.

[101] T. Chen, I. J. Goodfellow i J. Shlens, „Net2Net: Accelerating Learning via Knowledge Transfer," w *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio i Y. LeCun, red., 2016.

[102] F. Yang, W. Zhang, L. Tao i in., „Transfer Learning Strategies for Deep Learning-based PHM Algorithms," *Applied Sciences*, t. 10, nr. 7, s. 2361, 2020, ISSN: 2076-3417. DOI: 10.3390/app10072361.

[103] C. Sudlow, J. Gallacher, N. Allen i in., „UK Biobank: An Open Access Resource for Identifying the Causes of a Wide Range of Complex Diseases of Middle and Old Age," *PLOS Medicine*, t. 12, nr. 3, e1001779, 2015, ISSN: 1549-1676. DOI: 10.1371/journal.pmed.1001779.

[104] C. Doersch, A. Gupta i A. A. Efros, „Unsupervised Visual Representation Learning by Context Prediction," w *Proceedings of the IEEE International Conference on Computer Vision*, 2015, s. 1422–1430.

[105] N. Komodakis i S. Gidaris, „Unsupervised Representation Learning by Predicting Image Rotations," w *International Conference on Learning Representations (ICLR)*, 2018.

[106] M. Cimpoi, S. Maji, I. Kokkinos i in., „Describing Textures in the Wild," w *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14, USA: IEEE Computer Society, 2014, s. 3606–3613, ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.461.

[107] S. Yamaguchi, S. Kanai, T. Shioda i in., „Image Enhanced Rotation Prediction for Self-Supervised Learning," w *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, s. 489–493. DOI: 10.1109/ICIP42928.2021.9506132.

[108] M. Noroozi i P. Favaro, „Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles," w *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe i in., red., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2016, s. 69–84, ISBN: 978-3-319-46466-4. DOI: 10.1007/978-3-319-46466-4_5.

[109] D. Pathak, P. Krähenbühl, J. Donahue i in., „Context Encoders: Feature Learning by Inpainting," w *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, s. 2536–2544. DOI: 10.1109/CVPR.2016.278.

[110] G. Larsson, M. Maire i G. Shakhnarovich, „Learning Representations for Automatic Colorization," w *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe i in., red., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2016, s. 577–593, ISBN: 978-3-319-46493-0. DOI: 10.1007/978-3-319-46493-0_35.

[111] R. Zhang, P. Isola i A. A. Efros, „Colorful Image Colorization," w *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe i in., red., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2016, s. 649–666, ISBN: 978-3-319-46487-9. DOI: 10.1007/978-3-319-46487-9_40.

[112] A. Bardes, J. Ponce i Y. LeCun, „VICRegL: Self-supervised Learning of Local Visual Features," w *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal i in., red., t. 35, Curran Associates, Inc., 2022, s. 8799–8810.

[113] Y. Tian, D. Krishnan i P. Isola, „Contrastive Multiview Coding," w *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox i in., red., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, s. 776–794, ISBN: 978-3-030-58621-8. DOI: 10.1007/978-3-030-58621-8_45.

[114] K. He, H. Fan, Y. Wu i in., „Momentum Contrast for Unsupervised Visual Representation Learning," w *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, s. 9726–9735. DOI: 10.1109/CVPR42600.2020.00975.

[115] X. Chen, H. Fan, R. Girshick i in., *Improved Baselines with Momentum Contrastive Learning*, 2020. DOI: 10.48550/arXiv.2003.04297. arXiv: 2003.04297 [cs].

[116] P. Vincent, H. Larochelle, Y. Bengio i in., „Extracting and Composing Robust Features with Denoising Autoencoders," w *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08, New York, NY, USA: Association for Computing Machinery, 2008, s. 1096–1103, ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294.

[117] I. Goodfellow, J. Pouget-Abadie, M. Mirza i in., „Generative Adversarial Nets," w *Advances in Neural Information Processing Systems*, 2014, s. 2672–2680.

[118] J. Donahue, P. Krähenbühl i T. Darrell, „Adversarial Feature Learning," w *International Conference on Learning Representations*, 2022.

[119] M. Mehralian i B. Karasfi, „RDCGAN: Unsupervised Representation Learning With Regularized Deep Convolutional Generative Adversarial Networks," w *2018 9th Conference on Artificial Intelligence and Robotics and 2nd Asia-Pacific International Symposium*, 2018, s. 31–38. DOI: 10.1109/AIAR.2018.8769811.

[120] O. J. Hénaff, A. Srinivas, J. De Fauw i in., „Data-Efficient Image Recognition with Contrastive Predictive Coding," w *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20, JMLR.org, 2020, s. 4182–4192.

[121] Z. Wu, Y. Xiong, S. Yu i in., *Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination*, 2018. DOI: 10.48550/arXiv.1805.01978. arXiv: 1805.01978 [cs].

[122] B. Harangi, „Skin Lesion Classification with Ensembles of Deep Convolutional Neural Networks," *Journal of Biomedical Informatics*, t. 86, s. 25–32, 2018. DOI: 10.1016/j.jbi.2018.08.006.

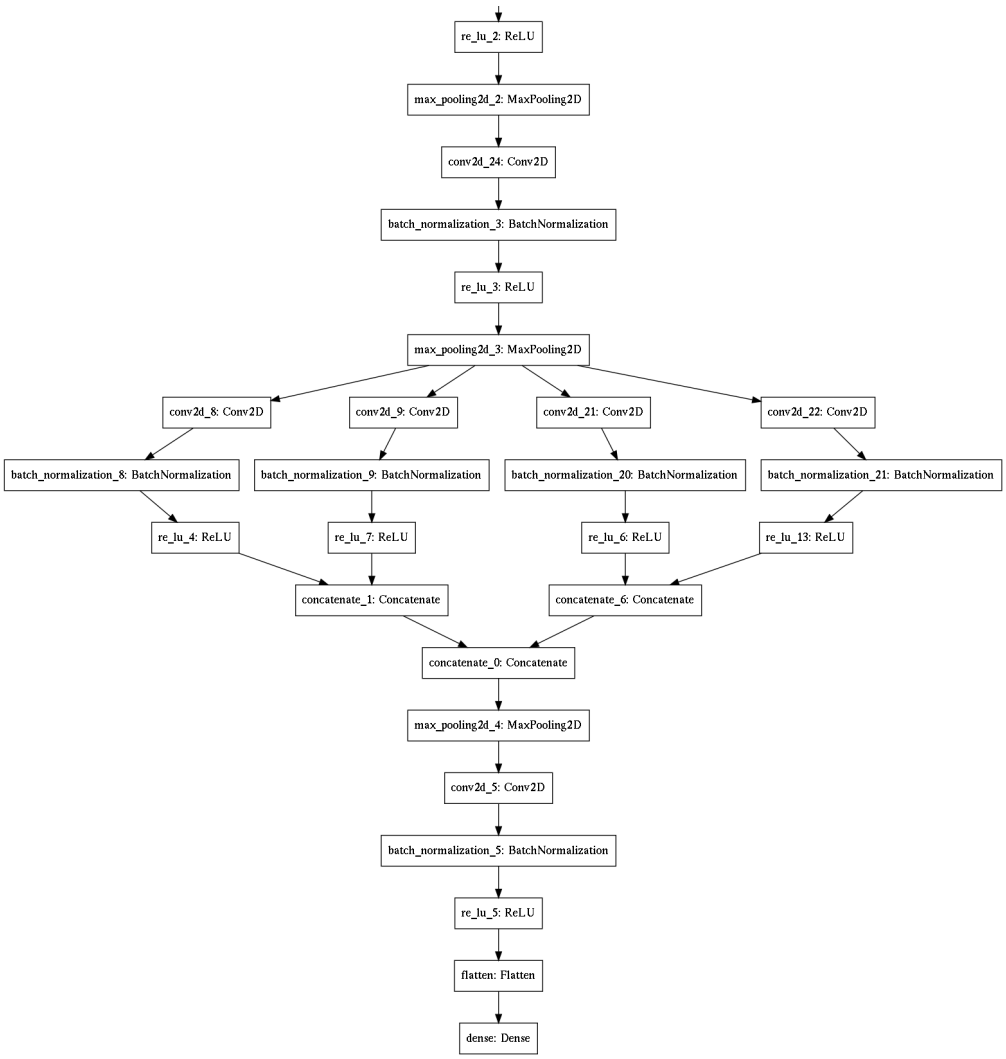
[123] K. Matsunaga, A. Hamada, A. Minagawa i in., *Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble*, 2017. eprint: arXiv:1703.03108.

[124] L. Bi, J. Kim, E. Ahn i in., *Automatic Skin Lesion Analysis Using Large-Scale Dermoscopy Images and Deep Residual Networks*, 2017. eprint: arXiv:1703.04197.

[125] C. Barata i J. S. Marques, „Deep Learning for Skin Cancer Diagnosis with Hierarchical Architectures," w *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, IEEE, 2019, s. 841–845.

Dodatek A

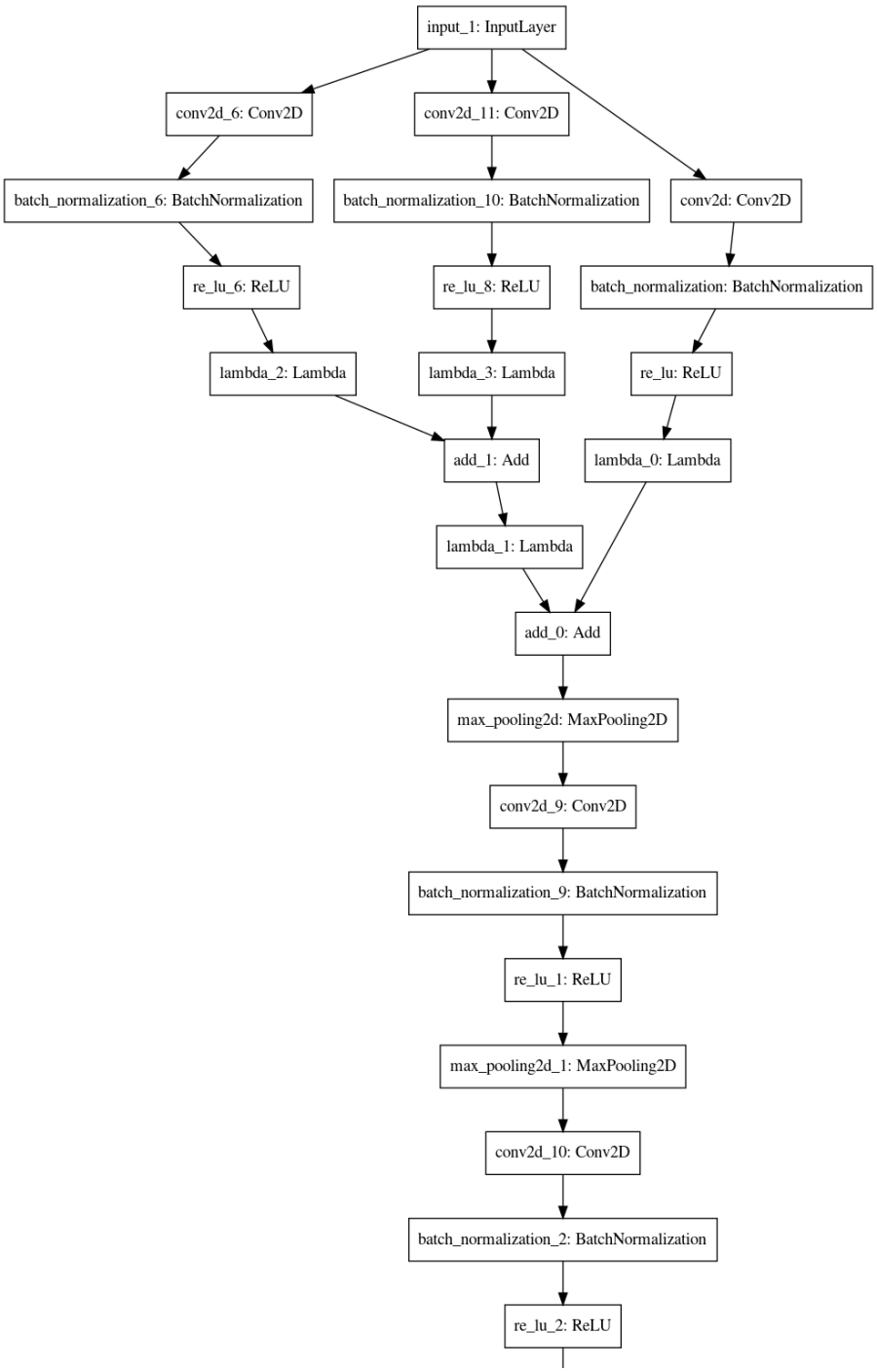
Rysunek A.1: Model 1 – część 1



Rysunek A.2: Model 1 – część 2

Tabela A.1: Model 1 – warstwy

Warstwa	**Liczba filtrów**
conv2d_15	64
conv2d_17	32
conv2d_18	32
conv2d_19	64
conv2d_20	64
conv2d_21	32
conv2d_22	32
conv2d_23	512
conv2d_24	128



Rysunek A.3: Model 2 – część 1

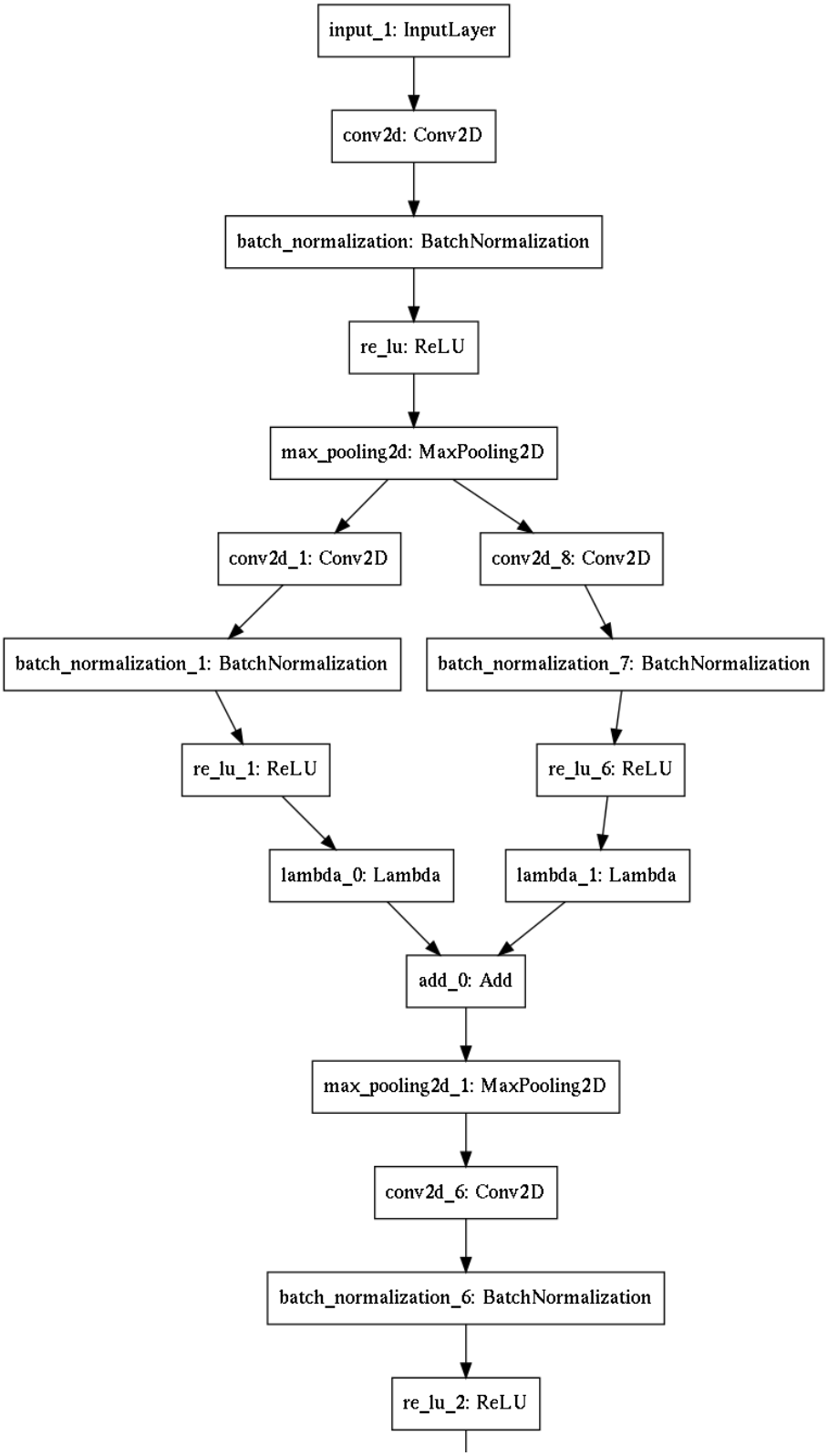
Rysunek A.4: Model 2 – część 2

Tabela A.2: Model 2 – warstwy

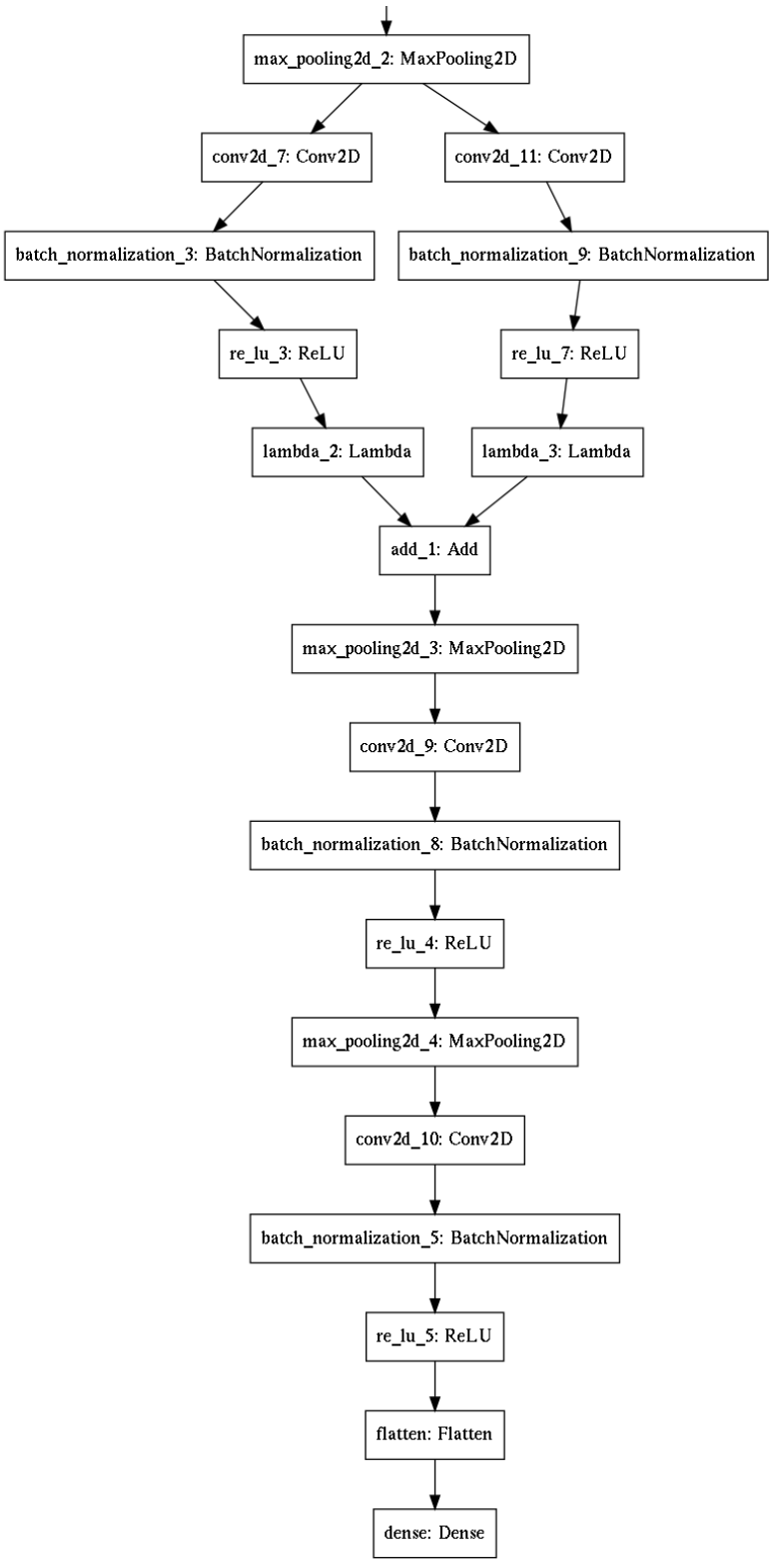
Warstwa	**Liczba filtrów**
conv2d	128
conv2d_3	128
conv2d_4	128
conv2d_6	128
conv2d_8	64
conv2d_9	256
conv2d_10	128
conv2d_11	128
conv2d_13	32
conv2d_14	16
conv2d_15	16

Tabela A.3: Model 3 – warstwy

Warstwa	**Liczba filtrów**
conv2d	128
conv2d_1	128
conv2d_6	256
conv2d_7	128
conv2d_8	128
conv2d_9	256
conv2d_10	128
conv2d_11	128



Rysunek A.5: Model 3 – część 1



Rysunek A.6: Model 3 – część 2