AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ
ZARZĄDZANIA

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

Budowa modelu uczenia maszynowego do wspomagania diagnoz medycznych

Construction of a machine learning model to support medical diagnoses

Autor: Agnieszka Barbara Pytel
Kierunek studiów: Informatyka i ekonometria
Opiekun pracy: dr inż. Jerzy Duda

Kraków, 2019

„Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.”

Podpis dyplomanta

……..

Spis treści

1. Wstęp

Uczenie maszynowe zdobywa coraz większą popularność ze względu na możliwość wykorzystania w różnorodnych dziedzinach, łatwość implementacji oraz bardzo dobre rezultaty jego zastosowania dla danego problemu. Dzięki wciąż powiększającej się mocy obliczeniowej komputerów mocno rozwinęło się w ostatnich latach szczególnie uczenie głębokie, które obecnie jest jednym z najczęściej wybieranych narzędzi w rozwiązywaniu zadań dotyczących przetwarzania obrazów, wideo czy tekstu. Zastosowanie uczenia głębokiego do klasyfikacji obrazów pozwala obecnie osiągnąć wyniki lepsze niż w przypadku klasyfikacji przez człowieka. Jest to szczególnie istotny fakt w kontekście analizy danych medycznych, gdzie prawidłowa lub nieprawidłowa klasyfikacja niejednokrotnie jest kwestią życia lub śmierci pacjenta.

Wykorzystanie uczenia maszynowego w medycynie pozwala na częściowe zautomatyzowanie procesu diagnozowania i szybszą reakcję na zaobserwowane objawy niż w przypadku standardowej analizy badań przez lekarza specjalistę. W przyszłości uczenie głębokie może zostać wykorzystane m. in. do zdalnej diagnozy pacjentów na podstawie danych przesyłanych z kamer lub urządzeń specjalistycznych. Wszelkie nowe osiągnięcia uczenia maszynowego w medycynie są niezwykle cenne.

1.1. Cel pracy

Celem niniejszej pracy jest przegląd dotychczasowych dokonań w obszarze rozpoznawania obrazów medycznych z wykorzystaniem uczenia maszynowego oraz stworzenie i porównanie sześciu modeli wykorzystujących głębokie sieci neuronowe do klasyfikacji binarnej na niewielkim zbiorze obrazów rentgenowskich w celu wykrycia u pacjenta zapalenia płuc. Wyboru modeli dokonano kierując się dotychczasowymi osiągnięciami uczenia maszynowego w klasyfikacji obrazów medycznych. Wykorzystano konwolucyjną sieć neuronową zbudowaną od podstaw oraz uczenie transferowe dla pięciu modeli, osiągających bardzo dobre wyniki w klasyfikacji obrazów w skali ogólnoświatowej, udostępnionych do użytku publicznego. W pracy przedstawiono rozwiązania dostępne dla każdego, niewymagające specjalistycznego sprzętu technicznego.

1.2. Struktura pracy

Niniejsza praca magisterska składa się z trzech głównych rozdziałów. W Rozdziale 2. przedstawiono historię osiągnięć z zakresu uczenia maszynowego w dziedzinie medycyny, ze szczególnym uwzględnieniem technik wykorzystanych w Rozdziale 4., oraz przeanalizowano dotychczasowe prace, których celem jest również klasyfikacja binarna wykorzystanego w tym opracowaniu zbioru obrazów rentgenowskich płuc.

Rozdział 3. zawiera podstawy teoretyczne uczenia maszynowego oraz sieci neuronowych, w tym szczegółowy opis konwolucyjnych sieci neuronowych oraz modeli wykorzystanych do uczenia transferowego, potrzebne do zrozumienia rozwiązań przedstawionych w Rozdziale 4. niniejszej pracy.

W Rozdziale 4. dokonano analizy wykorzystywanego zbioru danych oraz skonstruowano sześć modeli uczenia maszynowego do klasyfikacji obrazów ze zbioru danych w celu wykrycia zapalenia płuc. Rozdział kończy się szczegółowym porównaniem wyników osiągniętych przez autorów innych opracowań wykorzystujących ten sam zbiór danych oraz wyników modeli stworzonych w niniejszym opracowaniu.

2. Aktualny stan wiedzy

Tradycyjne metody diagnozy stanów chorobowych na podstawie obrazów medycznych, takich jak zdjęcia rentgenowskie czy obrazy tomografii komputerowej, opierają się na eksperckiej ocenie dokonanej przez lekarza specjalistę. W latach 70. XX wieku pojawiła się możliwość wspomagania procedur medycznych przy użyciu technologii komputerowych.

Rozwój uczenia maszynowego doprowadził do powstania wielu nowych sposobów analizy obrazów medycznych, w tym konwolucyjnych sieci neuronowych oraz uczenia transferowego, które są głównym przedmiotem niniejszej pracy. Wadą wymienionych wyżej technik jest potrzeba zapewnienia dużej ilości danych wejściowych, aby można było trafnie prognozować wyniki. Istnieją jednak metody pozwalające na uzyskanie wiarygodnych wyników nawet przy wykorzystaniu małego zbioru danych. W dziedzinie medycyny jest to szczególnie istotne ze względu na fakt, że wciąż istnieją choroby, w przypadku których dostępne zbiory danych są niewielkie.

W niniejszym rozdziale dokonano zwięzłego przeglądu dokonań w zakresie wykorzystania uczenia maszynowego w analizie obrazów medycznych oraz podsumowano dotychczasowe opracowania dotyczące zbioru danych wykorzystanego w Rozdziale 4.

2.1. Uczenie maszynowe w analizie obrazów medycznych

Pierwsze systemy do wspomagania analizy obrazów medycznych powstały wraz z pojawieniem możliwości skanowania i ładowania obrazów do komputera. Na początkowym etapie rozwoju technologii komputerowych wykorzystywano proste przetwarzanie pikseli obrazów, wykrywanie krawędzi i filtrowanie (Litjens i inni, 2017). Szybko doceniono rolę komputerów w usprawnianiu analizy obrazów medycznych, co doprowadziło do gwałtownego rozwoju w tej dziedzinie. Wciąż jednak potrzebna była duża wiedza ekspercka oraz wkład pracy człowieka, aby wyselekcjonować z obrazów cechy niezbędne do poprawnej klasyfikacji dokonywanej przez komputery.

Pojawienie się uczenia maszynowego umożliwiło efektywne obliczanie parametrów modelu matematycznego na podstawie wyodrębnionych cech i etykiet. Do ogromnej popularności uczenia maszynowego przyczyniła się w dużym stopniu

uniwersalność tej metody. Ze względu na możliwość przetwarzania dowolnych danych uczenie maszynowe znajduje zastosowanie w bardzo wielu dziedzinach, takich jak: matematyka, systemy eksperckie, planowanie i rozwiązywanie problemów, rozpoznawanie mowy, przetwarzanie języka naturalnego czy diagnostyka medyczna. Mimo że wydaje się być stosunkowo nowym odkryciem technologicznym, technika ta jest wykorzystywana już od lat 40. XX wieku (Carbonell, Michalski i Mitchell, 1983). Sama idea sieci neuronowych – jednej z metod uczenia maszynowego, obecnie bardzo popularnej – sięga jednak XVIII wieku, gdyż pierwsze nadzorowane sieci neuronowe były w istocie wariantami regresji liniowej (Schmidhuber, 2015). Na popularności zyskało również uczenie nienadzorowane, które na podstawie wyodrębnionych cech obrazów, jednak bez znajomości ich etykiet, znajduje nieznane wcześniej wzorce (Litjens i inni, 2017).

W analizie i rozpoznawaniu obrazów medycznych przełomowe okazało się głębokie uczenie maszynowe, a w szczególności konwolucyjne sieci neuronowe. Głębokie sieci neuronowe pozwalają zamodelować bardziej złożone problemy niż standardowe sieci neuronowe, charakteryzują się hierarchiczną reprezentacją cech – wyższe warstwy pozwalają na wychwycenie szczegółowych, drobnych cech (np. krawędzie), podczas gdy głębsze warstwy reprezentują cechy bardziej złożone i abstrakcyjne (np. kształty).Przez długi czas głębokie modele nie były wykorzystywane ze względu na trudności w trenowaniu. Główne ograniczenia dla stosowania uczenia głębokiego – niska moc obliczeniowa oraz konieczność zapewnienia bardzo dużej ilości danych wejściowych – obecnie w większości rozwiązywanych problemów nie stanowią przeszkody. Produkowany jest sprzęt coraz lepszej jakości, o coraz większej wydajności, powstały również specjalistyczne biblioteki pozwalające przeprowadzać obliczenia na GPU[1], takie jak CUDA i OpenCL. Dzięki użyciu procesorów graficznych do przyspieszenia obliczeń trenowanie głębokich modeli stało się znacznie mniej czasochłonne. W 2006 roku pojawiła się implementacja konwolucyjnej sieci neuronowej wykorzystującej GPU, która była do 4 razy szybsza niż z wykorzystaniem CPU[2] (Schmidhuber, 2015). Współczesny sprzęt pozwala na przyspieszenie trenowania zazwyczaj od 10 do 30 razy (Litjens i inni, 2017). Doceniono również modele wstępnie trenowane bez nadzoru, w których dostrajanie sieci przebiega w sposób nadzorowany.

[1] ang. *Graphics processing unit* – procesor graficzny
[2] ang. *Central processing unit* – procesor główny

Przykładami takiej architektury są stosy autoendkoderów (ang. *stacked auto-encoders*) i sieci deep belief (ang. *deep belief networks*), jednak nie są one często wykorzystywane ze względu na dużą złożoność i duży nakład pracy programistycznej wymagany do otrzymania zadowalających wyników (Litjens i inni, 2017).

Podczas gdy tworzenie modeli uczenia głębokiego zostało spopularyzowane i ułatwione wraz z pojawieniem się specjalistycznych bibliotek programistycznych takich jak Caffe, Theano, TensorFlow czy Keras[3], głównym celem badań nad sztuczną inteligencją stała się praca nad ulepszaniem otrzymywanych wyników. Nowe odkrycia w tej dziedzinie obejmują wykorzystanie algorytmu wstecznej propagacji błędu[4](ang. *back propagation of error*) i warstw zbierających[5] (ang. *pooling layers*), co usprawniło proces trenowania głębokich modeli. Obecnie modele wykorzystujące GPU, wsteczną propagację oraz warstwy konwolucyjne i zbierające osiągają najlepsze wyniki w dziedzinie analizy obrazów(Schmidhuber, 2015). W 2012 roku po raz pierwszy taki model wygrał konkurs dotyczącym wykrywania obiektów na obrazach – w tym przypadku wykrywania mitozy na obrazach histologicznych raka piersi (Cireşan, Giusti, Gambardella i Schmidhuber, 2013). Wykorzystanie nowych technologii w medycynie jest jednym z najważniejszych osiągnieć w dziedzinie uczenia maszynowego. Częściowa automatyzacja diagnoz medycznych oznaczałaby nie tylko oszczędności finansowe, ale przede wszystkim możliwość pomocy pacjentom, którzy nie mają dostępu do lekarzy specjalistów (Schmidhuber, 2015).

Dzięki bazie ImageNet[6] oraz konkursowi ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[7] naukowcy z całego świata zaczęli rywalizować w tworzeniu algorytmów dających najlepsze wyniki w analizie obrazów. W 2014 roku stworzono pierwsze dużo głębsze[8] modele. Konkurs ILSVRC wygrał model VGG19 składający się z 19 warstw (Simonyan i Zisserman, 2015), zbudowano również 22-warstwową sieć nazywaną GoogLeNet (inna nazwa: Inception), wykorzystującą bloki konwolucyjne – zbiór kilku warstw konwolucyjnych różnych rozmiarów (Szegedy

[3] Caffe, Theano, TensorFlow, Keras – biblioteki programistyczne otwartego źródła zawierające funkcje wspomagające budowę modeli uczenia maszynowego i głębokiego uczenia.
[4] opis znajduje się w Rozdziale 3.3.
[5] opis znajduje się w Rozdziale 3.4.
[6] Ideą projektu ImageNet jest zapewnienie pracownikom naukowym z całego świata łatwo dostępnej bazy obrazów online.
http://image-net.org/about-overview
[7]ILSVRC – konkurs, którego celem jest zachęcenie badaczy do weryfikacji stworzonych przez nich algorytmów klasyfikacji i wykrywania obiektów na dużym i zróżnicowanym zbiorze danych.
[8] tzn. mające więcej warstw ukrytych.

i inni, 2014). Na początku 2015 roku superkomputer zbudowany specjalnie do obsługi algorytmów głębokiego uczenia maszynowego, wykorzystujący bardzo rozbudowane sieci neuronowe zawierające wiele warstw, nowe techniki powiększania zbioru danych i wysokiej jakości obrazy, osiągnął w konkursie ILSVRC wskaźnik błędu na poziomie 5,98% (Wu, Yan, Shan, Dang i Sun, 2015) i był to aktualnie najlepszy wynik. Konkurs w 2015 roku wygrała architektura ResNet składająca się ze 152 warstw, wykorzystująca uczenie resztowe (ang. *residual learning*) (He, Zhang, Ren i Sun, 2015). Rok później naukowcy z firmy Google stworzyli Inception-v3, zmodyfikowaną wersję modelu Inception z 2014 roku (Szegedy, Vanhoucke, Ioffe, Shlens i Wojna, 2016). Wytrenowane na zbiorze ImageNet modeleResNet, Inception-v3 oraz wiele innych są obecnie dostępne do wykorzystania za pomocą biblioteki Keras (patrz: przypis 4.).

Niezależnie od ILSVRC w roku 2017 naukowcy z Uniwersytetu w Stanford stworzyli CheXNet – algorytm, który wykrywa zapalenie płuc na podstawie obrazów rentgenowskich. Wykorzystano w nim sieć konwolucyjną składającą się ze 121 warstw, która na wyjściu podaje prawdopodobieństwo choroby wraz z mapą cieplną wskazującą krytyczne fragmenty obrazu rentgenowskiego, które miały największy wpływ na diagnozę. Wyniki osiągane przez algorytm są lepsze niż wyniki diagnozy postawionej przez lekarzy radiologów (Rajpurkar i inni, 2017). W medycynie dokładność wyników jest kluczowa, bowiem zależy od niej zdrowie i życie pacjenta. Wykorzystanie wciąż rozwijających się technik uczenia maszynowego do rozpoznawania i klasyfikacji obrazów już teraz okazuje się być bardziej korzystne od tradycyjnych ręcznych metod. Uwzględniając fakt, że maszyny się nie męczą, a ryzyko błędnej diagnozy jest mniejsze, uczenie głębokie ma szansę niedługo stać się powszechnie wykorzystywane w służbie zdrowia, nawet jeżeli dana placówka nie ma dostępu do specjalistycznych superkomputerów czy algorytmów takich jak CheXNet. Istnieją podobne rozwiązania dostępne dla każdego, jak chociażby wspomniana wyżej sieć Inception-v3.

Zastosowanie uczenia głębokiego w szeroko pojętej analizie obrazów medycznych jest bogato udokumentowane. Od 2012 roku liczba artykułów dotyczących tego tematu znacząco wzrosła. Litjens i współautorzy (Litjens i inni, 2017) w swoim przeglądzie state-of-the-art podsumowują ponad 300 artykułów, opisując wykorzystywane modele w zależności od:

– typu rozwiązywanego problemu – klasyfikacja, detekcja obiektów, segmentacja,

– technologii pozyskiwania danych wejściowych – rezonans magnetyczny, mikroskop, tomografia komputerowa, rentgen, mammografia i inne,

– badanego organu – mózg, płuca, podbrzusze, serce, kości, siatkówka oka i inne.

Obszerna literatura dotycząca rozpoznawania obrazów medycznych wspomaganego sztuczną inteligencją jest jednocześnie zaletą i wadą. Stosunkowo łatwo znaleźć informacje potrzebne do samodzielnej implementacji wybranego algorytmu głębokiego uczenia maszynowego. Wiedzę na temat poprawy jakości klasyfikacji obrazów można znaleźć głównie w artykułach spoza branży medycznej, jednak metody opisane przez badaczy Lu i Weng nie znajdują zastosowania w głębokich sieciach neuronowych (Lu i Weng, 2007)(Zagoruyko i Komodakis, 2017). Wyjątkiem jest tutaj polski artykuł Alicji Kwaśniewskiej, Anny Giczewskiej i Jacka Rumińskiego, w którym wspomniano i krótko opisano sposoby poprawy wyników modelu, w szczególności zapobiegania nadmiernemu dopasowaniu modelu do danych (Kwaśniewska, Giczewska i Rumiński, 2017). Ze względu na ogromną liczbę publikowanych artykułów znalezienie wszystkich źródeł zawierających szukaną wiedzę stanowi duże wyzwanie.

Obrazowe dane medyczne są coraz łatwiej dostępne dzięki cyfryzacji i publicznemu udostępnieniu danych zebranych przez placówki służby zdrowia na całym świecie. W ostatnich latach znacząco wzrosła liczba ogólnodostępnych danych, zebranych w specjalistyczne bazy danych (Kwaśniewska, Giczewska i Rumiński, 2017). Ponadto rozwinęły się platformy podobne do Kaggle[9], gdzie zbiory danych również są wolnego dostępu. Dzięki temu każdy może wykorzystywać głębokie sieci neuronowe w swojej dziedzinie i nie jest to metoda zarezerwowana tylko dla dużych firm lub organizacji, które są w posiadaniu odpowiednio dużej ilości danych. Przykładem ogólnodostępnej medycznej bazy danych jest baza MedPix[10], utworzona przez The National Library of Medicine w Stanach Zjednoczonych. Wciąż jednak istnieje problem niewystarczającej wielkości zbiorów danych medycznych. Zastosowanie głębokiego uczenia maszynowego w takich przypadkach może prowadzić do otrzymania mało wiarygodnych wyników. Przeszkodą w utworzeniu dużych zbiorów nie jest jednak dostępność samych obrazów, ale opracowywanie etykiet czy opisów dla

[9] Platforma online do samodzielnego rozwijania wiedzy w zakresie uczenia maszynowego i data science. https://www.kaggle.com

[10] https://medpix.nlm.nih.gov/home

danego zbioru (Litjens i inni, 2017). Taka praca z danymi jest nie tylko czasochłonna, ale wymaga też nakładu pracy specjalistów z danej dziedziny, przy czym nigdy nie ma gwarancji poprawności wszystkich etykiet, gdyż nawet eksperci popełniają błędy.

Głównym problemem w trenowaniu modeli uczenia maszynowego na niewielkim zbiorze treningowym jest overfitting, czyli zbyt duże dopasowanie modelu do danych, skutkujące znacznymi błędami w przypadku analizy nowych danych. Najbardziej intuicyjnym rozwiązaniem wspomnianego problemu jest zwiększenie liczebności zbioru treningowego. Inne proponowane rozwiązania (Kwaśniewska, Giczewska i Rumiński, 2017) to między innymi:

- *early termination*[11] – metoda polegająca na zakończeniu trenowania modelu w sytuacji, kiedy poprawa działania sieci jest na tyle mała, że prawie niezauważalna,
- regularyzacja modelu – metoda wykorzystująca zwiększanie współczynnika błędu modelu o kary nałożone na zbyt wysokie wagi,
- *dropout*[12] – metoda polegająca na usunięciu losowo wybranych elementów z sieci neuronowej (wraz z połączeniami) podczas trenowania modelu.

Shen, Wu i Suk (2017) sugerują ponadto następujące techniki:

- inicjalizację – użycie dobrze zaprojektowanej losowej inicjalizacji parametrów modelu oraz powolne zwiększanie parametru impetu[13] wraz z kolejnymi iteracjami,
- wykorzystanie skorygowanej funkcji liniowej jako funkcji aktywacji sieci[14],
- normalizację wsadu[15].

Szeroko stosowaną w modelach regresyjnych i klasyfikacyjnych metodą zapobiegającą nadmiernemu dopasowaniu modelu do danych jest k-krotna walidacja krzyżowa. Polega ona na wielokrotnym trenowaniu i walidacji modelu na różnych podzbiorach zbioru początkowego, tak, że w jednym etapie jeden podzbiór służy za dane testowe, a pozostałe podzbiory to dane treningowe. Niestety ze względu na

[11] *early termination* - z języka angielskiego: wczesne zakończenie.
[12] *dropout* - z języka angielskiego: dziura, luka.
[13] ang. *momentum* - parametr regulujący wpływ zmiany wag w sieci na proces uczenia, patrz Rozdział 3.3.
[14] ta technika została opisana szerzej w Rozdziale 3.3.
[15] ta technika została opisana szerzej w Rozdziale 3.3.

złożoność obliczeniową nie jest ona stosowana w modelach wykorzystujących uczenie głębokie (Lu i Weng, 2007).

Innym problemem, z którym zmagają się twórcy modeli, szczególnie w medycynie, jest brak równowagi w liczebności klas (Litjens i inni, 2017). Skutkuje to często dobrym wytrenowaniem modelu dla bardziej licznej klasy, podczas gdy zazwyczaj bardziej istotne jest dobre przewidywanie etykiet mniej licznej klasy, przedstawiającej obrazy stanów chorobowych. Rozwiązaniem problemu zbyt małego lub niezrównoważonego zbioru danych w problemach dotyczących analizy obrazów mogą być różne sposoby sztucznego powiększania zbioru danych treningowych:

- skalowanie, obrót, odbicia lustrzane, zmiana jasności obrazów,
- wykorzystanie parametrów modeli wytrenowanych na danych niemedycznych do inicjalizacji parametrów tworzonego modelu i następnie ich dostrojenie[16],
- wykorzystanie wariacyjnych autoenkoderów VAE[17] (Kingma i Welling, Auto-encoding variational bayes, 2013),
- wykorzystanie generatywnych sieci przeciwstawnych GAN[18](Goodfellow i inni, 2014).

Wariancyjny autoenkoder to jednokierunkowa sieć składająca się z kodera, warstwy reprezentacji oraz dekodera. Zadaniem autoenkoderów jest rekonstrukcja sygnału wejściowego (Kingma i Welling, Auto-encoding variational bayes, 2013). Generatywne sieci neuronowe składają się z dwóch równolegle trenowanych modeli: generatywnego i dyskryminacyjnego. Pierwszy z nich generuje nowe dane, początkowo losowo, a drugi dokonuje selekcji wygenerowanych obrazów, odrzucając te, które jest w stanie odróżnić od obrazów prawdziwych. Wygenerowane obrazy, które pomyślnie przejdą selekcję służą do dalszego trenowania modelu generatywnego (Goodfellow i inni, 2014). Brakuje jednak dokumentacji efektów wykorzystania VAE i sieci GAN w generowaniu większej ilości danych medycznych na potrzeby późniejszej klasyfikacji również przeprowadzanej z użyciem głębokiego uczenia maszynowego.

[16] dostrojenie parametrów to znalezienie takich ich wartości, dla których model osiąga najlepsze wyniki; następuje podczas trenowania modelu lub jego części
[17] ang. *variational auto-encoders*
[18] ang. *generative adversarial networks*

Znaczącą popularnością cieszy się uczenie transferowe. Jego zamysłem jest zastosowanie wytrenowanych, dających dobre wyniki modeli sieci neuronowych dla danych innych niż te użyte do uczenia modelu, nawet jeśli nowe dane dotyczą innej, słabo spokrewnionej dziedziny. Inspiracją uczenia transferowego jest zaobserwowana u ludzi zdolność zastosowania zdobytej uprzednio wiedzy do szybszego i sprawniejszego rozwiązywanie nowych problemów (Pan i Yang, 2009). Dzięki możliwości wykorzystania wiedzy nabytej na innym, dużo większym zbiorze, uczenie transferowe jest rozwiązaniem problemu niedostatecznie licznego zbioru danych treningowych i dzięki temu jest tak cennym odkryciem w dziedzinie uczenia maszynowego w medycynie (Kwaśniewska, Giczewska i Rumiński, 2017). Utworzenie dużych, ogólnodostępnych i szczegółowo oraz rzetelnie opisanych zbiorów danych medycznych, podobnych do ImageNet, jest jednak równie istotne, jak opracowywanie nowych algorytmów (Shin i inni, 2016).

W 2017 roku Sergey Zagoruyko i Nikos Komodakis przedstawili metodę poprawy jakości klasyfikacji poprzez nauczenie sieci konwolucyjnej „zwracania szczególnej uwagi" na pewne cechy danych, którą nazwano transferem uwagi. Idea jest wzorowana na sposobie postrzegania obrazów przez ludzi. Opisywana „uwaga" to coś więcej niż ekstrapolacja cech charakterystycznych dla poszczególnych klas (w przypadku klasyfikacji), jak to się dzieje w każdej konwolucyjnej sieci neuronowej. Celem transferu uwagi jest nauczenie modelu wykrywania cech danych wejściowych, na płaszczyźnie aktywacyjnej (np. kształty) oraz gradientowej (np. barwa pikseli), których małe zmiany mają znaczny wpływ na wyniki osiągane przez model (Zagoruyko i Komodakis, 2017). Wytrenowany w ten sposób model może służyć za źródło wiedzy dla modelu-ucznia zastosowanego do pokrewnych, ale nie identycznych danych i zmniejszyć czas potrzebny do jego wyuczenia. Sam pomysł bardzo przypomina opisane w poprzednim akapicie uczenie transferowe, jednak obejmuje ekstrakcję bardziej złożonych cech z danych wejściowych.

Wraz z rozwojem technologii pojawiła się możliwość tworzenia zespołów modeli, które mogłyby czerpać informacje z więcej niż jednego typu danych wejściowych. Jako że w medycynie duże, dobrze opisane zbiory specjalistycznych danych wciąż są trudno dostępne w przypadku niektórych chorób, podjęto próby wzbogacenia posiadanych źródeł informacji przez dodanie opisów z raportów lekarskich do obrazów medycznych (Litjens i inni, 2017).

Kumar i inni wykorzystali zespół konwolucyjnych sieci neuronowych zamiast pojedynczej sieci, aby poprawić jakość klasyfikacji obrazów medycznych. Autorzy uzasadniają użycie takiej właśnie architektury faktem, że konwolucyjne sieci neuronowe mogą nie osiągać pożądanych wyników, kiedy dane wejściowe są zróżnicowane i ograniczone. Płytkie sieci CNN mogą mieć zbyt mało warstw, aby dobrze reprezentować cechy obrazów, podczas gdy skomplikowane sieci CNN mogą być zbyt wrażliwe na drobne różnice pomiędzy obrazami. Rozwiązaniem tego problemu jest wytrenowanie kilku różniących się między sobą konwolucyjnych sieci neuronowych i połączenie dawanych przez nie wyników(Kumar, Kim, Lyndon, Fulham i Feng, 2016).

Techniki sztucznej inteligencji znajdują coraz szersze zastosowanie w medycynie. Na zainteresowanie zasługuje idea wykorzystania głębokiego uczenia maszynowego do zdalnej analizy stanu zdrowia pacjentów w podeszłym wieku, którzy z powodu niedyspozycji fizycznych nie są w stanie dotrzeć osobiście na badania lub jest to bardzo utrudnione (Kwaśniewska, Giczewska i Rumiński, 2017). Diagnoza miałaby się opierać na analizie wizualnej i termograficznej – pomiarze i zapisie pulsu czy temperatury ciała. Dzięki takiemu rozwiązaniu zwiększa się komfort badania przy jednoczesnym zmniejszeniu ryzyka urazu w drodze do punktu badań lub zarażenia od innych pacjentów w danej placówce.

Rosnące możliwości wykorzystania sztucznej inteligencji do zdalnej diagnozy pociągają za sobą konieczność minimalizacji błędów w analizie i rozpoznawaniu obrazów co jest przedmiotem dalszych badań w tej dziedzinie. Metody uczenia głębokiego wciąż są postrzegane jako „czarne skrzynki". Można zwizualizować aktywacje poszczególnych warstw, ale zrozumienie ich działania nie jest intuicyjne. W medycynie, gdzie błędna diagnoza sugerowana przez stworzony model może oznaczać śmierć pacjenta. Nie wystarczy, aby model osiągał bardzo dobre wyniki, potrzebna jest jeszcze możliwość uzasadnienia tych wyników (Litjens i inni, 2017). Jest to temat, który będzie miał ogromne znaczenie w dalszym rozwoju uczenia maszynowego w analizie danych medycznych.

2.2. Dotychczasowe prace wykorzystujące wybrany zbiór danych

Rentgen płuc to najbardziej powszechne narzędzie w diagnozie zapalenia płuc (Litjens i inni, 2017). Stworzenie ogólnodostępnego systemu komputerowego do

wspomagania diagnozy umożliwiłoby szybsze rozpoczęcie leczenia i byłoby szansą na przeżycie dla tych, którzy nie mają dostępu do lekarzy specjalistów. W Rozdziale 4. przedstawione zostały metody klasyfikacji zdjęć rentgenowskich płuc przy pomocy powszechnie dostępnych narzędzi, niewymagających wyspecjalizowanego sprzętu. Zbiór danych[19], analizowany w niniejszej pracy został dotychczas użyty trzykrotnie[20] do budowy modeli głębokiego uczenia w celu klasyfikacji obrazów rentgenowskich.

W pierwszym[21] z opracowań autor wykorzystuje konwolucyjną sieć neuronową składającą się z trzech podwójnych warstw konwolucyjnych, jednej warstwy gęstej[22] oraz gęsto połączonego klasyfikatora[23]. Osiągnięto trafność klasyfikacji na poziomie 94,71%.

Drugie[24] opracowanie jest bardziej obszerne. Podkreślono i rozwiązano problem niezrównoważonego zbioru danych poprzez zwiększenie liczebności klasy 0 oznaczającej zapalenie płuc. Oprócz stworzenia i wytrenowania konwolucyjnej sieci neuronowej, składającej się z trzech warstw konwolucyjnych, warstwy gęstej oraz gęsto połączonego klasyfikatora autor zamieszcza również macierz błędów klasyfikacji dla wytrenowanego modelu oraz wizualizację obrazów wejściowych po przetworzeniu przez kolejne warstwy konwolucyjne sieci. Zaprojektowana architektura charakteryzuje się dokładnością klasyfikacji 92,07%.

W trzecim[25] opracowaniu autor wykorzystuje konwolucyjną sieć neuronową z trzema warstwami konwolucyjnymi, czteroma warstwami gęstymi i gęstym klasyfikatorem, która osiąga wynik 95,32%.

W niniejszej pracy standardowa architektura sieci konwolucyjnej z warstwami zbierającymi, normalizacją wsadu i gęsto połączonym klasyfikatorem zostanie porównana z ogólnodostępnymi rozbudowanymi modelami, takimi jak wspomniane wcześniej Inception, ResNet, VGG, wykorzystanymi w uczeniu transferowym na małym zbiorze danych.

---

[19] Wybrany do opracowania zbiór danych to baza zdjęć rentgenowskich płuc wraz z plikiem przypisującym zdjęciom etykiety: 0 – zapalenie płuc, 1 – płuca zdrowe. Zbiór został pobrany z platformy Kaggle (patrz: przypis 9., s. 7): https://www.kaggle.com/parthachakraborty/pneumonia-chest-x-ray [dostęp: 5.06.2019]

[20] stan na 5.06.2019.

[21] https://www.kaggle.com/parthachakraborty/classificationpneumonia-binary [dostęp: 5.06.2019]

[22] szerszy opis znajduje się w Rozdziale 3.4.

[23] szerszy opis znajduje się w Rozdziale 3.4.

[24] https://www.kaggle.com/dmitrypukhov/chest-x-rays-cnn-and-it-s-subconscious [dostęp: 5.06.2019]

[25] https://www.kaggle.com/swarajp/pneumonia-detection [dostęp: 5.06.2019]

3. Uczenie maszynowe

Niniejszy rozdział rozpoczyna wprowadzenie w teorię uczenia maszynowego, sieci neuronowych i głębokich sieci neuronowych. Jest ona potrzebna do zrozumienia bardziej skomplikowanych modeli, których implementacja jest głównym tematem Rozdziału 4. Opisany jest również mechanizm klasyfikacji w uczeniu maszynowym oraz miary wykorzystywane do oceny wydajności klasyfikatorów. W dalszej części znajduje się szczegółowy opis konwolucyjnych sieci neuronowych oraz ich elementów składowych. W końcowej części rozdziału objaśniono szerzej mechanizm uczenia transferowego oraz dokonano charakterystyki wybranych pięciu modeli udostępnionych w bibliotece Keras, które uwzględniono w analizie porównawczej wyników klasyfikacji binarnej osiąganych na małym zbiorze danych medycznych.

3.1. Podstawy uczenia maszynowego

Programowanie opiera się o trzy elementy: dane wejściowe, zbiór reguł określających sposób, w jaki dane mają zostać przetworzone, oraz dane wyjściowe – odpowiedzi. Klasyczne programowanie polega na uzyskiwaniu odpowiedzi poprzez zastosowanie ręcznie określonych reguł do danych wejściowych. Wymaga to zazwyczaj eksperckiej wiedzy dziedzinowej i jest czasochłonne, a cały ciężar pracy spoczywa na człowieku. Uczenie maszynowe w porównaniu do stosowanych wcześniej metod pozwala wykorzystać moc obliczeniową komputerów do odkrywania reguł na podstawie dostarczonych danych i odpowiedzi (Chollet, 2019). Niestety wraz ze wzrostem skomplikowania używanych metod coraz trudniej interpretować stworzone przez komputer reguły.

Ze względu na gwałtowny wzrost ilości danych, spowodowany w dużej mierze rozpowszechnieniem Internetu oraz zwiększeniem pamięci komputerów, ręczne przetwarzanie gromadzonych informacji przestało być opłacalne. Uczenie maszynowe szybko zyskało na popularności i zaczęto je szeroko wykorzystywać w wielu różnych dziedzinach (Carbonell, Michalski i Mitchell, 1983). Główne zadania rozwiązywane przy pomocy uczenia maszynowego to:

- regresja: określenie wzajemnej zależności między zmiennymi,
- klasyfikacja: przyporządkowanie elementów do odpowiadającym im klas,

– grupowanie: utworzenie skupisk elementów podobnych do siebie, charakteryzujących się podobnymi cechami.

W problemach regresji i klasyfikacji stosuje się metody uczenia nadzorowanego (ang. *supervised learning*). Wybrany do rozwiązania danego zadania algorytm otrzymuje na wejściu dane oraz odpowiedzi, a jego wynikiem jest zbór reguł, otrzymany poprzez minimalizację błędu pomiędzy odpowiedzią prawdziwą i przewidywaną przez algorytm (Schmidhuber, 2015). Grupowanie to zadanie rozwiązywane metodami uczenia nienadzorowanego (ang. *unsupervised learning*), w którym algorytm wyłącznie na podstawie danych wejściowych, bez znajomości odpowiedzi, definiuje skupiska elementów podobnych do siebie – komputer przejmuje całą pracę wykonywaną dotąd ręcznie przez człowieka.

Największą trudnością podczas wykorzystywania uczenia maszynowego do rozwiązania danego zadania jest zebranie i odpowiednie przygotowanie danych (ang. *preprocessing*). Jest to najważniejszy i najbardziej czasochłonny etap tworzenia algorytmu. Gromadzenie danych można przeprowadzić samodzielnie – ręcznie lub z wykorzystaniem stworzonych przez siebie lub ogólnodostępnych narzędzi – lub wykorzystać dane dostępne publicznie, których jest coraz więcej. (Harrington, 2012). Następnie należy przystosować dane tak, aby mogły zostać przetworzone przez algorytm. Oznacza to często ujednolicenie formatu danych oraz zamianę danych tekstowych na liczbowe. Kolejne kroki to wybranie odpowiedniej metody do rozwiązania danego zadania, wytrenowanie modelu i przetestowanie jego działania.

3.2. Klasyfikacja

Zadanie klasyfikacyjne polega na przyporządkowaniu danych do uprzednio zdefiniowanych klas, grup różniących się pomiędzy sobą i skupiających elementy podobne. W przypadku analizy surowych danych, bez posiadania dodatkowej wiedzy na ich temat, klasyfikację poprzedza proces grupowania lub ręcznego definiowania reguł klasyfikacji.

W metodach nadzorowanych standardową praktyką jest podział zbioru danych na trzy rozłączne podzbiory: treningowy, walidacyjny i testowy. Proces trenowania polega na wielokrotnym przetwarzaniu danych przez algorytm i odpowiedniej modyfikacji parametrów. Pojedyncza iteracja algorytmu na zbiorze treningowym nazywana jest epoką. Pod koniec każdej epoki na zbiorze walidacyjnym obliczany jest

błąd pomiędzy odpowiedziami prawdziwymi i przewidywanymi na podstawie uzyskanych w tej epoce parametrów. Odpowiedzi w zadaniu klasyfikacyjnym nazywane są etykietami – są to oznaczenia klasy, do której należy dany element. Zbiór testowy służy do sprawdzenia faktycznej zdolności algorytmu do uogólniania, czyli ostatecznej oceny działania na zupełnie nieznanych danych (Chollet, 2019).

Ze względu na liczbę klas rozróżnia się dwa rodzaje klasyfikacji: klasyfikację binarną oraz klasyfikację wieloklasową. W przypadku klasyfikacji binarnej określa się jakąś cechę, którą dany element posiada lub nie. Jeżeli posiada, zostaje zakwalifikowany do klasy pozytywnej, jeżeli nie – do negatywnej. Problem rozwiązywany w Rozdziale 4. niniejszej pracy dotyczy klasyfikacji obrazów rentgenowskich ze względu na występowanie zapalenia płuc. Jako klasę pozytywną określono „zapalenie płuc", a jako klasę negatywną – „płuca zdrowe". Dany obraz może zostać przypisany do jednej z tych dwóch klas, prawidłowo lub nieprawidłowo:

- TP (ang. *true positive*) – obraz przedstawiający zapalenie płuc zaklasyfikowany prawidłowo, wynik prawdziwie pozytywny,
- FP (ang. *false positive*) – obraz przedstawiający zapalenie płuc zaklasyfikowany jako płuca zdrowe, wynik fałszywie pozytywny,
- FN (ang. *false negative*) – obraz przedstawiający płuca zdrowe zaklasyfikowany jako zapalenie płuc, wynik fałszywie negatywny,
- TN (ang. *true negative*) - obraz przedstawiający płuca zdrowe zaklasyfikowany prawidłowo, wynik prawdziwie negatywny.

W dalszej części pracy zostaną użyte skróty angielskie ze względu na ich powszechne wstępowanie w literaturze i możliwość pomyłki w interpretacji w przypadku wprowadzenia skrótów polskich. W tabeli 1. przedstawiono możliwe wyniki klasyfikacji. Takie zestawienie nazywa się macierzą pomyłek[26] (ang. *confusion matrix*).

[26] inne spotykane polskie tłumaczenia to: macierz błędów, tablica pomyłek, tablica błędów.

Tabela 1. Macierz pomyłek w klasyfikacji binarnej.

Klasa przewidziana	**Klasa prawdziwa**	
	Zapalenie płuc	Płuca zdrowe
Zapalenie płuc	**TP**	**FN**
Płuca zdrowe	**FP**	**TN**

Źródło: opracowanie własne.

Metryki obliczane dla klasyfikacji binarnej na podstawie macierzy pomyłek to:

1. trafność, dokładność (ang. *accuracy*) – jedna z najczęściej wykorzystywanych miar, określa ogólną efektywność klasyfikatora poprzez stosunek etykiet poprawnych do etykiet błędnych, obliczana według wzoru:

$$\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN},$$

2. precyzja (ang. *precision*) – określa zgodność etykiet klasy prawdziwej z etykietami przypisanymi przez klasyfikator, obliczana według wzoru:

$$\text{precision} = \frac{TP}{TP + FP},$$

3. czułość (ang. *recall, sensitivity*) – określa zdolność klasyfikatora do rozpoznawania elementów należących do klasy pozytywnej, obliczana według wzoru:

$$\text{recall} = \frac{TP}{TP+FN},$$

4. swoistość (ang. *specificity*) –określa zdolność klasyfikatora do rozpoznawania elementów należących do klasy negatywnej, obliczana według wzoru:

$$\text{specificity} = \frac{TN}{FP+TN},$$

5. Miara F (ang. *F-score*) –określa ogólną efektywność klasyfikatora, średnia harmoniczna precyzji i czułości, obliczany według wzoru:

$$\text{F-score} = \frac{2TP}{2TP+FP+FN},$$

(Sokolova i Lapalme, 2009).

W niniejszej pracy do oceny algorytmów klasyfikacji zostaną wykorzystane cztery miary: trafność, precyzja, czułość i miara F. Należy zwrócić uwagę na fakt, że w przypadku silnie niezrównoważonego zbioru danych miara trafności daje zafałszowane wyniki[27], dlatego zaleca się korzystać z innych wskaźników.

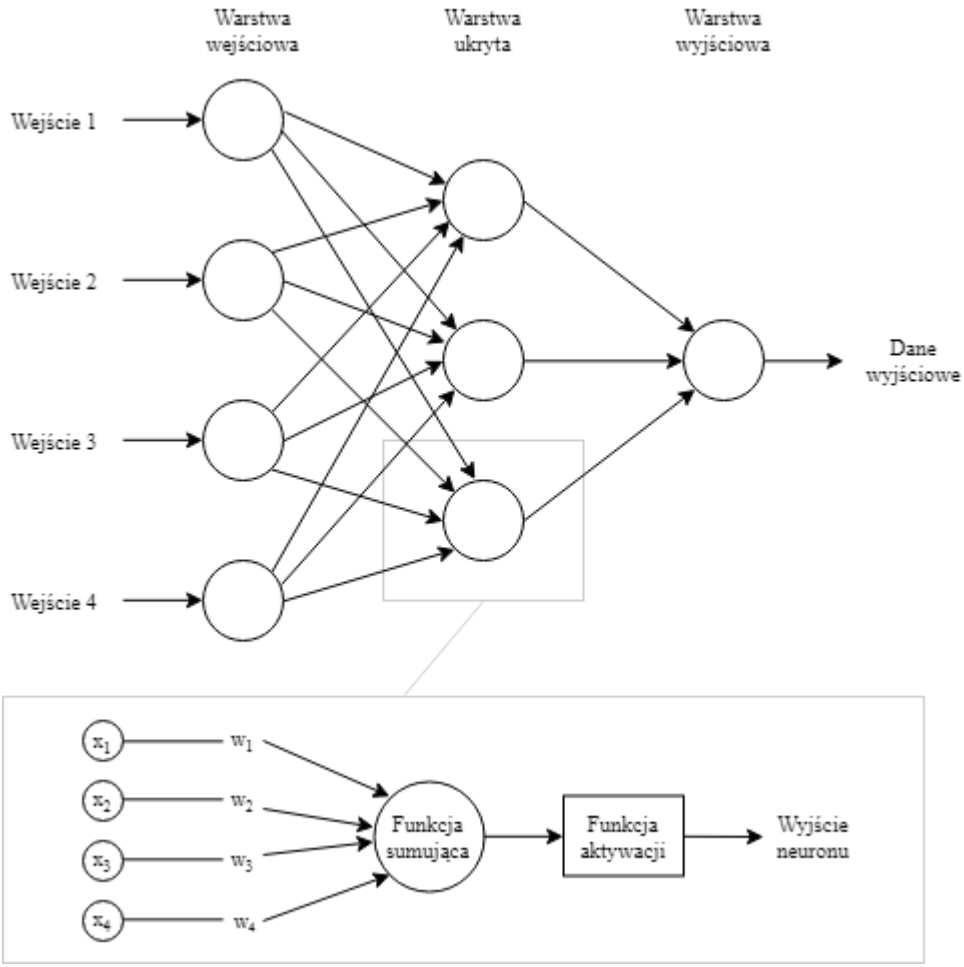
3.3. Sieci neuronowe

Najbardziej podstawowa sieć neuronowa, składająca się jedynie z warstwy wejściowej i wyjściowej, nazywana jest perceptronem jednowarstwowym[28]. Sieci, które posiadają co najmniej jedną warstwę poza wejściową i wyjściową, noszą nazwę perceptronów wielowarstwowych, natomiast dodatkowe warstwy są określane jako warstwy ukryte (Shen, Wu i Suk, 2017). Poszczególne warstwy składają się z neuronów. Neurony każdej warstwy połączone są ze wszystkimi neuronami warstw sąsiadujących, ale nie istnieją połączenia między neuronami tej samej warstwy. Każdemu połączeniu przypisana jest waga. Rysunek 1. przedstawia schemat wielowarstwowej sieci neuronowej oraz działanie pojedynczego neuronu. Podstawowe elementy, z których składa się sieć neuronowa, to:

- wejścia$\mathbf{X} = [\, x_1, \ldots, x_n \,]$ (ang. *inputs*) – miejsca wprowadzania sygnałów zewnętrznych do sieci,
- wagi $\mathbf{W} = [\, w_1, \ldots, w_n \,]$ (ang. *weights*) – określające ważność danego wejścia,
- obciążenie **b** (ang. *bias*) – wartość skalarna dodawana do wartości wejściowej, dzięki której sieć może uczyć się na podstawie słabych sygnałów,
- funkcja aktywacji **act** (ang. *activation function*) –określająca, w jaki sposób na podstawie wejść **X** i wag **W** będzie obliczane pobudzenie **a**,
- pobudzenie **a** (ang. *activation value*) – określające aktywność danego neuronu,
- wartość wyjścia $\mathbf{Y} = [\, y_1, \ldots, y_n \,]$ (ang. *output*),
- funkcja przenosząca **f** (ang. *transfer function*) – określająca stan wyjścia **Y** na podstawie pobudzenia **a**

(Krawiec i Stefanowski, 2003).

[27] jest to tzw. *false accuracy*, fałszywa trafność.
[28] warstwa wejściowa zwykle nie jest wliczana do ogólnej liczby warstw sieci.



Rysunek 1. Schemat sieci neuronowej oraz pojedynczego neuronu.
(źródło: opracowanie własne)

Główną funkcją neuronu jest sumowanie sygnałów wejściowych. Ostateczny wynik obliczany jest za pomocą funkcji przenoszącej, na podstawie aktywacji neuronu:

$$a = \sum_{i=1}^{n} x_i * w_i + b,\ Y = f(a)\ .$$

Najczęściej stosowane funkcje przenoszące to:

- funkcja liniowa:

$$f(a) = a\ ,$$

- tangens hiperboliczny, wykorzystywany głównie w klasyfikacji binarnej:

$$f(a) = tgh(a)$$

– funkcja sigmoidalna, wykorzystywana w klasyfikacji binarnej:

$$f(a)= \frac{1}{1+ \exp(-a)},$$

– funkcja softmax, wykorzystywana w klasyfikacji wieloklasowej:

$$f(a) = \frac{\exp(a)}{\sum_{j=0}^{n} \exp(a_j)},$$

– skorygowana jednostka liniowa ReLU (ang. *rectified linear unit*):

$$f(a)=\max(0, a)$$

(Krawiec i Stefanowski, 2003) (Litjens i inni, 2017). Standardowo w zadaniach klasyfikacji w warstwach wyjściowych sieci stosuje się funkcje sigmoidalną i softmax.

Celem trenowania sieci neuronowej jest znalezienie takich wag, aby różnice pomiędzy danymi wyjściowymi **Y** oraz spodziewanymi odpowiedziami **Z** były jak najmniejsze, co powinno zapewnić dobrą zdolność modelu do generalizacji na nowych danych. Funkcję, według której obliczany jest całkowity błąd sieci, określa się jako funkcję straty. Najczęściej stosowane funkcje straty to:

– błąd średniokwadratowy MSE (ang. *mean square error*):

$$MSE= \frac{1}{n}\sum_{i=1}^{n} (z_i-y_i)^2,$$

– średni błąd bezwzględny MAE (ang. *mean absolute error*):

$$MAE= \frac{1}{n}\sum_{i=1}^{n} |z_i-y_i|,$$

– średni bezwzględny błąd procentowy MAPE (ang. *mean absolute percentage error*):

$$MAPE= \frac{100\%}{n}\sum_{i=1}^{n} \left(\frac{z_i-y_i}{y_i}\right),$$

– entropia krzyżowa (ang. *crossentropy*) – binarna (ang. *binary*) w klasyfikacji dwuklasowej i kategoryzacyjna (ang. *categorical*) w klasyfikacji wieloklasowej:

$$CE= -\frac{1}{n}\sum_{i=1}^{n} [z_i \log(y_i)+(1-z_i)\log(1-y_i)]$$

(Patterson i Gibson, 2018) (Changhau, 2017).

Wraz z funkcją straty, której minimalizacja jest celem trenowania sieci neuronowej, należy określić sposób zmiany wag na końcu każdej epoki, czyli wybrać algorytm optymalizacji. Najczęściej stosuje się algorytm losowego spadku wzdłuż gradientu (ang. *stochastic gradient descent, SGD*). Gradient to uogólnienie koncepcji różniczkowania funkcji wieloargumentowych na tensory, czyli macierze N-wymiarowe. Działanie sieci neuronowej opiera się na łańcuchu operacji na tensorach. Obliczenie pochodnej takiego łańcucha funkcji sprowadza się do wykorzystania algorytmu propagacji wstecznej, nazywanego także odwrotnym różniczkowaniem, czyli obliczenia pochodnej funkcji wielokrotnie złożonej według reguły matematycznej:

$$[\,f(g(x))]' = f'(g(x)) * g'(x)$$

(Chollet, 2019).

Mechanizm SGD opiera się na losowym wybraniu małego podzbioru danych, na którym obliczany jest gradient. Następnie wagi modyfikowane są o wartość nazywaną stałą uczenia (ang. *learning rate*) w kierunku przeciwnym do gradientu (Litjens i inni, 2017). Jest to uproszczenie podstawowego algorytmu spadku wzdłuż gradientu, gdzie gradient oblicza się na całym zbiorze danych (Chollet, 2019).

W Rozdziale 4. niniejszej pracy wykorzystano również algorytm optymalizacji o nazwie Adam (akronim od ang. *adaptive moment estimation*). Łączy on zalety dwóch wcześniej opracowanych algorytmów:

- AdaGrad (akronim od ang. *Adaptive Gradient Algorithm*), w którym stałe uczenia są obliczane niezależnie dla każdego parametru (Duchi, Hazan i Singer, 2011),
- RMSProp (akronim od ang. *Root Mean Square Propagation*), w którym również wykorzystuje się różne stałe uczenia dla różnych parametrów, przy czym są one modyfikowane na podstawie średniej wielkości gradientu w poprzednich epokach trenowania. (Tieleman i Hinton, 2012).

Zmiany stałych uczenia w algorytmie Adam również są różne dla indywidualnych parametrów, opierają się na średniej poprzednich zmian gradientu oraz ponadto na średniej wariancji gradientu. Pomimo faktu, iż wykorzystanie optymalizacji Adam pozwala osiągnąć lepsze wyniki, niż przy zastosowaniu SGD, czy bardziej

skomplikowanych RMSProp i AdaGrad, w niektórych przypadkach najlepszy okazuje się algorytm losowego spadku wzdłuż gradientu (Kingma i Ba, 2015).

3.4. Konwolucyjne sieci neuronowe

Sieci neuronowe, które składają się z wielu warstw ukrytych, określane są jako sieci głębokie. Przez długi czas głębokie sieci neuronowe nie były wykorzystywane ze względu na niewystarczającą moc obliczeniową komputerów, co znacząco zmieniło się na początku XXI wieku (Litjens i inni, 2017). Obecnie sieci głębokie lub ich zespoły są szeroko wykorzystywane i osiągają najlepsze wyniki w wielu konkursach, takich jak ILSVRC czy konkursy na platformie Kaggle. W zadaniach dotyczących obrazów najczęściej stosuje się konwolucyjne sieci neuronowe (Schmidhuber, 2015). Działają one na tzw. mapach cech, czyli trójwymiarowych tensorach, gdzie pierwszym wymiarem jest wysokość, drugim szerokość, a trzecim głębia obrazu, czyli liczba kanałów RGB.

Najważniejszym mechanizmem wykorzystywanym w warstwach konwolucyjnych, podstawowych składowych konwolucyjnych sieci neuronowych, jest operacja matematyczna określana jako konwolucja (również: splot, mnożenie splotowe funkcji). W przypadku przetwarzania obrazów **m** filtrów $\mathbf{W} = [\mathbf{w_1}, \dots, \mathbf{w_m}]$ określonych rozmiarów, nazywanych jądrami konwolucji, jest przesuwanych po macierzy pikseli obrazu wejściowego (Schmidhuber, 2015). Rezultatem dodania obciążenia **b** do wyniku konwolucji jest mapa cech $\mathbf{X_m}$, która następnie jest przetwarzana przez funkcję aktywacji **f** danej warstwy konwolucyjnej **c-1** i przekazywana jako dane wejściowe dla kolejnej warstwy **c**:

$$x_n^c = f(w_n^{c-1} * x^{c-1} + b_n^{c-1})$$

(Litjens i inni, 2017).

Dzięki uwzględnieniu nie tylko pojedynczych pikseli, ale również ich otoczenia, oraz współdzieleniu wag w obrębie warstw, sieci konwolucyjne są w stanie nauczyć się lokalnych wzorców i tworzyć ich hierarchię. Wykorzystanie tych samych wag, reprezentowanych w sieciach CNN przez filtry, pozwala na wykrywanie tych samych cech w danych wejściowych. Neurony początkowych warstw są aktywowane przez małe lokalne wzorce (np. krawędzie), podczas gdy neurony głębszych warstw

rozpoznają większe, bardziej abstrakcyjne struktury (Chollet, 2019). Dzięki temu konwolucyjne sieci neuronowe osiągają tak dobre wyniki w dziedzinie analizy obrazów.

W sieciach CNN wykorzystuje się dodatkowo warstwy zbierające, których zadaniem jest zmniejszenie wymiarowości modelu poprzez zebranie wartości sąsiednich pikseli w jedną, zazwyczaj poprzez uśrednienie lub wartość maksymalną. Dzięki temu znacząco zmniejsza się liczba parametrów modelu, co oznacza przyspieszenie procesu trenowania (Litjens i inni, 2017).

Standardowa architektura konwolucyjnych sieci neuronowych opiera się na warstwach konwolucyjnych i następujących po nich warstwach zbierających oraz warstwach gęsto połączonych. Te ostatnie, będące warstwami standardowo wykorzystywanymi w prostych sieciach neuronowych, gdzie każdy neuron połączony jest ze wszystkimi neuronami warstw sąsiednich, wykorzystywane są w sieciach CNN m. in. jako końcowe klasyfikatory[29] (Litjens i inni, 2017).

Obecnie standardowo wykorzystywanym elementem sieci głębokich jest warstwa normalizująca. Ze względu na duży rozmiar zbioru danych potrzebny do efektywnego trenowania sieci i związane z tym koszty obliczeniowe odstąpiono od trenowania na wszystkich danych jednocześnie na rzecz trenowania w każdej epoce kolejno na małych podzbiorach. Pojedynczy podzbiór określany jest jako wsad. Normalizacja wsadu polega na ujednoliceniu całego wsadu poprzez odjęcie od każdego elementu średniej z wsadu i podzielenie przez wariancję wsadu. Dzięki temu sieć może lepiej wychwycić wspólne cechy danych wejściowych (Ioffe i Szegedy, 2015).

Końcowym etapem opracowywania modeli uczenia głębokiego jest etap strojenia parametrów. Nawet niewielka zmiana stałej uczenia czy odsetka połączeń, które powinny być usunięte przy zastosowaniu techniki regularyzacji *dropout*, mogą znacząco wpłynąć na wyniki modelu, podobnie zmiana liczby lub rodzaju warstw ukrytych. Nie istnieją dokładne wytyczne dotyczące ustalania wartości parametrów, a jedynie porady, wskazujące ich zalecane wartości lub strategie modyfikacji, często różniące się w zależności od rodzaju wykorzystywanego modelu (Chollet, 2019). Wciąż rosnące możliwości komputerów pozwalają na wielokrotne wytrenowanie różnych modeli dla różnych wartości parametrów i wybranie najlepszego z nich, co jednak nie zmienia faktu, iż głębokie uczenie jest w dużym stopniu nauką eksperymentalną.

[29] Stąd nazwa „gęsto połączony klasyfikator", wykorzystana w rozdziale 2.2.

3.5. Uczenie transferowe

Powszechne wykorzystanie modeli uczenia maszynowego do rozwiązywania problemów z różnych dziedzin było motywacją do poszukiwania sposobów na tworzenie bardziej uniwersalnych modeli, które nie musiałyby być dla nowych danych trenowane od podstaw. Rezultatem poszukiwań jest uczenie transferowe, którego inspiracją była ludzka możliwość zastosowania zdobytej wiedzy do rozwiązywania nowych zadań (Pan i Yang, 2009). Jest to technika polegająca na ponownym wykorzystaniu złożonych modeli, wytrenowanych na dużym i dobrze opisanym zbiorze danych. W uczeniu transferowym wyróżnia się dwie główne strategie:

- wykorzystanie wytrenowanego uprzednio modelu jako ekstraktora cech z nowego zbioru danych, a następnie klasyfikatora, w którym danymi wejściowymi są uzyskane cechy,
- dostrojenie ostatnich warstw wytrenowanego uprzednio modelu, odpowiedzialnych za ostateczną klasyfikację danych, na nowym zbiorze

(Litjens i inni, 2017).

Możliwe jest również wytrenowanie stworzonego uprzednio modelu od podstaw na nowym zbiorze danych, jednak to rozwiązanie nie jest często wykorzystywane, ze względu na fakt, że wytrenowanie rozbudowanych modeli – takich jak Inception-v3 czy ResNet, dostępnych w pakiecie Keras – zajmuje często dni, a nawet tygodnie. Należy jednak zwrócić uwagę na fakt, że jeżeli źródłowy zbiór danych oraz zbiór danych dla nowego problemu zbytnio się od siebie różnią, może wystąpić zjawisko tzw. negatywnego transferu, kiedy osiągnięte wyniki są dużo gorsze w przypadku uczenia transferowego niż bez jego wykorzystania (Pan i Yang, 2009). Poniżej przedstawiono krótkie charakterystyki pięciu modeli wykorzystanych do uczenia transferowego w Rozdziale 4. niniejszej pracy.

ResNet (He, Zhang, Ren i Sun, 2015)

Wraz z rozpowszechnieniem uczenia głębokiego podejmowano próby tworzenia modeli składających się z coraz większej liczby warstw w nadziei, że taka architektura pozwoli lepiej rozwiązać złożone problemy. Niestety okazało się, że tak duże rozmiary modelu powodują wyższy błąd trenowania, co negatywnie wpływa na wyniki.

W modelach ResNet (akronim od ang. *residual networks*) wykorzystywane są tzw. połączenia szczątkowe (ang. *residual connections*), które pomijają wybrane warstwy. Wejścia pomijanej warstwy (lub kilku warstw) są dodawane do wyjścia tych warstw na końcu połączenia. Dzięki temu zmniejsza się błąd wynikający z nadmiernego rozmiaru modelu. Połączenia szczątkowe mogą być wykorzystywane zarówno dla warstw gęstych, jak i warstw konwolucyjnych. Model ResNet zdobył pierwsze miejsce w zadaniu klasyfikacji w konkursie ILSVRC 2015. Tabela 2. przedstawia architekturę modeli ResNet w zależności od ich rozmiaru. Poszczególne warstwy charakteryzowane są przez wielkość jądra konwolucji oraz liczbę filtrów (np. 3x3, 64). W Rozdziale 4. zostanie wykorzystany model składający się z 50 warstw.

Tabela 2. Architektura modeli ResNet.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Źródło: (He, Zhang, Ren i Sun, 2015).

VGG[30] (Simonyan i Zisserman, 2015)

Modele VGG zostały skonstruowane w celu zbadania wpływu głębokości sieci konwolucyjnej na trafność klasyfikacji obrazów. W modelach zostały wykorzystane standardowe warstwy konwolucyjne z filtrami małych rozmiarów (3x3). Przeanalizowano wydajność modeli składających się z od 11 do 19 warstw. Architektura modeli VGG opiera się na blokach 2 lub 3 warstw konwolucyjnych rozdzielonych warstwami zbierającymi. Model ConvNet zdobył drugie miejsce w zadaniu klasyfikacyjnym w konkursie ILSVRC 2014. W Tabeli 3. przedstawiono

[30] Autorzy nazwali swój model ConvNet (akronim od *convolutional network*); nazwa VGG pochodzi od grupy, której członkami byli autorzy modelu (ang. *Visual Geometry Group*), dzięki temu łatwo odróżnić tę architekturę od innych sieci konwolucyjnych.

dokładną budowę modeli VGG różnych rozmiarów. Oznaczenia warstw są następujące: np. conv3-64 jest określeniem warstwy konwolucyjnej z 3 kanałami i 64 filtrami. W niniejszej pracy zostanie wykorzystany model VGG16 (model D, 16-warstwowy).

Tabela 3. Architektura modeli VGG.

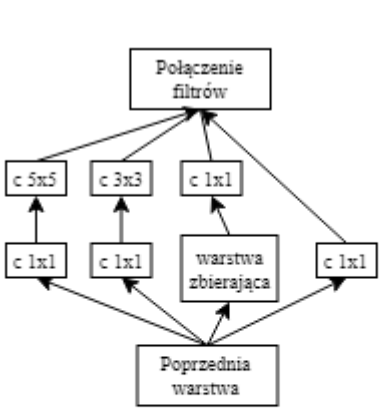
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	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**
maxpool					
conv3-512 conv3-512	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**
maxpool					
conv3-512 conv3-512	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**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Źródło: (Simonyan i Zisserman, 2015).

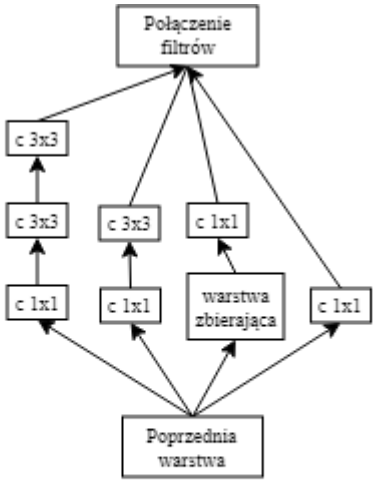
InceptionV3 (Szegedy, Vanhoucke, Ioffe, Shlens i Wojna, 2016)

Architektura modelu InceptionV3 wzorowana jest na stworzonym dwa lata wcześniej modelu GoogLeNet, nazywanym również Inception. W sieci konwolucyjnej Inception wykorzystano po raz pierwszy tzw. moduły incepcji (ang. *inception modules*), w których wykorzystywane są równolegle warstwy konwolucyjne z filtrami o rozmiarach 1x1 (te dodatkowo stosowane są w celu redukcji wymiarowości w warstwach z większymi filtrami), 3x3, 5x5 oraz warstwa zbierająca (Szegedy i inni, 2014).Ich nazwa pochodzi od faktu, że moduły incepcji tworzą jakby wiele małych modeli w jednym dużym modelu. Oryginalne moduły incepcji przedstawiono na

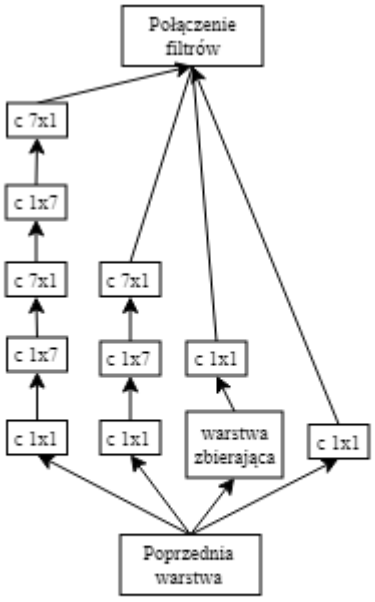
Rysunku 2a. Model InceptionV3 wprowadza trzy nowe moduły incepcji będące modyfikacją oryginalnych modułów z modelu Inception. W pierwszym z nich warstwy konwolucyjne z filtrami 5x5 są zastąpione dwoma warstwami konwolucyjnymi z filtrami 3x3 w celu zmniejszenia wymiarowości sieci (Rysunek 2b.). W drugim module warstwy konwolucyjne wykorzystujące większe filtry rozmiarów NxN są rozdzielane na kilka warstw wykorzystujących naprzemiennie filtry 1xN oraz Nx1 (Rysunek 2c.). Trzeci moduł charakteryzuje się podobną architekturą do drugiego, ale mniejsze warstwy są ułożone równolegle zamiast szeregowo (Rysunek 2d.).



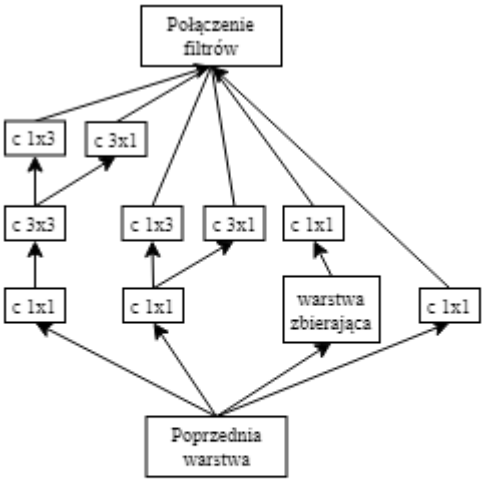
a.Oryginalny moduł incepcji.



b. Moduł incepcji z konwolucją 5x5 zastąpioną dwoma konwolucjami 3x3.



c. Moduł incepcji z konwolucją 7x7 zastąpioną szeregiem konwolucji 1x7 i 7x1.



d. Moduł incepcji z konwolucją 3x3 zastąpioną równoległymi konwolucjami 1x3 i 3x1.

Rysunek 2. Moduły incepcji wykorzystane w modelach Inception oraz InceptionV3.

Źródło: opracowanie własne na podstawie (Szegedy, Vanhoucke, Ioffe, Shlens i Wojna, 2016).

Podział złożonych warstw konwolucyjnych na kilka mniejszych skutkuje ułatwieniem obliczeń i przyspieszeniem trenowania. Budowę sieci InceptionV3 przedstawia Tabela 4 – „*As infigure 5*", „*As infigure 6*" oraz „*As infigure 7*" odnoszą się odpowiednio do modułów przedstawionych na Rysunkach 2b., 2c. oraz 2d.

Tabela 4. Architektura modelu InceptionV3.

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Źródło: (Szegedy, Vanhoucke, Ioffe, Shlens i Wojna, 2016).

MobileNet (Howard i inni, 2017)

Model MobileNet to sieć CNN, wykorzystująca warstwy konwolucyjne o dającej się odseparować głębokości (ang. *depthwise separable convolutions*) zamiast standardowych warstw konwolucyjnych. Różnica polega na rozdzieleniu funkcji zwykłej warstwy konwolucyjnej na dwa etapy: konwolucję wgłębną (ang. *depthwise convolution*), gdzie dla każdego kanału wejściowego[31] stosowany jest jeden osobny filtr, i konwolucję punktową (ang. *pointwise convolution*), której zadaniem jest połączenie wyników z konwolucji wgłębnej. Skutkuje to znacznym skróceniem obliczeń, co przekłada się na szybsze trenowanie modelu. Sieć MobileNet składa się z 28 warstw, architekturę modelu przedstawia Tabela 2. Po każdej warstwie konwolucyjnej zastosowano normalizację wsadu i skorygowaną jednostkę liniową jako funkcję aktywacji.

[31] patrz przypis 28.

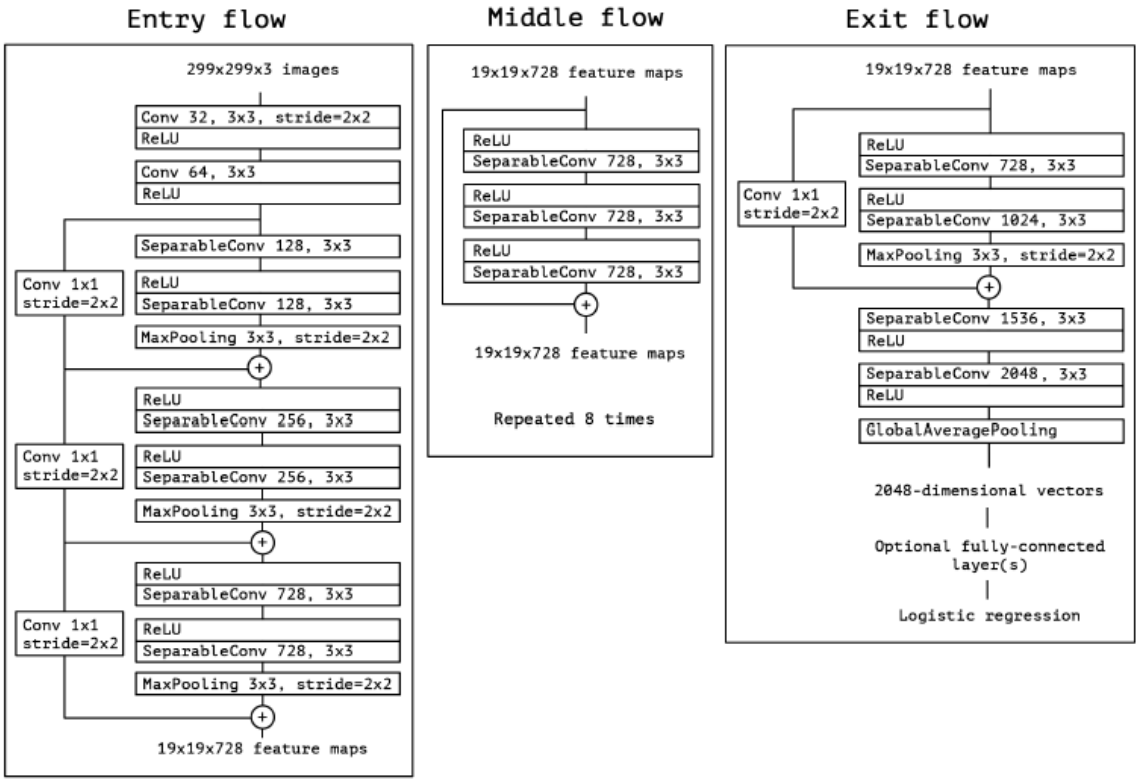
Tabela 5 . Architektura modelu MobileNet.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Źródło: (Howard i inni, 2017)

Xception (Chollet, 2017)

Model Xception łączy zalety skonstruowanych wcześniej modeli ResNet i MobileNet, opisanych powyżej. Składa się on z 36 warstw konwolucyjnych zorganizowanych w 14 modułów. Każdy z modułów jest otoczony połączeniem szczątkowym z warstwą konwolucyjną z jądrem 1x1. We wszystkich modułach, poza pierwszym, wykorzystano warstwy konwolucyjne o dającej się odseparować głębokości oraz ReLU jako funkcję aktywacji. Model Xception posiada tyle samo parametrów, co opisany powyżej model InceptionV3, jednak ze względu na bardziej efektywne ich wykorzystanie osiąga lepsze wyniki. Warstwy konwolucyjne o dającej się odseparować głębokości oraz moduły incepcji charakteryzują się podobnymi właściwościami, jednak te pierwsze są dużo łatwiejsze w użyciu. Budowa modelu Xception przedstawiona jest na Rysunku 3.



Rysunek 3. Architektura modelu Xception.

Źródło: (Chollet, 2017).

4. Implementacja algorytmów

Głównym celem niniejszej pracy jest stworzenie modelu uczenia maszynowego do klasyfikacji binarnej zdjęć rentgenowskich płuc. Narzędzia wybrane do rozwiązania tego zadania są ogólnodostępne, a ich użycie nie wymaga posiadania kosztownego sprzętu, specjalnie zoptymalizowanego pod kątem trenowania głębokich sieci neuronowych. Dzięki temu przedstawione rozwiązania są uniwersalne i mogą zostać wykorzystane przez każdego.

Niniejszy rozdział zawiera opis narzędzi wykorzystanych do stworzenia algorytmów, charakterystykę wybranego zbioru danych zdjęć rentgenowskich płuc oraz opis tworzenia modeli użytych do klasyfikacji. Rozdział kończy zestawienie wyników osiągniętych na małym zbiorze danych przez zbudowaną od podstaw kilkuwarstwową konwolucyjną sieć neuronową oraz modele opisane w Rozdziale 3.5, wykorzystane tutaj do uczenia transferowego, a także porównanie wyników osiągniętych we wszystkich opracowaniach korzystających z wybranego zbioru danych.

4.1. Wykorzystane narzędzia

Kryterium wyboru opisanych poniżej narzędzi była ich łatwość użycia i przyswojenia przez osobę początkującą oraz, w przypadku środowiska programistycznego, możliwość uniezależnienia się od własnego sprzętu i wykorzystania zasobów obliczeniowych dostępnych obecnie online.

Język programowania

Wszystkie algorytmy zostały napisane w języku Python. Jest on powszechnie używany do tworzenia modeli uczenia maszynowego ze względu na intuicyjną i prostą składnię. Dodatkowo wykorzystano bibliotekę do uczenia głębokiego o nazwie Keras[32], udostępnioną do użytku publicznego w 2015 roku, polecaną dla początkujących (Chollet, 2019). Jako podstawową bibliotekę do uczenia maszynowego, która jest bazą dla biblioteki Keras, wybrano TensorFlow, gdyż jest to najczęściej stosowane połączenie. Niestety przy wyborze TensorFlow bardzo trudno uzyskać dokładną powtarzalność wyników, pomimo zapewnienia jednowątkowej realizacji obliczeń i ustalenia ziaren wykorzystywanych generatorów losowych.

[32] F. Chollet i inni, 2015, https://keras.io

Środowisko programistyczne i obliczeniowe

Ze względu na dużą moc obliczeniową wymaganą do trenowania modeli głębokiego uczenia wykorzystano zasoby dostępne online poprzez platformę Kaggle zamiast sprzętu prywatnego. Użytkownicy platformy mogą skorzystać z procesora CPU (Intel® Xeon®2.30GHz) lub GPU (NVIDIA Tesla P100) oraz zasobów pamięci: 16 GB w przypadku użycia CPU oraz 13 GB w przypadku użycia GPU[33]. Kod może być wykonywany na platformie jako skrypt w języku Python lub fragment notatnika Jupyter – narzędzia, które pozwala na wzbogacenie kodu notatkami czy wizualizacjami. Kod wykorzystany w niniejszej pracy powstał w notatniku Jupyter na platformie Kaggle i jest publicznie dostępny pod następującym adresem:

https://www.kaggle.com/agnieszkapytel/chest-x-ray-classification-cnn-transfer-learning

4.2. Analiza zbioru danych

Zbiór danych użyty w niniejszej pracy został pobrany z platformy Kaggle[34]. Zawiera on 5856 obrazów rentgenowskich płuc w formacie JPEG oraz plik CSV zawierający etykiety obrazów, gdzie:

- „0” oznacza zapalenie płuc,
- „1” oznacza obraz zdrowych płuc.

Rysunek 4. przedstawia pierwsze 10 wierszy pliku CSV. Każdy obraz identyfikowany jest przez unikalny numer Id.

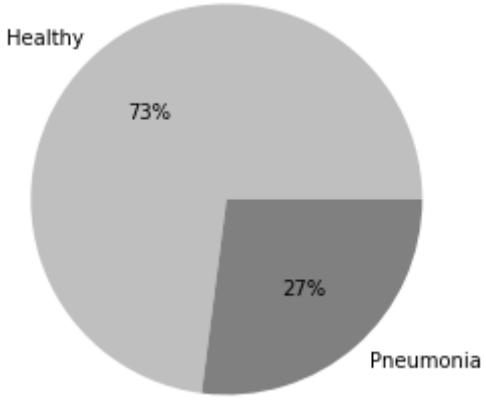
	Id	Ground_Truth
0	643781546	1
1	540270208	0
2	585452583	1
3	341665171	0
4	940983956	1

Rysunek 4. Początkowe wiersze pliku z etykietami obrazów ze zbioru danych.

Źródło: opracowanie własne – wynik działania programu.

[33]https://www.kaggle.com/docs/kernels
[34]https://www.kaggle.com/parthachakraborty/pneumonia-chest-x-ray

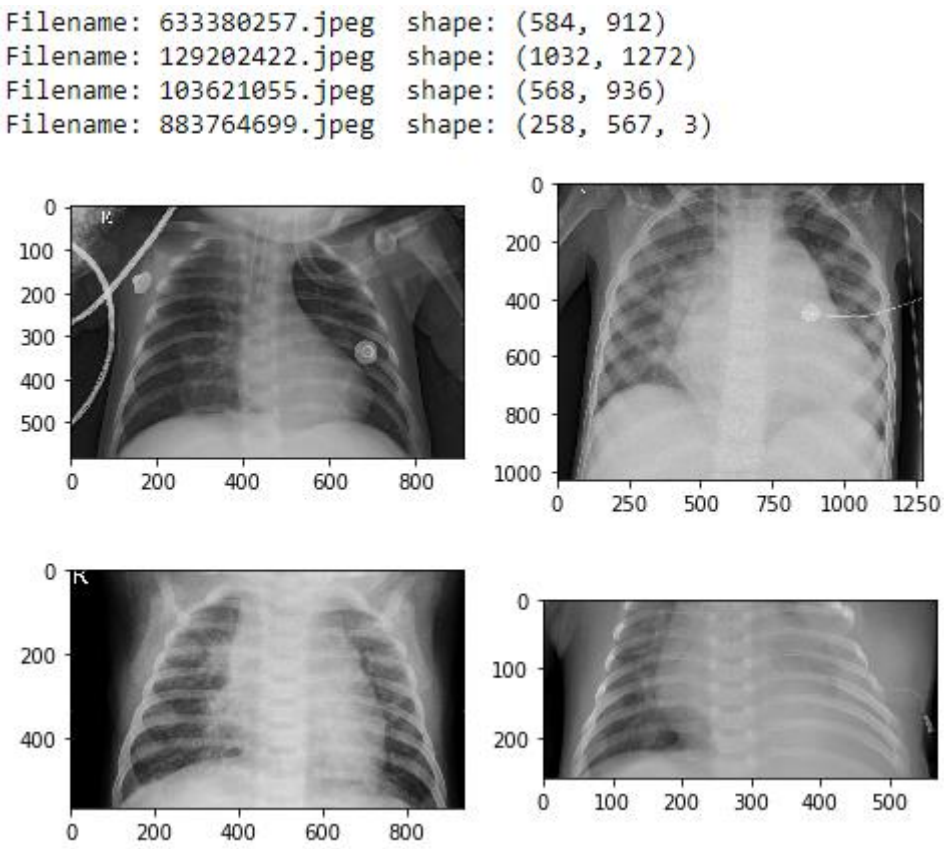
Na Rysunku 5. ukazano proporcje obrazów zdrowych i chorych płuc w zbiorze danych. Obrazów zdrowych płuc jest ponad dwukrotnie więcej niż obrazów zapalenia płuc: odpowiednio 73% (4273 obrazy) i 27% zbioru (1583 obrazów).



Rysunek 5. Procentowy udział obrazów zdrowych i chorych płuc w całym zbiorze danych. Źródło: opracowanie własne – wynik działania programu.

Na Rysunku 6. przedstawiono przykładowe zdjęcia obrazów rentgenowskich z analizowanego zbioru danych. Cechą obrazowych danych medycznych utrudniającą trenowanie modeli jest ich zróżnicowanie, co wyraźnie widać na przykładzie losowych zdjęć. Mają one zarówno różne rozmiary, jak i liczbę kanałów (2 kanały dla obrazów w skali szarości oraz 3 kanały dla obrazów RGB). Na Rysunku 6. liczby podane w nawiasach powyżej zdjęć to odpowiednio wysokość, szerokość i liczba kanałów, jeśli obraz jest zapisany jako RGB. W analizie obrazów rentgenowskich w praktyce wykorzystywana jest tylko skala szarości, jednak niektóre modele wykorzystane do uczenia transferowego w niniejszym opracowaniu są przystosowane jedynie do obrazów, w których liczba kanałów wynosi 3, dlatego w celu uproszczenia obliczeń wszystkie obrazy zostaną wczytane do pamięci jako obrazy RGB. Jak wspomniano w Rozdziale 3.4., konwolucyjne sieci neuronowe uczą się wykrywania lokalnych wzorców, dzięki czemu są niewrażliwe na przesunięcia. W przypadku różnorodnych danych medycznych jest to ogromna zaleta sieci CNN.

Aby uniknąć problemów związanych z niezrównoważonym zbiorem danych ograniczono liczbę obrazów treningowych do 1280 dla każdej klasy.



Rysunek 6. Przykładowe obrazy rentgenowskie ze zbioru danych wraz z podstawowymi danymi (nazwa pliku, wysokość, szerokość, liczba kanałów).

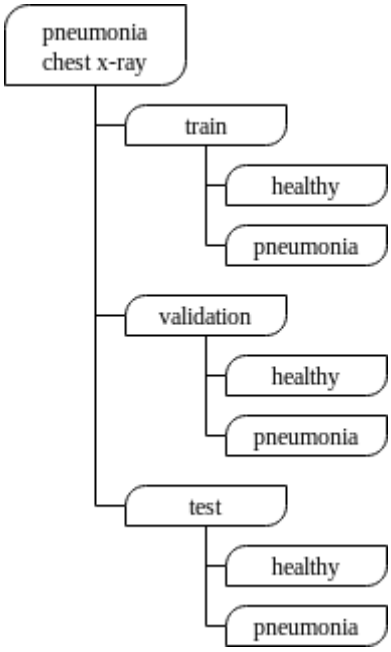
Źródło: opracowanie własne – wynik działania programu.

4.3. Wstępna obróbka danych

W przypadku zbiorów danych, których rozmiar to kilka GB, wczytanie danych bez wstępnego ich przetworzenia jest właściwie niewykonalne. Biblioteka Keras udostępnia klasę *ImageDataGenerator* która zawiera funkcje pozwalające na obróbkę danych przy ich wczytywaniu, np. zmianę rozmiaru, co znacząco odciąża procesor. Wspomnianą klasę można wykorzystać również do sztucznego powiększenia zbioru danych. Obrazy przy wczytywaniu mogą zostać poddane losowym przekształceniom (w zadanym zakresie), takim jak: skalowanie, obrót, zmiana jasności, przybliżenie, odbicie w pionie lub w poziomie. W niniejszej pracy nie zostaną jednak wykorzystane żadne techniki powiększania zbioru danych.

Na potrzeby wykorzystania funkcji *flow_from_directory* klasy *ImageDataGenerator*, która wczytuje obrazy z danego folderu, utworzono katalog główny o nazwie *„pneumonia chestx-ray”* oraz osobne katalogi dla zbioru

treningowego(ang. *train*), walidacyjnego (ang. *validation*) i testowego (ang. *test*), a w nich po jednym katalogu dla każdej z dwóch klas z etykietami *„pneumonia”* dla obrazów zapalenia płuc i *„healthy”* dla obrazów płuc zdrowych. Strukturę folderów przedstawia Rysunek 7.



Rysunek 7. Struktura folderów utworzonych na potrzeby funkcji flow_from_directory.

Źródło: opracowanie własne.

Pomniejszony zbiór danych, składający się z 2560 obrazów w dwóch klasach został podzielony na zbiór treningowy, walidacyjny i testowy w następujących proporcjach: odpowiednio 80%, 10% i 10%. Ostateczna liczba obrazów w folderach przedstawionych na Rysunku 7.:

- w folderze *„train”*: 2048 obrazów,
- w folderze *„validation”*: 256 obrazów,
- w folderze *„test”*: 256 obrazów.

Zaleca się, aby rozmiar wsadów (ang. *batch size*) zadawanych modelowi do walidacji i predykcji był podzielnikiem liczby obrazów, w celu zapewnienia, że każdy obraz będzie przetwarzany przez model dokładnie jeden raz. Jako wielkość wsadu wybrano liczbę 64 będącą potęgą liczby 2, ponieważ procesory są zoptymalizowane pod kątem operacji binarnych. Dzięki temu, że pomniejszony zbiór składa się z 1280

obrazów dla każdej klasy, liczba obrazów w zbiorze treningowym, walidacyjnym i testowym jest podzielna przez wielkość wsadu. Kroki na epokę (ang. *steps per epoch*), czyli liczbę iteracji algorytmu na kolejnych wsadach, obliczono jako iloraz liczby obrazów treningowych (lub testowych w przypadku ostatecznej oceny wytrenowanego modelu) i wielkości wsadu, ustalonej wcześniej ręcznie.Rozmiar wejściowy obrazów wynosi 128x128 pikseli. Wszystkie obrazy zostały wczytane jako RGB, a wartości pikseli zostały przeskalowane, tak by mieściły się w zakresie 0-1, w celu uniknięcia obliczeń na zbyt dużych liczbach.

4.4. Konwolucyjna sieć neuronowa

Na potrzeby zadania klasyfikacji dwuklasowej obrazów rentgenowskich płuc zbudowano od podstaw model CNN składający się z 3 warstw konwolucyjnych (każda z następującymi po sobie warstwami normalizacji wsadu i zbierającej), warstwy porzucania *Dropout* ze współczynnikiem porzucania na poziomie 0,3[35], warstwy gęstej i klasyfikatora. Dla każdej z warstw, poza ostatnią, jako funkcję aktywacji wykorzystano ReLU. Funkcja aktywacji ostatniej warstwy, klasyfikatora, to sigmoida. Budowę modelu, z pominięciem warstw normalizacji wsadu oraz warstwy porzucania, które nie wpływają na zmianę rozmiaru wejścia), przedstawia Tabela 6.

Tabela 6. Architektura zbudowanej od podstaw sieci CNN.

Typ warstwy	Rozmiar filtra	Rozmiar wejścia
Konwolucyjna	3 x 3, 16	128 x 128 x 16
Zbierająca	2 x 2	126 x 126 x 16
Konwolucyjna	3 x 3, 32	63 x 63 x 16
Zbierająca	2 x 2	61 x 61 x 32
Konwolucyjna	3 x 3, 64	30 x 30 x 32
Zbierająca	2 x 2	28 x 28 x 64
Gęsta	512 x 12544	1 x 1 x 12544
Gęsta - klasyfikator	Sigmoida	1 x 1 x 512

Źródło: opracowanie własne.

[35] oznacza to usunięcie 30% połączeń.

Liczbę wykorzystanych warstw konwolucyjnych wybrano tak, aby model był możliwie jak najprostszy i jednocześnie zapewniający wysoką trafność klasyfikacji. Trenowanie modelu z większą liczą warstw konwolucyjnych zajmowałoby znacznie więcej czasu i przekroczono by maksymalny czas wykonywania całego programu w wykorzystanym środowisku, tzn. 8 godzin. Przetestowano kilka różnych wartości współczynnika porzucania z przedziału od 0,1 do 0,6 i wybrano wartość 0,3, dla której osiągnięto najlepsze wyniki. Jako funkcję aktywacji wykorzystano ReLU, gdyż jest to najczęściej spotykane rozwiązanie, polecane również przez twórcę wykorzystanej biblioteki do uczenia głębokiego (Chollet, 2019).

Ostateczna liczba parametrów modelu podlegających trenowaniu wynosi 6 448 385. Model trenowano przez 50 epok pod kątem minimalizacji binarnej entropii krzyżowej. Jako metrykę wydajności modelu wybrano trafność, jako algorytm optymalizacji – SGD ze stałą uczenia o wartości 0,01.W celu ograniczenia nadmiernego dopasowania modelu do danych wykorzystano regularyzację L2, polegającą na zwiększeniu wartości funkcji straty o karę nakładaną na zbyt duże parametry (Chollet, 2019). Podczas trenowania wykorzystano wywołania zwrotne (ang. *callbacks*) udostępniane w bibliotece Keras, aby proces trenowania był bardziej efektywny. Wywołania zwrotne to funkcje, które umożliwiają wykonywanie pewnych operacji podczas treningu:

- *ReduceLROnPlateau* - pozwala na dynamiczną redukcję szybkości uczenia się, gdy nie ma poprawy wyniku funkcji straty w danej liczbie epok (w tym przypadku po 3 epokach stała uczenia była zmniejszana o połowę),
- *EarlyStopping* - pozwala przerwać trenowanie modelu, kiedy nie ma poprawy wyniku funkcji straty w danej liczbie epok (w tym przypadku 5 epok),
- *ModelCheckpoint* - pozwala zapisać najlepsze wagi modelu zanim zaczną się pogarszać w wyniku nadmiernego dopasowywania modelu do danych treningowych

(Chollet, 2019).

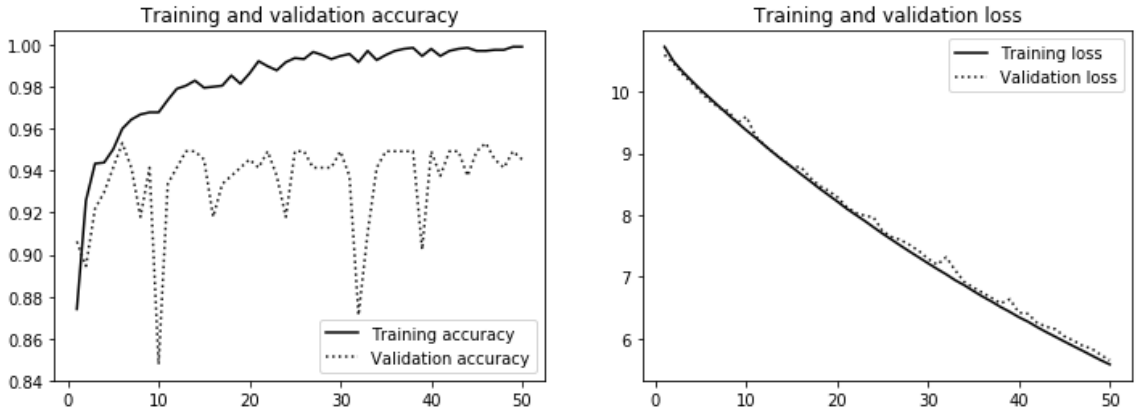
4.5. Uczenie transferowe

W niniejszej pracy wybrano strategię uczenia transferowego polegającą na wykorzystaniu modeli wytrenowanych na dużym zbiorze danych[36] do pozyskania cech (ang. *feature extraction*) ze zbioru danych medycznych, a następnie użycia tych informacji jako danych wejściowych klasyfikatora. Zastosowano pięć modeli opisanych w Rozdziale 3.5., udostępnianych w bibliotece Keras. Klasyfikację przeprowadzono przy użyciu prostej sieci neuronowej składającej się z warstwy gęstej z funkcją aktywacji ReLU, warstwy porzucania ze współczynnikiem 0,5 oraz warstwy gęstej pełniącej funkcję finalnego klasyfikatora, z sigmoidą jako funkcją aktywacji. Jako optymalizator został wybrany Adam z bardzo małą stałą uczenia – 0,00001. Jako funkcji straty i metryki wydajności modelu użyto odpowiednio binarnej entropii krzyżowej oraz trafności, podobnie jak w modelu CNN opisanym w Rozdziale 4.4. Dodatkowo wykorzystano dwa wywołania zwrotne: *ModelCheckpoint* oraz *EarlyStopping* z wstrzymaniem trenowania modelu po 10 epokach bez poprawy wartości funkcji straty na zbiorze walidacyjnym. Wszystkie pięć modeli było trenowanych przez 50 epok.

4.6. Wyniki

Wytrenowano łącznie sześć modeli w czasie 6 godzin. Pierwszy z nich to skonstruowana od podstaw sieć konwolucyjna z warstwami zbierającymi, normalizacją wsadu, warstwą porzucania i regularyzacją L2. Rysunek 8. przedstawia wykresy funkcji trafności (ang. *accuracy*) i straty (ang. *loss*) dla zbioru treningowego i walidacyjnego sieci CNN. Trafność na zbiorze treningowym pod koniec trenowania wyniosła niemal 100%, co oznacza, że model dokładnie dopasował się do danych. Trafność walidacji mocno się wahała pomiędzy epokami, co najprawdopodobniej było spowodowane niewielkim rozmiarem zbioru treningowego. Pomimo trenowania trwającego aż 50 epok, trafność walidacji nie wzrosła powyżej wartości 96%. Daje to podstawę do wnioskowania, że dłuższe trenowanie nie spowodowałoby lepszej klasyfikacji na danych walidacyjnych. Poprawę wyników mogłaby przynieść modyfikacja jego architektury, np. poprzez dodanie nowych warstw konwolucyjnych. Funkcja straty zarówno na danych treningowych, jak i walidacyjnych malała niemal liniowo. Dzięki wykorzystaniu regularyzacji L2 wahania wartości funkcji straty są niewielkie.

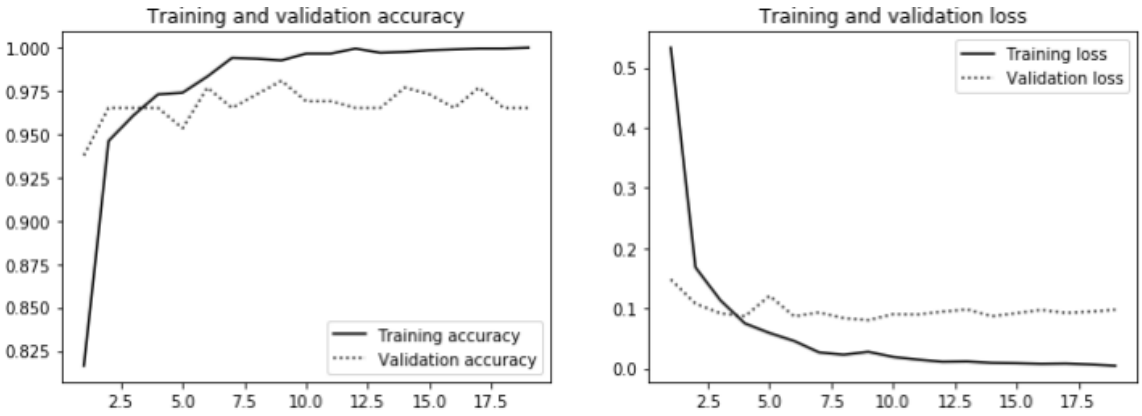
[36] ImageNet, http://www.image-net.org



Rysunek 8. Wykresy trafności i straty treningowej i walidacyjnej modelu CNN.

Źródło: opracowanie własne – wynik działania programu.

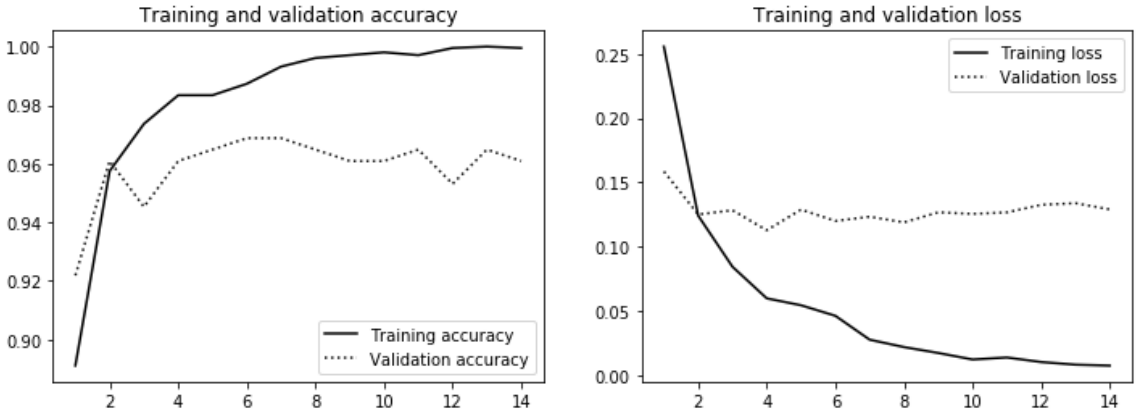
Kolejne pięć wytrenowanych modeli to bardzo proste klasyfikatory o jednakowej strukturze, składające się z warstwy gęstej, warstwy porzucania i warstwy gęstej w funkcji klasyfikatora. Wszystkie modele zostały wykorzystane w uczeniu transferowym – ich wejściami były cechy pozyskane ze zbioru treningowego przez modele wytrenowane na zbiorze ImageNet. Kolejność trenowania modeli odpowiadała ich rozmiarom w bibliotece Keras: pierwszym modelem służącym do ekstrakcji cech był, najmniejszy ze wszystkich pięciu, model MobileNet. Wykresy trafności i straty klasyfikatora trenowanego na cechach z tego modelu ukazano na Rysunku 9. Model od pierwszej epoki osiągał wysoką trafność, ponad 95%. Trenowanie zakończyło się po 19 epokach, zatem najniższa wartość straty walidacji została osiągnięta w 9 epoce.



Rysunek 9. Wykresy trafności i straty treningowej i walidacyjnej klasyfikatora cech z modelu MobileNet.

Źródło: opracowanie własne – wynik działania programu.

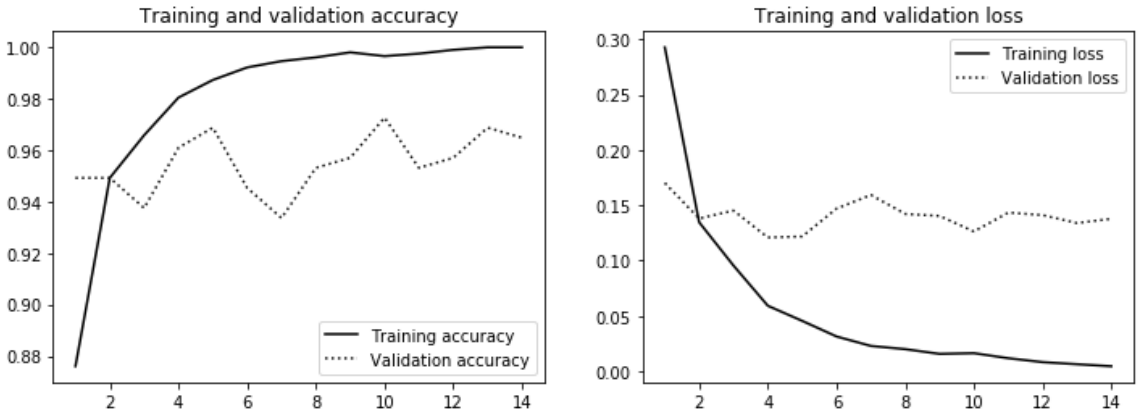
Drugim modelem zastosowanym do uczenia transferowego był model Xception, wykorzystujący warstwy konwolucyjne o dającej się odseparować głębokości. Wyniki trenowania klasyfikatora z danymi wejściowymi pozyskanymi z modelu Xception przedstawia Rysunek 10. Trenowanie klasyfikatora trwało 14 epok, co oznacza, że najmniejsza wartość straty walidacji została osiągnięta bardzo wcześnie – w 4 epoce. Model już na początku trenowania, w pierwszej epoce, osiągnął wysoką trafność na poziomie 92%.



Rysunek 10. Wykresy trafności i straty treningowej i walidacyjnej klasyfikatora cech z modelu Xception.

Źródło: opracowanie własne – wynik działania programu.

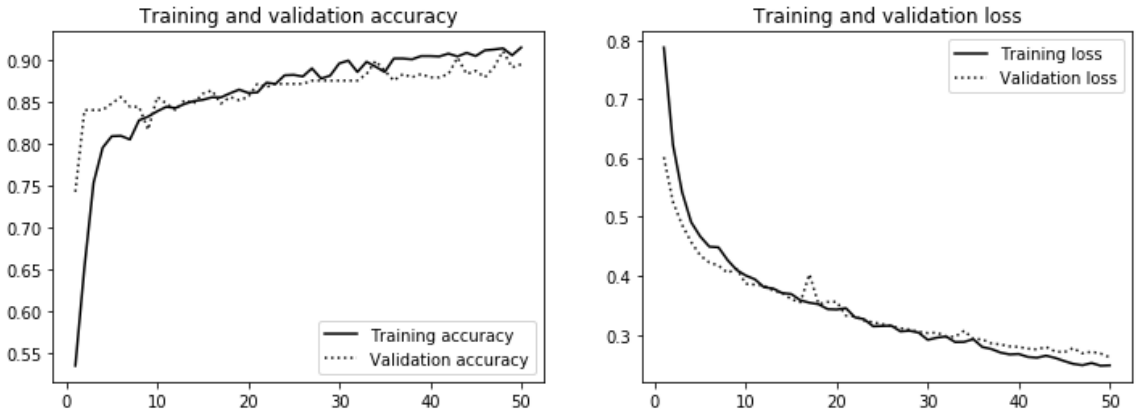
Kolejny model wykorzystany jako ekstraktor cech to InceptionV3. Najważniejszymi elementami jego architektury są bloki incepcyjne, przypominające modele w środku modelu. Przebieg trenowania klasyfikatora cech z modelu InceptionV3 obrazuje Rysunek 11. Model klasyfikujący był trenowany przez 14 epok, podobnie jak poprzedni, funkcja straty na zbiorze walidacyjnym osiągnęła minimum w 4 epoce. W pierwszej epoce wartość trafności wyniosła niemal 95%.



Rysunek 11. Wykresy trafności i straty treningowej i walidacyjnej klasyfikatora cech z modelu InceptionV3.

Źródło: opracowanie własne – wynik działania programu.

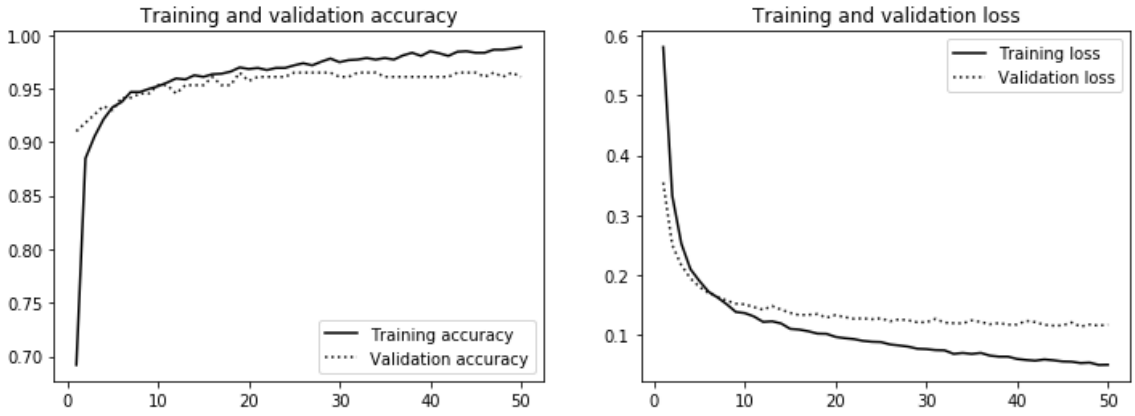
Przedostatni model użyty do pozyskania cech z danych treningowych to ResNet50, którego najważniejszym elementem budowy są połączenia szczątkowe. Wykres funkcji trafności i straty klasyfikatora dla danych z modelu ResNet50 przedstawiono na Rysunku 12. Trenowanie modelu trwało pełne 50 epok. Początkowa trafność walidacji wyniosła jedynie około 74%, mniej niż w przypadku wszystkich innych trenowanych modeli, jednak stabilnie wzrastała w miarę postępu treningu. Wydłużenie czasu trenowania tego modelu mogłoby prowadzić do osiągnięcia wyników porównywalnych z wynikami pozostałych modeli lub nawet lepszych.



Rysunek 12. Wykresy trafności i straty treningowej i walidacyjnej klasyfikatora cech z modelu ResNet50.

Źródło: opracowanie własne – wynik działania programu.

Ostatnim z sześciu trenowanych w niniejszej pracy modeli był klasyfikator dla cech uzyskanych z modelu VGG16, największego z modeli wybranych do uczenia transferowego. Przebieg jego trenowania pokazano na Rysunku 13.



Rysunek 13. Wykresy trafności i straty treningowej i walidacyjnej klasyfikatora cech z modelu VGG16.

Źródło: opracowanie własne – wynik działania programu.

Model był trenowany przez 50 epok. W pierwszej epoce uzyskano trafność klasyfikacji na zbiorze walidacyjnym na poziomie 90%. W kolejnych epokach strata walidacji konsekwentnie malała, jednak nie w takim stopniu, jak w klasyfikatorze dla modelu ResNet50, zapewne ze względu na to, że w przypadku modelu VGG16 klasyfikator już od początku osiągał dobre wyniki.

W Tabeli 7. przedstawiono wyniki klasyfikacji osiągniętej na zbiorze testowym oraz charakterystyczne cechy modeli stworzonych dla wykorzystanego w niniejszej pracy zbioru danych zdjęć rentgenowskich, opisanych w Rozdziale 2.2. Zbudowana od podstaw konwolucyjna sieć neuronowa (CNN) osiągnęła najlepsze wyniki spośród modeli stworzonych dotąd od podstaw dla wybranego zbioru. Dane przedstawione w Tabeli 7. są czysto informacyjne. Mimo że modele mają podobną architekturę, to różnią się między sobą w wielu szczegółach, jak np. etapami, na których została zastosowana normalizacja wsadu, co sprawia, że wyciąganie wniosków o wpływie charakterystyki tych modeli na osiągnięte wyniki jest bezpodstawne.

Tabela 7. Porównanie modeli stworzonych dotychczas od podstaw dla wybranego zbioru danych.

Model	Trafność	Zbiór danych	Liczba warstw konwolucyjnych	Normalizacja wsadu
CNN	95,70%	Pomniejszony	3	Tak
CNN #1[37]	94,71%	Bez zmian	6 (3 podwójne)	Nie
CNN #2[38]	92,07%	Powiększony	3	Nie
CNN #3[39]	95,32%	Bez zmian	3	Tak

Źródło: opracowanie własne.

Ze względu na różnice w architekturze i rozmiarze modeli z biblioteki Keras wykorzystanych w uczeniu transferowym, znacząco różniły się czasy ekstrakcji cech oraz trenowania klasyfikatorów na uzyskanych danych. Duże znaczenie ma również fakt, że klasyfikator dla danych z modelu MobileNet był trenowany jedynie przez 19 epok, z modeli Xception oraz InceptionV3 – przez 14 epok, a z modeli ResNet oraz VGG – przez pełne 50 epok. Najkrótszy czas ekstrakcji cech oraz czas obliczeń odnotowano dla MobileNet, najmniejszego z modeli, a najdłuższy – dla modelu Xception. Wytrenowanie klasyfikatora zajęło najmniej czasu w przypadku modelu MobileNet, jednak najwięcej dla modelu ResNet50. Jedynie dla modelu ResNet czas trenowania klasyfikatora przekroczył czas potrzebny na pozyskanie cech z danych.

Tabela 8. Porównanie szybkości obliczeń w uczeniu transferowym w zależności od rozmiaru modeli - ekstraktorów cech.

Model	Rozmiar	Czas ekstrakcji cech	Czas trenowania klasyfikatora	Czas całkowity[40]
MobileNet	16 MB	455 s	435 s	889 s
Xception	88 MB	2349 s	1286 s	3636 s
InceptionV3	92 MB	1032 s	870 s	1902 s
ResNet50	98 MB	1037 s	2281 s	3318 s
VGG16	528 MB	1508 s	599 s	2107 s

Źródło: opracowanie własne.

[37] patrz: przypis 21, s. 15.
[38] patrz: przypis 24, s. 15.
[39] patrz: przypis 25, s. 15.
[40] w niektórych przypadkach suma czasu ekstrakcji cech i trenowania nie dają arytmetycznie dokładnego czasu całkowitego ze względu na zaokrąglenia wykorzystane w wyświetlaniu liczb.

Tabela 9. zawiera porównanie wyników wszystkich modeli wytrenowanych w niniejszej pracy. Najlepszą trafność (95,70%) osiągnięto dla trzech modeli: zbudowanej od podstaw CNN oraz klasyfikatora dla cech z modelu MobileNet i z modelu Xception. Nieco gorsze wyniki osiągnęły klasyfikatory dla cech z modeli VGG16 oraz InceptionV3: odpowiednio 95,31% oraz 94,92%. Najniższą wartość trafności (89,45%) osiągnięto dla klasyfikatora, którego danymi wejściowymi były cechy pozyskane przy pomocy modelu ResNet, jednak ten wynik mógłby być wyższy w przypadku wydłużonego trenowania.

Najbardziej czasochłonny w trenowaniu był zbudowany od podstaw model CNN. Dwa razy krócej zajęło wytrenowanie klasyfikatorów dla danych z modeli InceptionV3 oraz VGG16. Najkrótszy czas trenowania odnotowano dla klasyfikatora dla danych z modelu MobileNet.

Precyzja w rozwiązywanym zadaniu klasyfikacyjnym oznacza odsetek zdjęć rentgenowskich zaklasyfikowanych jako „zapalenie płuc", które w rzeczywistości również przedstawiają zapalenie płuc. Najwyższą wartość precyzji osiągnięto dla modelu CNN oraz klasyfikatora dla danych z modelu Xception.

Czułość algorytmu oznacza w tym przypadku zdolność modelu do rozpoznawania zapalenia płuc na obrazach rentgenowskich. Najlepszy wynik w tym kontekście otrzymano dla modelu MobileNet (95%), nieco niższy dla modeli Xception, InceptionV3 oraz VGG16 (94%).

Tabela 9. Porównanie wyników modelu CNN i modeli wykorzystanych w uczeniu transferowym.

Model	**Trafność**	**Precyzja**	**Czułość**	**Miara F**	**Czas trenowania**[41]
CNN	95,70%	0,98	0,93	0,96	4426 s
MobileNet	95,70%	0,96	0,95	0,96	902 s
Xception	95,70%	0,98	0,94	0,96	3719 s
InceptionV3	94,92%	0,96	0,94	0,95	1925 s
ResNet50	89,45%	0,92	0,87	0,89	2470 s
VGG16	95,31%	0,97	0,94	0,95	1900 s

Źródło: opracowanie własne.

[41] dla modelu CNN: czas całkowity = czas trenowania; dla modeli wykorzystanych w uczeniu transferowym: czas całkowity = czas ekstrakcji cech + czas trenowania klasyfikatora.

W przypadku pracy ze zbilansowanym zbiorem danych, jak w niniejszym opracowaniu, wartości miary F odpowiadają trafności. W medycynie najcenniejsze jest prawidłowe zdiagnozowanie choroby, zwykle oznaczanej jako klasa pozytywna. Nieprawidłowa klasyfikacja obrazu zdrowych płuc jako obrazu zapalenia płuc jest obarczona mniejszym kosztem niż nieprawidłowa klasyfikacja zapalenia płuc jako obrazu zdrowych płuc, co może doprowadzić do śmierci pacjenta. W związku z tym najlepszą metryką oceny modeli jest w tym kontekście czułość. Na tej podstawie najlepszym rozwiązaniem problemu klasyfikacji binarnej obrazów rentgenowskich w niniejszej pracy jest uczenie transferowe z wykorzystaniem modelu MobileNet.

Na podstawie wyników osiągniętych w tej pracy można stwierdzić, że uczenie transferowe jest dużo lepszym rozwiązaniem niż tworzenie modelu od podstaw, gdyż pozwala w prosty sposób otrzymać zadowalające efekty, bez potrzeby długiego dopasowywania architektury i parametrów modelu.

5. Zakończenie

Do rozwiązania problemu klasyfikacji binarnej zbioru zdjęć rentgenowskich płuc wytrenowano sześć różnych modeli, w tym jeden od podstaw oraz pięć z wykorzystaniem uczenia transferowego. W każdym modelu wykorzystano konwolucyjne sieci neuronowe. Osiągnięte wyniki ukazują, że samodzielne tworzenie modelu może być dobrym rozwiązaniem, jeżeli posiada się wiedzę potrzebną do stworzenia wydajnej architektury, jednak cały proces konstruowania algorytmu do klasyfikacji binarnej obrazów można znacząco uprościć z wykorzystaniem uczenia transferowego. Dzięki publicznemu udostępnieniu modeli wygrywających ogólnoświatowe konkursy klasyfikacji wizualnej każdy może wykorzystać współczesne osiągnięcia w dziedzinie uczenia maszynowego na prywatne potrzeby. Zaskakującym odkryciem jest fakt, że modele w uczeniu transferowym, pomimo wytrenowania na niezwiązanym z medycyną zbiorze ImageNet, osiągają tak dobre wyniki na danych medycznych.

Dziedzina uczenia maszynowego stale się rozwija, co prowadzi do odkrywania nowych rozwiązań pozwalających otrzymać lepsze wyniki. Niestety jest to dziedzina w dużej mierze eksperymentalna, co oznacza, że w celu znalezienia najlepszego algorytmu dla danego zadania należy przetestować wiele różnych modeli oraz metod. Trafność klasyfikacji osiągnięta przez modele w tej pracy jest bardzo wysoka, jednak istnieją rozwiązania, które potencjalnie pozwoliłyby na dalszą poprawę wyników. Są to m. in:

- zwiększenie rozmiaru wczytywanych podczas trenowania, co pozwoliłoby wykryć bardziej szczegółowe cechy, niż przy mniejszej rozdzielczości,
- wykorzystanie procesorów graficznych do przetwarzania obrazów, co pozwoliłoby znacząco skrócić czas trenowania modeli i umożliwiłoby trenowanie przez większą liczbę epok,
- walidacja krzyżowa, dzięki której model jest trenowany kilka razy z zastosowaniem innego podziału zbioru danych na treningowy i walidacyjny, co pozwala w pełni wykorzystać posiadane dane oraz zapobiegać nadmiernemu dopasowaniu modelu do danych,

- modyfikacja architektury modelu i parametrów, takich jak stała uczenia czy współczynnik porzucenia, a następnie porównanie wyników osiąganych przed i po zmianie,
- wykorzystanie innych modeli do uczenia transferowego,
- wykorzystanie pełnego zbioru danych oraz dodatkowo różnych technik sztucznego powiększania zbioru danych treningowych, np. poprzez przekształcenia dostępnych danych czy wygenerowanie nowych danych za pomocą generatywnych sieci przeciwstawnych,
- zbudowanie zespołu modeli zamiast pojedynczego modelu,
- połączenie metod klasyfikacji o różnym działaniu, np. konwolucyjnych sieci neuronowych oraz drzew decyzyjnych,
- ponowne przetrenowanie modeli na połączonym zbiorze treningowym i walidacyjnym, w celu pełnego wykorzystania posiadanych informacji.

Niestety uruchamianie kodu w środowisku udostępnionym na platformie Kaggle ma swoje ograniczenia: wykonanie całości kodu nie może zająć więcej niż 8 godzin. Powyższe rozwiązania nie zostały wykorzystane w niniejszym opracowaniu ze względu na wspomniane restrykcje używanego środowiska obliczeniowego.

Uczenie maszynowe, a w szczególności uczenie głębokie, ma ogromny potencjał w zastosowaniach w dziedzinie medycyny. Praca nad ciągłym udoskonalaniem modeli motywowana jest faktem, że każdy sposób poprawy wyników to szansa na bardziej trafną diagnozę i objęcie opieką medyczną tych, którzy nie mają dostępu do lekarzy specjalistów. Jest to doskonały przykład wykorzystania nowych technologii w służbie człowiekowi.

Bibliografia

Carbonell, J. G., Michalski, R. S. i Mitchell, T. M. (1983). An Overview of Machine Learning. W:Machine Learning: An Artificial Intelligence Approach (3-20). Palo Alto, California: TIOGA Publishing Co.

Changhau, I. (2017). Loss Functions in Neural Networks.

https://isaacchanghau.github.io/post/loss_functions/, dostęp: [24.07.2019].

Chollet, F. (2017). Xception: Deep Dearning with Depthwise Separable Convolutions. arXiv:1610.02357.

Chollet, F. (2019). Deep Learning. Praca z językiem Python i biblioteką Keras. Gliwice: Helion.

Cireşan, D., Giusti, A., Gambardella, L. i Schmidhuber, J. (2013). Mitosis detection in breast cancer histology images with deep neural networks. Med Image Comput Comput Assist Interv., 411-418.

Duchi, J., Hazan, E. i Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research 12, 2121-2159.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative Adversarial Nets. Montréal: Université de Montréal.

Harrington, P. (2012). Machine learning basics. W:Machine Learning in Action (3-18). Shelter Island: Manning Publications Co.

He, K., Zhang, X., Ren, S. i Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv: 1512.03385.

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861.

Ioffe, S. i Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167.

Kingma, D. i Ba, J. (2015). Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kingma, D. i Welling, M. (2013). Auto-encoding variational bayes. arXiv:1312.6114.

Krawiec, K. i Stefanowski, J. (2003). Wprowadzenie do sieci neuronowych. W Uczenie maszynowe i sieci neuronowe (82-100). Poznań: Wydawnictwo Politechniki Poznańskiej.

Kumar, A., Kim, J., Lyndon, D., Fulham, M. i Feng, D. (2016). An Ensemble of Fine-Tuned Convolutional Neural Networks for Medical Image Classification. IEEE Journal of Biomedical and Health Informatics 21, 31-40.

Kwaśniewska, A., Giczewska, A. i Rumiński, J. (2017). Duże zbiory danych w zdalnej diagnostyce medycznej z wykorzystaniem technik głębokiego uczenia. Gdańsk: Politechnika Gdańska.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A., Ciompi, F., Ghafoorian, M., Laak, J. A. W. M., Ginneken, B., Sánchez, C. I. (2017). A Survey on Deep Learning in Medical Image Analysis. Medical Image Analysis(42), 60-88.

Lu, D. i Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. International Journal of Remote Sensing, 28(5), 823-870.

Pan, S. i Yang, Q. (2009). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering 22, 1345-1359.

Patterson, J. i Gibson, A. (2018). Podstawy sieci neuronowych i głębokiego uczenia. W:Deep learning: praktyczne wprowadzenie (53-87). Gliwice: Helion.

Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul A., Ball, R. L., Langlotz, C., Shpanskaya, K., Lungren, M. P., Ng, A. (2017). CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. Stanford: Stanford ML Group.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks 61, 85-117.

Shen, D., Wu, G. i Suk, H.-I. (2017). Deep Learning in Medical Image Analysis. Annual Review of Biomedical Engineering 19, 221-248.

Shin, H.-C., Roth, H., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., Summers, R. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. IEEE Trans Med Imaging 5, 1285-98.

Simonyan, K. i Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations.

Sokolova, M. i Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing and Management 45, 427–437.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2014). Going Deeper With Convolutions. arXiv:1409.4842.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. i Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Tieleman, T. i Hinton, G. (2012). Lecture 6.5 - RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Wu, R., Yan, S., Shan, Y., Dang, Q. i Sun, G. (2015). Deep Image: Scaling up Image Recognition. Baidu Research.

Zagoruyko, S. i Komodakis, N. (2017). Paying More Attention To Attention: Improving The Performance Of Convolutional Neural Networks Via Attention Transfer. Paris: Université Paris-Est.

Spis rysunków

Spis tabel