

**Uniwersytet Warszawski
Wydział Fizyki**

Katarzyna Agnieszka Filipuk

Nr albumu: 324483

Urszula Agata Romaniuk

Nr albumu: 347442

Izabela Maria Szopa

Nr albumu: 363600

**Klasyfikacja obrazów tarczy nerwu
wzrokowego z użyciem sieci
neuronowej typu deep learning**

Praca magisterska

na kierunku Zastosowania Fizyki w Biologii i Medycynie
specjalność Neuroinformatyka

Praca wykonana pod kierunkiem
dra hab. Jacka Pniewskiego
Zakład Optyki Informacyjnej
Instytut Geofizyki
Wydział Fizyki UW

Warszawa, czerwiec 2019

Oświadczenie kierującego pracą

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data

Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data

Podpis autora (autorów) pracy

Streszczenie

W niniejszej pracy omówiono problem automatycznego rozpoznawania zmian chorobowych jaskry na obrazach tarczy nerwu wzrokowego wykonanych za pomocą funduskamery, bez użycia mydriatyków. Przedstawiono definicję jaskry i najczęściej używane sposoby automatycznej klasyfikacji obrazów. Wprowadzono pojęcie sieci neuronowych oraz zaprezentowano najbardziej znane modele sieci konwolucyjnych. Omówiono zagadnienie *transfer learningu* oraz wytwarzania sztucznych obrazów uczących. Przeprowadzono eksperymenty polegające na poszukiwaniu najskuteczniejszego klasyfikatora obrazów. Klasyfikatory te bazowały na konwolucyjnych sieciach neuronowych, a do ich stworzenia użyto *transfer learningu* oraz *data augmentation*. Do tworzenia modeli używano dwóch bibliotek: caffe oraz fastai. Biblioteki te zostały ze sobą porównane pod kątem użytkowania oraz wyników klasyfikatora na nich osiągniętych. Przekonano się, że biblioteka fastai jest nie tylko prostsza w obsłudze, ale także posiada implementację innowacyjnych algorytmów. Dzięki nim wyniki otrzymywane z modeli stworzonych przy użyciu fastai są lepsze od wyników uzyskanych przy użyciu caffe. Ostateczne wartości miar oceny najlepszego klasyfikatora wynosiły $78,1 \pm 6,4\%$ dla czułości, $78,7 \pm 1,9\%$ dla dokładności, $75,2 \pm 5,1\%$ dla precyzji, $79,2 \pm 6,0\%$ dla specyficzności.

Słowa kluczowe

jaskra, deep learning, sieci neuronowe, transfer learning, fine-tuning, data augmentation, wytwarzanie sztucznych obrazów uczących, konwolucyjne sieci neuronowe, caffe, fastai

Dziedzina pracy (kody wg programu Socrates-Erasmus)

13.2 Fizyka

Tytuł pracy w języku angielskim

Classification of optic nerve head images with deep neural networks

Spis treści

Cel pracy	3
1. Układ pracy	5
2. Motywacja	6
3. Definicja jaskry	8
3.1. Wykrywanie jaskry	8
4. Automatyczna klasyfikacja obrazów	10
4.1. Metody uczenia maszynowego	10
4.1.1. Drzewa decyzyjne	10
4.1.2. Maszyny wektorów wspierających	11
4.1.3. Naiwny klasyfikator Bayesa	11
5. Sztuczne sieci neuronowe	12
5.1. Typy sieci neuronowych	13
5.2. Uczenie sieci neuronowych	14
5.2.1. Propagacja wsteczna	14
6. Sieci neuronowe typu deep learning	15
6.1. Konwolucyjne sieci neuronowe	16
6.1.1. Architektura	17
6.1.2. Zastosowania	18
7. Transfer learning	20
8. Wytwarzanie sztucznych obrazów uczących	22
9. Przegląd publikacji medycznych	23
9.1. Publikacje medyczne dotyczące automatycznego wykrywania jaskry	24
10. Metody analizy statystycznej	25
10.1. Ocena klasyfikatora	25
10.2. Walidacja krzyżowa	26
10.3. Testy statystyczne	26
10.3.1. Test Shapiro-Wilka	27
10.3.2. Test Fishera-Snedecora	27
10.3.3. Test t-Studenta	27

11. Część eksperymentalna	29
11.1. Dane eksperymentalne	29
11.1.1. Obrazy medyczne	29
11.1.2. Środowisko programowania	29
11.2. Eksperiment przy użyciu caffe	30
11.2.1. Konwolucyjna sieć neuronowa	30
11.2.2. Preprocessing	30
11.2.3. Transfer learning	30
11.2.4. Sztuczne powiększanie zbioru uczącego	34
11.2.5. Podsumowanie	38
11.3. Eksperiment przy użyciu fastai	40
11.3.1. Tworzenie klasyfikatora obrazów przy użyciu fastai	40
11.3.2. Specyfikacja techniczna	44
11.3.3. Charakterystyka danych wejściowych	44
11.3.4. Wybór modelu konwolucyjnej sieci neuronowej	44
11.3.5. Sztuczne powiększanie zbioru uczącego	47
11.3.6. Podsumowanie	53
11.4. Porównanie eksperymentów	54
12. Dyskusja	56
13. Wnioski	57

Cel pracy

Stworzenie klasyfikatora obrazów tarczy nerwu wzrokowego (zrobionych bez użycia mydriatyków) wykonanych za pomocą funduskamery, w celu automatyzacji rozpoznawania zmian chorobowych.

Rozdział 1

Układ pracy

W poniższej pracy wyróżnić można cztery części: wstęp, część teoretyczną, część eksperymentalną i podsumowanie.

Wstęp skupia się na przedstawieniu powodów, dla których autorzy zdecydowali się na próbę stworzenia testu przesiewowego w kierunku jaskry.

Część teoretyczna zawiera wprowadzenie zagadnień używanych w części eksperymentalnej. Ma to na celu zapoznanie czytelnika z zastosowanymi definicjami.

Rozdział poświęcony eksperymetowi podzielony jest na dwie części. Pierwsza z nich prezentuje tradycyjne, często stosowane podejście, jakim jest użycie biblioteki caffe do tworzenia klasyfikatorów. W drugiej części przedstawiono najnowsze podejście do implementacji sieci neuronowych. Jest nim zastosowanie biblioteki fastai.

W podsumowaniu zaprezentowano dyskusję otrzymanych wyników oraz wyciągnięte wnioski.

Rozdział 2

Motywacja

K. A. FILIPIUK

Postęp cywilizacyjny nierozerwalnie wiąże się odkrywaniem przez człowieka kolejnych praw rządzących otaczającym go światem. Zrozumienie jego reguł pozwala na świadomą ingerencję nie tylko w środowisko, ale także w samego człowieka.

Rozwój medycyny prowadzi przede wszystkim do wydłużenia życia. W latach 1990 - 2017, średnia oczekiwana długość życia wzrosła o 6,8 lat na świecie oraz o 7 lat w Polsce. W roku 2017 średnia ta wynosiła 72,2 lat dla świata oraz 77,9 lat dla Polski [1].

Niestety, wraz z wydłużaniem się ludzkiego życia, wydłuża się także liczba lat utraconych w wyniku niepełnosprawności. W celu kwantytatywnego opisu tejże liczby lat stworzony został wskaźnik YLD (ang. *Years Lived with Disability*). Wyraża on sumaryczną liczbę lat przeżytych od utraty sprawności do jej odzyskania bądź śmierci dla 100 000 osób.

Według danych zebranych przez *Institute for Health Metrics and Evaluation*, od 1990 do 2017 roku wzrost wskaźnika YLD wyniósł 2 000 lat w skali globalnej oraz 750 lat dla Polski [2].

Jednym z czynników powodujących utratę sprawności jest jaskra. Jej udział we wszystkich schorzeniach zagregowanych w YLD wydaje się nieznaczący i wynosi odpowiednio 0,08% oraz 0,04% dla świata i dla Polski. Jednakże analizując istotność jaskry dla omawianego wskaźnika w ciągu przytaczanych 27 lat, wyraźnie kształtuje się tendencja wzrostowa. Przekłada się ona na zwiększenie YLD o 2 lata globalnie oraz o 1,4 lat dla Polski [2].

Raport WHO zwraca uwagę na jaskrę jako drugą główną przyczynę ślepoty na świecie. W porównaniu do katarakty, która jest odpowiedzialna za najwięcej przypadków utraty wzroku, jaskra jest dużo poważniejszym schorzeniem, ponieważ powoduje nieodwracalne uszkodzenia nerwu wzrokowego. Kolejnym czynnikiem przyczyniającym się do powagi tej choroby jest fakt, iż wczesne objawy są bardzo często ignorowane z powodu ich małej uciążliwości, bądź po prostu niezauważane [3].

Znacznym problemem w walce z jaskrą jest niedobór wyszkolonego personelu medycznego. Szacunkowo, w Europie jeden okulista przypada na 10 000 pacjentów. Sytuacja w innych częściach świata kształtuje się już znacznie gorzej. Przykładowo, w Indiach jeden okulista przypada na 400 000 pacjentów, a w Afryce mniej niż 1 na 1 000 000 osób [3].

Rozwiązańm pozwalającym na ograniczenie problemu występowania jaskry może być badanie przesiewowe. Prowadziłoby ono nie tylko do wczesnego wykrycia cech mających znamiona uszkodzenia nerwu, będących wskazaniem do wykonania rozleglejszych testów w kierunku jaskry, ale także na odciążenie personelu medycznego na rzecz personelu technicznego wykonujących badania.

Problemem napotykany podczas tworzenia badań przesiewowych jest klasyfikacja wyników wykonywanych testów. Ze względu na znaczącą liczbę badanych osób, klasyfikacja ta

powinna być wykonana szybko oraz bez udziału lekarzy. Obiecującym sposobem jest zastosowanie uczenia maszynowego, które nie tylko nie wymaga udziału człowieka, ale także zwraca wyniki w bardzo krótkim czasie.

Rozdział 3

Definicja jaskry

U. A. ROMANIUK

Jaskra to neuropatia nerwu wzrokowego, która polega na utracie komórek zwojowych siatkówki, co prowadzi do powstawania ubytków pola widzenia. Wraz z jej postępem ubytki te powiększają się i powielają, prowadząc do funkcjonalnej ślepoty. Dominującym patomechanizmem tej choroby jest zaburzenie równowagi pomiędzy produkcją cieczy wodnistej a możliwością jej odpływu przez kąt przesaczania, co skutkuje wzrostem ciśnienia śródgałkowego. Stan ten powoduje ucisk komórek siatkówki, zmniejsza przepływ w krążeniu siatkówkowym i zaburza transport w aksonach komórek zwojowych, co wiąże się z ich degeneracją. Jednakże etiologia tej choroby nie jest do końca poznana i bierze się pod uwagę także inne patomechanizmy [4, 5].

Pod względem klinicznym, jaskrę dzieli się na pierwotną i wtórną, a także na jaskrę z otwartym i wąskim kątem przesaczania. Najczęstszym występującym typem jest jaskra pierwotnie otwartego kąta, która występuje w 74% przypadków [6].

Przebieg tej choroby jest w większości przypadków bezobjawowy, ponieważ obwodowe ubytki pola widzenia zwykle nie sąauważane przez chorych. Jej podejrzenie wysuwa się w trakcie badania okulistycznego na podstawie oceny tarczy nerwu wzrokowego i pomiaru ciśnienia śródgałkowego. Cechami tarczy, które wzbudzają podejrzenie okulisty, jest: duża wartość *cup to disc ratio*, czyli stosunku pionowego wymiaru zagębienia tarczy do pionowego wymiaru całej tarczy, obecność krwotoczków na tarczy, atrofia okołobrodawkowa i nierówność brzegów rąbka tarczy nerwu wzrokowego [6].

Obecnie jedynym udowodnionym sposobem leczenia jest utrzymywanie ciśnienia śródgałkowego w odpowiednich granicach metodami farmakologicznymi lub zabiegowymi, co zatrzymuje lub spowalnia postęp choroby [6].

3.1. Wykrywanie jaskry

Rozpoznanie jaskry można postawić po stwierdzeniu charakterystycznych ubytków w badaniu pola widzenia i wykazaniu zmian w optycznej koherencyjnej tomografii (ang. *Optical Coherence Tomography*, OCT), która pozwala ocenić zarówno grubość warstwy komórek zwojowych, jak i warstwę włókien nerwu wzrokowego [7, 8, 9]. Skany otrzymane w badaniu OCT są porównywane do bazy normatywnej, dobranej pod kątem płci, rasy i wieku, co umożliwia ustalenie, czy u pacjenta wystąpiło uszkodzenie struktur dna oka.

Badania stosowane w diagnostyce jaskry nie nadają się do prowadzenia badań przesiewowych, ponieważ optyczna koherencyjna tomografia wymaga drogiego sprzętu, co ogranicza

jej dostępność, a badanie pola widzenia w perymetrze trwa stosunkowo długo i aby otrzymać wiarygodne badanie, pacjent musi „nauczyć się” wykonywać polecenie będące częścią badania, co wymaga odbycia kilku sesji. Ma to szczególne znaczenie w grupie ludzi starszych, najbardziej narażonych na jaskrowe uszkodzenie nerwu wzrokowego [6, 4].

Ocena dna oka w lampie szczelinowej lub oftalmoskopie pozwala na wysunięcie podejrzenia jaskry, jednak z powodu ograniczonego dostępu do badania okulistycznego, poszukuje się narzędzi umożliwiających prowadzenie badań przesiewowych. Jednym z proponowanych rozwiązań jest badanie w funduskamerze, którego koszty i czas przeprowadzania przez przeszkołonego technika są mniejsze, niż konsultacja lekarza okulisty. Wykonanie zdjęcia dna oka w takim aparacie nie wymaga podania leku rozszerzającego żyrenicę dospojówkowo pod warunkiem przeprowadzenia go w ciemni bądź otoczeniu nieznacznie zanieczyszczonym światłem.

Rozdział 4

Automatyczna klasyfikacja obrazów

I. M. SZOPA

Ostatnie lata niosą za sobą dynamiczny rozwój metod automatycznej klasyfikacji różnych obiektów w tym zdjęć oraz obrazów medycznych. Głównym elementem rozpoznawania obiektów, czyli projektowania i tworzenia automatycznych systemów segregacji, jest klasyfikator. Jest to algorytm, który na podstawie danych wejściowych, opisujących obiekt, przyporządkowuje go do danej klasy (klasy predykcji).

Najbardziej rozpowszechnionym typem metod automatycznej klasyfikacji obrazów są algorytmy uczenia maszynowego. Można je podzielić na uczenie z nadzorem (z nauczycielem) i uczenie bez nadzoru (samouczenie).

Uczenie z nauczycielem wymaga nadzoru człowieka przy tworzeniu funkcji, która odwzorowuje wejście na wyjście systemu. Do procesu uczenia wykorzystuje się zbiór znanych elementów. Taki zbiór można nazwać zbiorem uczącym lub zbiorem treningowym. Składa się on z kolekcji par, z których każda zawiera obiekt wejściowy i odpowiadające mu prawidłowe wyjście. Wejścia systemów mogą być bardzo zróżnicowane. Każdemu stworzonemu systemowi odpowiada wybrany typ wejścia, jednakże można zaprojektować klasyfikator o praktycznie dowolnym rodzaju danych wejściowych, takich jak zdjęcia, zapisy dźwięków bądź wektory cech. Analogicznie do wejść, wyjścia tworzonych programów również mogą być zróżnicowane.

Celem uczenia bez nadzoru jest odkrycie struktury dostarczonych danych bez pomocy z zewnętrz. W tym rodzaju systemów podawane są tylko dane wejściowe.

4.1. Metody uczenia maszynowego

Istnieje wiele metody uczenia z nadzorem, jednakże nie istnieje uniwersalny algorytm dający w każdym przypadku najlepsze wyniki statystyczne. Liczba poprawnie zaklasyfikowanych obrazów zależy od zbioru uczącego i sposobu uczenia. Najbardziej popularnymi metodami uczenia maszynowego są drzewa decyzyjne, maszyny wektorów wspierających, naiwny klasyfikator Bayesa oraz sieci neuronowe [10].

Wszystkie te metody, z wyjątkiem sieci neuronowych, są pokrótko opisane poniżej. Sieci neuronowe zostaną opisane w rozdziale 5.

4.1.1. Drzewa decyzyjne

Algorytmy oparte na drzewie decyzyjnym (ang. *Decision Tree*) są jednymi z najstarszych opracowanych metod. Można je stosować do różnych typów danych. Zasadą działania omawianego algorytmu jest znalezienie optymalnej funkcji (reprezentowanej w formie drzewa decyzyj-

nego), która oddziela klasy różnych danych od siebie. Struktura drzewa jest grafem, w którym węzły, krawędzie oraz liście mają swoje określone znaczenie. Węzły (ang. *nodes*) symbolizują testy, które są przeprowadzane na cechach opisujących dane. Krawędzie (ang. *edges*) odpowiadają wynikom tych testów, natomiast liście (ang. *leaves*) określają etykiety klas.

4.1.2. Maszyny wektorów wspierających

Maszyny wektorów wspierających (Maszyny wektorów nośnych, ang. *Support Vector Machine*, SVM) są jednymi z najpopularniejszych algorytmów uczenia maszynowego. Wykazują dobrą skuteczność nawet w wielowymiarowych przestrzeniach, to znaczy także w przypadku, w którym liczba cech może być większa niż liczba elementów w zbiorze danych opisywanych przez te cechy. SVM nie wymaga podania wielu parametrów początkowych, co nie przeszczadza w poprawnym klasyfikowaniu nawet złożonych hipotez. Maszyna wektorów wspierających tworzy zbiór hiperpłaszczyzn w przestrzeni maksymalnie rozdzielających przykłady z podanych klas.

Algorytm może być wykorzystany do klasyfikacji lub zadań takich jak wykrywanie wartości odstających.

4.1.3. Naiwny klasyfikator Bayesa

Naiwny klasyfikator Bayesa (ang. *Naive Bayes Classifier*) to metoda oparta na twierdzeniu sformułowanym przez Thomasa Bayesa [11]. Algorytm stosujący jego tezę to klasyfikator oparty na wnioskowaniu statystycznym. Jego naiwność przejawia się poprzez założenie niezależności zmiennych losowych reprezentujących cechy klasyfikowanych obiektów. Postulat ten prawie nigdy nie jest prawdziwy w zagadnieniach rzeczywistych.

Rozdział 5

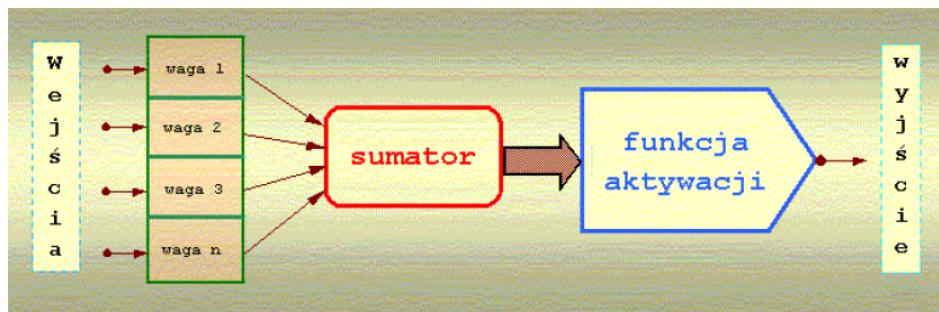
Sztuczne sieci neuronowe

I. M. SZOPA

Pierwotnym celem stworzenia sztucznych sieci neuronowych była kreacja systemu obliczeniowego, którego zasada działania byłaby analogiczna do funkcjonowania mózgu.

Podstawową jednostką sieci jest sztuczny neuron. Neuron otrzymuje jedno lub wiele wejść x_i , każde z przypisaną mu indywidualną wartością zwaną wagą w_i . Sumę ważoną N sygnałów wejściowych wraz z dodatkowym składnikiem stałym zwanym obciążeniem b określa się mianem ekscytacji neuronu. Wartość ekscytacji przekształcana jest przez funkcję aktywacji f i wysyłana na wyjście jednostki (wzór 5.1, rysunek 5.1). Wyjście to może stać się sygnałem wejściowym dla kolejnych neuronów. Sygnał wyjściowy oznaczany jest przez y .

$$y = f\left(\sum_{i=1}^N w_i x_i + b\right) \quad (5.1)$$



Rysunek 5.1: Schemat działania sztucznego neuronu [12].

Funkcja aktywacji może przyjmować różne postaci w zależności od oczekiwanej działania sieci. Powinna być łatwa do obliczania, ponieważ jest wykonywana tysiące albo nawet miliony razy dla każdego elementu zbioru uczącego.

Funkcja może przybierać jedną z trzech postaci: progowa, liniowa, nieliniowa. Postać ta wybierana jest na podstawie problemu, jaki ma do rozwiązania sieć. Przy wielowarstwowych sieciach neuronowych najczęściej stosowane są funkcje nieliniowe. Daje to możliwość odwzorowywania dowolnej zależności między wejściem a wyjściem sieci.

Najczęściej używanymi funkcjami aktywacji są funkcje nieliniowe f o argumentem u . Najpopularniejszymi z nich są: funkcja sigmoidalna (wzór 5.2), tangens hiperboliczny (wzór 5.3)

i funkcja prostownicza (wzór 5.4).

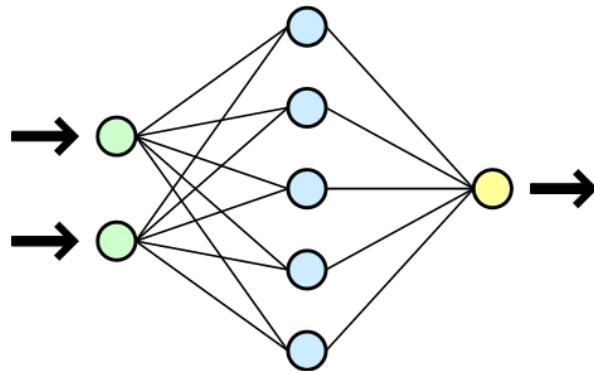
$$f(u) = \frac{1}{1 + \exp^{-u}} \quad (5.2)$$

$$f(u) = \frac{\exp^u - \exp^{-u}}{\exp^u + \exp^{-u}} \quad (5.3)$$

$$\begin{aligned} f(u) &= \max(0, c) \\ c &= Wx + b \end{aligned} \quad (5.4)$$

Istnieje również funkcja *Softmax*, która jest znormalizowaną funkcją wykładniczą. Oblicza ona prawdopodobieństwo zdarzenia w różnych n zdarzeniach. Ogólnie rzecz biorąc, funkcja ta określa prawdopodobieństwo każdej docelowej klasy we wszystkich możliwych klasach. Tak otrzymane prawdopodobieństwa są używane do określania klasy podawanych danych. Ta funkcja, w odróżnieniu od pozostałych funkcji aktywacji, jest wykonywana tylko i wyłącznie na samym końcu uczenia sieci.

Budowa najprostszej sieci składa się z trzech typów warstw: warstwy wejściowej, ukrytej oraz wyjściowej (rysunek 5.2). Warstwa wejściowa przyjmuje dane wejściowe dostarczane do sieci. Warstw ukrytych może być kilka, ich działania nie można bezpośrednio obserwować ani na wejściu, ani na wyjściu. Liczba neuronów zawartych w warstwie wyjściowej odpowiada liczbie docelowych klas.



Rysunek 5.2: Schemat jednowarstwowej sztucznej sieci neuronowej [13].

5.1. Typy sieci neuronowych

Modele sieci neuronowych różnią się zarówno rodzajem neuronów, które je tworzą, jak i budową oraz układem połączeń międzyneuronalnych lub sposobem propagacji sygnałów w obrębie sieci. Można wyróżnić dwa najczęściej stosowane typy:

- sieci jednokierunkowe (ang. *Feedforward networks*),
- sieci rekurencyjne (ang. *Recurrent networks*).

W sieci jednokierunkowej nie występują żadne sprzężenia zwrotne. Na każdy neuron w warstwie przekazywany jest tylko sygnał, który pochodzi z poprzedniej warstwy. To oznacza, że sieć działa tylko w kierunku od wejścia do wyjścia. Ze względu na swoją prostotę sieci jednokierunkowe są najbardziej rozpowszechnionymi sieciami neuronowymi.

W sieć rekurencyjnej występują sprzężenia zwrotne, które polegają na podawaniu wyjścia sieci na jej wejście. Niektóre z implementacji sieci pozwalają na sprzężenie bezpośrednie, które sprzęga neuron z nim samym. Wśród różnorodności sieci rekurencyjnych można wyróżnić: sieć Hopfielda i maszynę Boltzmanna. Stosuje się je na przykład jako pamięci adresowane kontekstowo do rozpoznawania obrazów [14], mowy [15], a także do rozwiązywania problemów minimalizacji [16].

5.2. Uczenie sieci neuronowych

Główną zaletą sieci neuronowych jest fakt, że nie wymagają ręcznej ekstrakcji cech pochodzących z danych wejściowych. Poddaje się taką sieć procesowi uczenia, czyli takim doborze współczynników wag, które są optymalne do klasyfikacji podanego problemu. W większości przypadków uczenia maszynowego sieci stosuje się algorytm wstecznej propagacji, który jest jedną z najstarszych metod uczenia. Oprócz niego istnieje wiele innych algorytmów.

5.2.1. Propagacja wsteczna

Algorytm wstecznej propagacji (ang. *Back Propagation*) ma za zadanie zminimalizować funkcję błędu (funkcję kosztu). Funkcja ta określa błąd popełniany przez sieć na zbiorze uczącym. Intuicyjnie pojęciem błędu określa się różnicę między wartościami wektora wyjściowego, otrzymanymi przez układ w trakcie propagacji przedniej (ang. *Forward Propagation*) (przebieg informacji w sieci od warstwy wejściowej do wyjściowej), a wartościami prawdziwymi. Odejmowanie daje wartości ujemne, jak i dodatnie, przez co nie jest dobrą miarą błędu, dla którego znak nie ma znaczenia. Można zastosować wartość bezwzględną, jednak nie jest ona różniczkowalna, a warunek ten wymagany jest to w tym problemie optymalizacyjnym. Udoskonaloną formą funkcji kosztu, która również eliminuje problem ujemnych wartości, jest funkcja błędu średniokwadratowego [17]. Inną stosowaną funkcją jest entropia krzyżowa [18].

Do minimalizacji stosuje się algorytm spadku gradientowego, którego postulat mówi, że zmiana funkcji dążącej do minimalizacji parametrów musi przebiegać przeciwnie do gradientu tej funkcji. Algorytm zaczyna iterację w losowym punkcie i przesuwa się w stronę minimum, aktualizując wagi sieci.

Ważnym parametrem minimalizacji błędu jest współczynnik uczenia (ang. *learning rate*), który odpowiada za wielkość krok wykonywanego przez algorytm w trakcie poszukiwania minimum globalnego. Jeśli będzie on zbyt mały, to funkcja będzie bardzo wolno zbiegała, natomiast za duża wartość może spowodować, że algorytm nie znajdzie minimum.

Rozdział 6

Sieci neuronowe typu deep learning

I. M. SZOPA

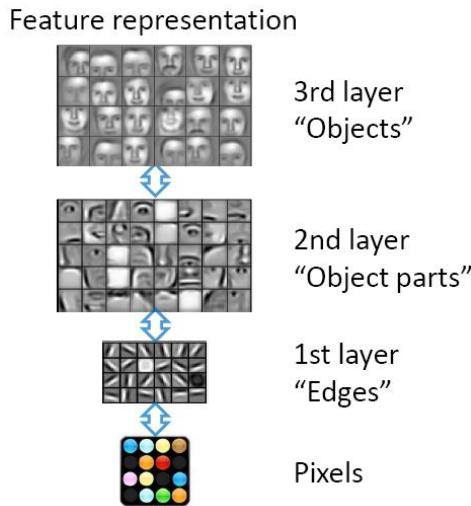
Głębokość architektury sieci neuronowych jest zdefiniowana jako długość ścieżki między neuronem wejściowym a neuronem wyjściowym. W przypadku sieci jednokierunkowych przekłada się ona bezpośrednio na liczbę warstw. Nie ma ściśle określonej liczby warstw, od której sieć nazywa się głęboką. Przyjęte jest, że sieć powyżej dwóch warstw ukrytych jest już siecią głęboką, jednak przy obecnym rozwoju coraz większych sieci ta granica może zostać przesunięta.

Głębokie uczenie (ang. *deep learning*) opiera się na metodach uczenia maszynowego, zbudowanych z wielu warstw nieliniowych transformacji. Wszystkie warstwy tworzą własny, pełny opis danych obiektu wejściowego. Każda warstwa uczy się przekształcania danych w coraz bardziej abstrakcyjną i złożoną reprezentację. Pierwsza warstwa może poszukiwać krawędzi i różnice w kolorze pomiędzy sąsiadującymi pikselami, kolejna warstwa może tworzyć układy krawędzi, a następna jeszcze rozpoznawać nos, oczy lub już całe obiekty. Taka budowa modelu nazywana jest budową hierarchiczną (rysunek 6.1).

Z twierdzenia o uniwersalnej aproksymacji (zmodyfikowane twierdzenie Kołmogorowa [19]) wynika, że sieć neuronowa z jedną warstwą ukrytą jest w stanie aproksymować dowolną funkcję z dowolną dokładnością. Niestety, wraz z rosnącą złożonością problemu, liczba neuronów w warstwie ukrytej rośnie wykładniczo. Wadą takiej architektury jest posiadanie zbyt dużej liczby parametrów sieci co prowadzi do zjawiska przeuczania (ang. *overfitting*).

W idealnym przypadku sieć neuronowa powinna nauczyć się generalizacji cech obrazów treningowych. Przeuczanie (przetrenowanie) prowadzi do zbytniego dopasowania się sieci do istniejących danych. Zjawisko to powoduje, że bardzo dobrze rozpoznawane są obiekty treningowe, jednak sieć nie jest w stanie rozpoznać tego samego typu obrazów z innego zbioru.

Przy mniej skomplikowanych problemach, jak i problemach bez struktury hierarchicznej, działanie sieci głębokiej będzie gorsze od sieci płytkiej.



Rysunek 6.1: Schemat hierarchii warstw w sieci głębokiej [20].

Idea głębokich sieci neuronowych powstała już w latach 70., jednak istniały powody, przez które sieci te nie rozwijały się zbyt szybko:

- Wraz z rosnącą liczbą warstw rośnie również liczba parametrów wymaganych do nauczania modelu. Ucząc dużo warstw, ale na małym zbiorze danych, bardzo szybko występuje przeuczenie sieci.
- Zwiększając rozmiar sieci, automatycznie zwiększa się ilość danych do niej podawanych. Wymagania sprzętowe do przetworzenia takiej ilości informacji są bardzo wysokie.

6.1. Konwolucyjne sieci neuronowe

Konwolucyjne sieci neuronowe (ang. *Convolutional Neural Networks*) są sieciami o budowie jednokierunkowej. Sieci te zostały specjalnie zaprojektowane do klasyfikacji obrazów, ponieważ zwykła sieć miałaby ogromną ilość parametrów w każdej warstwie ukrytej, co szybko prowadzi do przetrenowania.

Inspiracją dla sieci konwolucyjnych jest ludzki układ wzrokowy, którego neurony aktywują się tylko wtedy, gdy poszukiwane elementy znajdują się w danym kawałku pola widzenia.

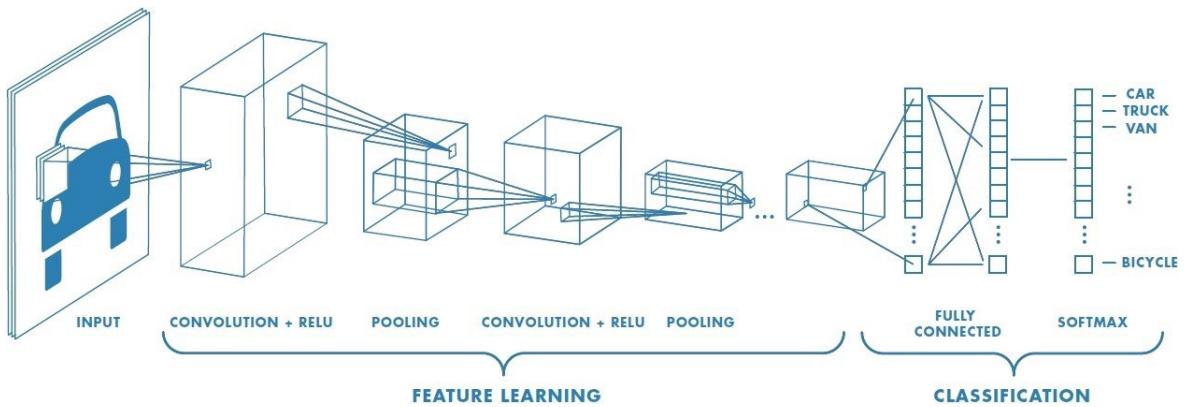
Wejście sieci konwolucyjnej jest w postaci trójwymiarowej macierzy, określonej przez szerokość i wysokość obrazu (zazwyczaj w pikselach) oraz grubość (liczba kanałów w systemie RGB). Warstwy ukryte składają się z map, które odpowiadają obrazowi wejściowemu. Mapy mają za zadanie wyszukiwanie wzorców graficznych, które reprezentowane są przez filtry z nimi powiązane. Filtr ma postać macierzy wag połączzeń między warstwą poprzednią a neuronem w danej mapie. Pole recepcyjne, czyli pole widzenia filtra, jest to obszar macierzy wejściowej, który łączy się z konkretnym neuronem bezpośrednio lub przez inne neurony. Wielkość takiego obszaru zależy od warstwy, w której znajduje się podany neuron albo od parametrów warstw poprzednich. Aktywacja na neuronie odwzorowuje podobieństwo obrazu, który zawiera się w polu recepcyjnym, do wzorca zawartego w filtrze.

Ponadto, sieć konwolucyjna redukuje pełne obrazy do pojedynczych wektorów wyjściowych reprezentujących wyniki dla poszczególnych klas.

6.1.1. Architektura

Kompletna architektura sieci konwolucyjnej (rysunek 6.2) składa się z następujących warstw:

- wejściowej (ang. *input*),
- konwolucyjnej (ang. *convolution*),
- aktywacyjnej (ang. *activation*),
- głosującej (ang. *pooling*),
- w pełni połączonej (ang. *fully connected*),
- stosującą funkcję Softmax.



Rysunek 6.2: Schemat przykładowej sieci konwolucyjnej [21].

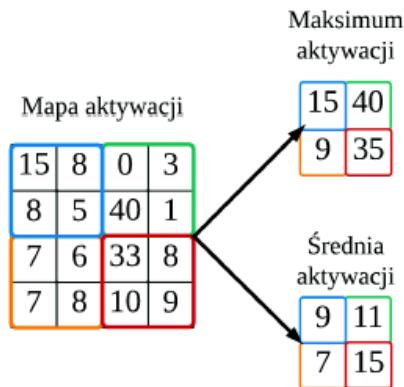
W standardowej architekturze sieci konwolucyjnej wejście stanowi kwadratowy obraz w systemie RGB.

Warstwa konwolucyjna (ang. *convolution layer*) albo splotowa to zbiór map filtrów, czyli cech obrazu. Pierwsza warstwa jest bardzo często używana jako warstwa wejściowa. Każda z warstw jest trójwymiarowa. Konwolucja lub splot jest iloczynem skalarnym wejścia sygnału z filtrem.

Warstwa aktywacyjna nie jest tak naprawdę oddzielną warstwą. Aplikuje ona funkcję aktywacji do każdego neuronu w warstwie konwolucyjnej. Nie zmienia wymiarów reprezentacji danych. W sieciach konwolucyjnych podstawowo stosowaną funkcją aktywacji jest funkcja prostownicza (ang. *ReLU*, *Rectified linear unit*), ponieważ pokonuje ona problem zanikającego gradientu [22] pozwalając na lepsze i szybsze uczenie sieci.

Warstwa głosująca (ang. *pooling layer*) wyciąga z lokalnego zakresu mapy neuronów proste statystyki o aktywacji. Ma to na celu redukcję wymiarów sieci. Najczęściej stosowane są statystyki maksymalnej aktywacji (ang. *max pooling*) lub średniej aktywacji (ang. *mean pooling*), przykładowe działania przedstawiono na rysunku 6.3.

Warstwa w pełni połączona (ang. *fully connected layer*) występuje na końcu większości modeli konwolucyjnych. Wykorzystuje się ją na ostatnim etapie przetwarzania danych, ponieważ wyciąga ona wnioski na temat danego problemu ze znalezionych cech. Nie różni się ona specjalnie funkcjonalnością od warstwy konwolucyjnej. Obliczany jest w niej iloczyn skalarny wejścia oraz określonych wag.



Rysunek 6.3: Schemat działania warstwy głosującej [23].

Warstwa stosująca funkcję Softmax jest używana jako ostatnia warstwa w sieciach neuronowych, ponieważ rozwiązuje problemy klasyfikacyjne. Wartości wyjściowe sieci mogą być interpretowane jako oszacowania prawdopodobieństw przynależności wybranego sygnału wejściowego do poszczególnych klas. Funkcja Softmax jest dokładniej opisana w rozdziale 5.

Liczba używanych warstw danego typu jest zależna od złożoności podawanych danych i ilości klas, do których mają być przydzielone. Nie ma dokładnego przepisu na architekturę pasującą do klasyfikacji konkretnego problemu stawianego przed siecią konwolucyjną.

6.1.2. Zastosowania

Istnieje kilka architektur, które w konwolucyjnych sieciach neuronowych są znane z nazwy:

- LeNet – pierwsza sieć neuronowa zbudowana przez Yanna LeCun'a w 1990 roku [24]. Jedna z architektur LeNetu była używana do czytania kodów pocztowych, cyfr itp.
- AlexNet – pierwsza praca, która zaczęła popularyzację sieci konwolucyjnych w wizji komputerowej. Stworzona przez Alexa Krizhevsky'ego, Ille Sutskever i Geoffa Hintona [25]. AlexNet jako pierwsza sieć konwolucyjna wygrała *ImageNet ILSVRC challenge* [26] w 2012 roku. Architekturę ma bardzo podobną do LeNetu, ale jest większa, głębsza i posiada szereg warstw konwolucyjnych umieszczonych po sobie (wcześniej używano tylko jednej warstwy konwolucyjnej).
- ZF Net – zwycięska sieć konwolucyjna *ILSVRC* z 2013 stworzona przez Matthew Zeilera oraz Roba Fergusa [27]. Autorzy ulepszyli architekturę AlexNetu poprzez poprawienie parametrów początkowych uczenia sieci. W szczególności powiększyli rozmiar średkowych warstw konwolucyjnych sieci oraz zmniejszyli krok i rozmiar filtra pierwszej warstwy.
- GoogLeNet – sieć zaprojektowana przez Christiana Szegedy'ego i in. z firmy Google, wygrała konkurs *ILSVRC* w 2014 roku [28]. Głównym wkładem było opracowanie modułu początkowego (ang. *Inception module*), który znacznie zmniejszył liczbę parametrów w sieci (4 miliony w porównaniu do 60 milionów AlexNetu). Dodatkowo używają w sieci warstw *average pooling* zamiast warstw w pełni połączonych, co eliminuje dużą część parametrów, które wydają się nieistotne.

- VGGNet – sieć, która w roku 2014 zajęła drugie miejsce w *ILSVRC*, stworzona przez Karen Simonyan i Andrew Zissermana [29]. Pokazali oni, że głębokość sieci jest kluczowym elementem do osiągnięcia dobrych wyników. Ich najlepsza sieć ma łącznie 16 warstw konwolucyjnych i w pełni połączonych. Ma wyjątkowo jednorodną architekturę, wykonuje tylko 3×3 konwolucje i 2×2 *pooling* od początku do końca. Problemem jest to, że jest dużo droższa obliczeniowo do oceny i potrzebuje dużo pamięci oraz parametrów (140 milionów). Większość tych parametrów jest w pierwszej warstwie w pełni połączonej, wykazano, że usunięcie tych warstw nie pogarsza znacząco wyników, a dzięki temu zmniejszona jest liczba parametrów.
- ResNet – *Residual Network* zaprojektowana przez Kaiminga He i in. zwyciężyła w konkursie *ILSVRC* w 2015 roku [30]. Stosuje funkcje do pomijania połączeń i bardzo dużo funkcji normalizacyjnych (ang. *batch normalization*). Nie posiada również warstw w pełni połączonych na końcu sieci. ResNet jest w tej chwili najczęściej wybieraną siecią konwolucyjną.

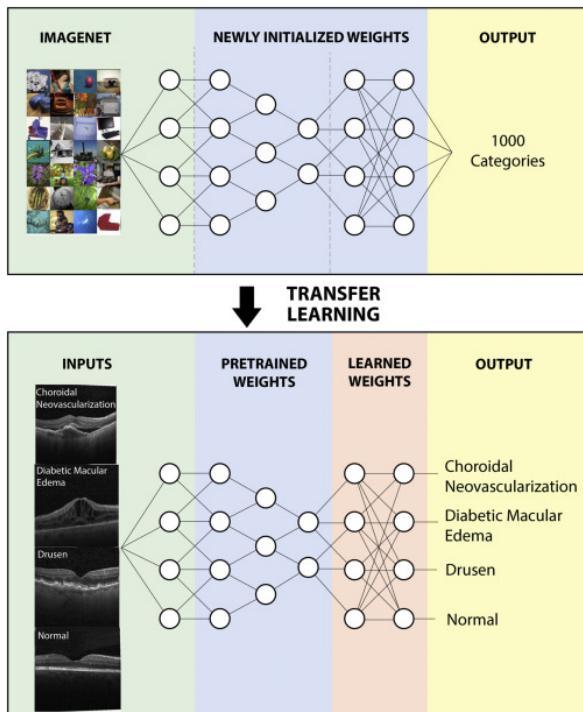
Rozdział 7

Transfer learning

U. A. ROMANIUK

Obecnie pojęcie *transfer learningu* zyskuje na popularności. W ciągu ostatnich 5 lat liczba zapytań hasła *transfer learning* w wyszukiwarce Google wzrosła o 150% [31]. *Transfer learning* można rozumieć jako przeniesienie wiedzy, ale nie posiada polskiego odpowiednika.

W uczeniu maszynowym metoda ta polega na wykorzystaniu umiejętności modelu nabytych podczas trenowania na pewnym zbiorze danych w celu nauczenia sieci rozpoznawania nowych kategorii. Schemat działania *transfer learningu* został przedstawiony na rysunku 7.1.



Rysunek 7.1: Schemat działania *transfer learningu*. Przykładowa architektura sieci przystosowana do danych wejściowych w postaci obrazu (ze zbioru danych ImageNet) oraz wyjście w postaci klasy. Po zastosowaniu *transfer learningu* na wejście wstawiane są nowe obrazy. Architektura sieci zostaje zmieniona, na wyjściu znajduje się oczekiwana ilość klas. Część wag na początkowych warstwach zostaje zachowana, a część wag zostaje zmieniona (doucza się) [32].

Istnieje również pojęcie *fine-tuningu*, nierozerwalnie związane z *transfer learningiem*. *Fine-tuning* to jedna z metod *transfer learning*. Metoda ta polega na użyciu modelu, który jest już częściowo wytrenowany i dotrenowaniu go, ale wagi nie są już inicjowane losowo. Dzięki zastosowaniu wstępnej propagacji dokonuje się aktualizacji uprzednio wyszkolonych wag, przy czym liczba węzłów w warstwie wyjściowej musi zostać dopasowana do ilości klas predykcji.

Transfer learning to szersza kategoria, gdzie chcemy przenieść wiedzę zdobytą na uczeniu sieci z jednej dziedziny do innej. Powiązana nazwa to *meta-learning* lub *learning to learn*, gdzie celem jest nauczenie sieci działania w różnych środowiskach. *Transfer learning* polega na uczeniu sieci od podstaw na dużym zbiorze danych, który jest dobrze zaklasyfikowany, aby rozpoznawać ogólne cechy obrazu. Następnie, do uczenia dokładany jest mniej liczny zbiór uczący, który zawiera inną kategorię danych, niż zbiór początkowy. Dzięki temu sieć specjalizuje się w rozpoznawaniu konkretnych cech obrazu dla danej kategorii. Można także wykorzystać wcześniej przeszkoloną sieć i dodać ją. *Transfer learning* pozwala na interencję w architekturę sieci. Można, między innymi, dodać warstwę, zmienić ilość neuronów w warstwie, albo szybkość uczenia się sieci. Zamiast uczyć całą sieć od nowa, można zachować wagi na niższych warstwach, które odpowiadają za rozpoznawanie kształtów, kolorów i niewyraźnych zarysów, a zmieniać wagi jedynie na warstwach wyższych. Uzasadnieniem takiego postępowania jest fakt, że dokładne dopasowanie i szczegóły na obrazie uwzględniane są dopiero przez warstwy późniejsze. *Transfer learning* stosuje się do uczenia sieci w sytuacji dostępu do stosunkowo niewielkiej ilości danych [32, 33, 34].

W obu przypadkach dane wejściowe muszą być dostosowane do rozmiaru wejścia sieci. Wybór pomiędzy dwoma sposobami zależy od rozmiaru wejścia sieci i od podobieństwa nowego zestawu danych do oryginalnego zestawu, na którym uczyła się sieć. *Fine-tuning* stosuje się, gdy nowy zestaw danych jest wystarczająco duży [32, 34].

Granice pomiędzy tymi definicjami nie są precyzyjnie wyznaczone. Zauważono, że pojęcia *transfer learning* i *fine-tuning* często są ze sobą mylone lub używane zamiennie. Różnice między tymi pojęciami są subtelne. Przyjmuje się, że w *fine-tuningu* nie zmienia się model, odwrotnie niż w przypadku *transfer learningu*. W fine tuningu zbiór nowej kategorii musi być podobny do wcześniejszej użytego zbioru podczas uczenia tej sieci. W przypadku *transfer learning* zmieniony może być zbiór, na którym sieć jest douczana.

Transfer learning jest często wykorzystywany przy uczeniu sieci na danych medycznych [35].

Rozdział 8

Wytwarzanie sztucznych obrazów uczących

U. A. ROMANIUK

Wytwarzanie sztucznych obrazów uczących (ang. *data augmentation*), pozwala na poprawienie wyników klasyfikatora oraz zmniejszenia przeuczania się sieci, ze względu na poprawę generalizacji zbioru uczącego. *Data augmentation* jest powszechnie stosowane niezależnie od wielkości zbioru uczącego, jednakże najbardziej widoczne rezultaty osiągane są w przypadku posiadania niewielu obrazów uczących.

Wytwarzanie sztucznego zbioru danych polega na edycji oryginalnego obrazu tak, aby powstało kilka obrazów z jednego. W tym celu stosuje się transformacje geometryczne i barw. W przypadku transformacji geometrycznych wykorzystuje się rotację, odbicie wzdłuż pionowej i pionowej osi, skalowanie obrazu i wiele innych [34]. Transformacje barw mogą polegać na zmianach kontrastu, koloru i ostrości zdjęcia.

Często stosowanym sposobem jest tworzenie kilku obrazów z jednego, poprzez wycięcie środka oraz brzegów obrazu z zachowaniem ważnych informacji, a następnie uśrednieniu wyniku [33].

Rozdział 9

Przegląd publikacji medycznych

U. A. ROMANIUK

W wielu artykułach medycznych opisane jest zastosowanie *transfer learningu* do tworzenia klasyfikatora. Sztuczne powiększanie danych jest częstym zabiegiem stosowanym, aby poprawić wyniki modelu.

W pracy Hussain Zeshan, Gimenez Francisco, Yi Darvin i Rubin Daniel *Differential data augmentation techniques for medical imaging classification tasks* [36] przedstawiono wpływ sztucznego powiększania danych uczących na wynik sieci. Badacze użyli obrazów medycznych, 1650 mammogramów osób z rakiem piersi i 1650 mammogramów osób zdrowych. Zbiór podzielono na zbiór treningowy 80% i testowy 20%. Sprawdzano 8 różnych strategii, jakie mogą poprawić wynik klasyfikatora. Między innymi: odbicie obrazu w płaszczyźnie poziomej i pionowej, dodanie szumu gaussowskiego do każdego piksela, skalowanie obrazu czy zwiększenie kontrastu obrazu. Obraz przeskalowano z oryginalnego wymiaru do 1000×1000 pikseli, następnie do 500×500 pikseli, później do 224×224 pikseli. Użyto sieci typu VGG-16. Dane znormalizowano i ustalono parametry uczenia. W większości wyniki uległy poprawie po zastosowaniu *data augmentation*. Najlepszy wynik dokładności to 0,881, który powstał po dodaniu szumu gaussowskiego.

W pracy *An ensemble of fine-tuned convolutional neural networks for medical image classification* autorów Kumar Ashnil, Kim Jinman, Lyndon David, Fulham, Michael i Feng Dagan [33] zbiór danych zawierał obrazy medyczne opublikowane w *ImageCLEF* 2016. Dane zawierały 30 kategorii, 6776 obrazów treningowych i 4166 obrazów testowych. Zastosowano *data augmentation*, aby powiększyć zbiór uczący. Stworzono 10 folderów. Obrazy przygotowano według schematu: przycięto obrazy do rozmiaru wejścia sieci, następnie wycięto 5 obrazów z jednego oryginalnego (zachowując środek i brzegi obrazu), wygenerowano dodatkowo pięć obrazów, używając obrotów wzduż osi poziomej z każdego wyciętego obrazu (z poprzedniego punktu). Zbiór danych treningowych powiększono 10-krotnie. Następnie zastosowano dwa podejścia. Pierwsze to użycie wektorów nośnych (SVM), a drugie to klasyfikator z funkcją Softmax. Dla każdego przypadku sprawdzono dwie sieci: AlexNet i GoogLeNet. Zastosowano *fine-tuning* i *transfer learning*. Przede wszystkim zmieniono ostatnią warstwę, aby na wyjściu miała 30 klas, czyli tyle ile było kategorii obrazów. Eksperyment pokazał ok. 80% skuteczność zastosowanych metod.

9.1. Publikacje medyczne dotyczące automatycznego wykrywania jaskry

W pracy *Performance assessment of the deep learning technologies in grading glaucoma severity* autorów Zhen Yi, Wang Lei, Liu Han, Zhang Jian i Pu Jiantao [37] zbiór danych zawierał zdjęcia dna oka wykonane funduskamerą. Zdjęcia zostały zaklasyfikowane przez dwóch okulistów na 4 kategorie: brak jaskry, łagodna, umiarkowana i zaawansowana jaskra. Razem zebrano prawie 6000 obrazów. Zdjęcia przyjęto, zachowując tarczę nerwu wzrokowego. Sprawdzano osiem architektur: VGG16, VGG19, ResNet, DenseNet, InceptionV3, Inception-ResNet, Xception, i NASNetMobile. Zastosowano *transfer learning*, aby dostosować sieć do rozpoznawania jaskry. Dodano dwie warstwy w pełni połączone.

Do powiększania zbioru uczącego użyto odbić wzduż linii pionowej i poziomej, przeskakowanie obrazów o wartości 0,85 - 1,15 oraz rotacje obrazów o kąt 30° w dwie strony. Wyniki pokazują, że użycie *transfer learningu* poprawia rozpoznawanie jaskry przez sieć. Początkowo dokładność wynosiła 0,83% a po zastosowaniu *transfer learningu* 0,93%.

Rozdział 10

Metody analizy statystycznej

10.1. Ocena klasyfikatora

U. A. ROMANIUK

Klasyfikatory binarne mają za zadanie przypisanie danemu wynikowi jednej z dwóch możliwych klas. Może to być klasyfikacja na osoby chore i zdrowe. Pewien klasyfikator na wejściu dostaje dane i na wyjściu zwraca, określając na podstawie tych danych, czy pacjent jest zdrowy czy chory. W medycynie przyjmuje się, że pozytywny wynik testu oznacza, że pacjent jest chory. Znając odpowiedź klasyfikatora i faktyczny stan zdrowia możemy podzielić rezultaty na cztery przypadki (tabela 10.1).

	klasa przewidywana prawdziwa	klasa przewidywana nieprawdziwa
klasa rzeczywista prawdziwa	prawdziwie pozytywna PP	falszywie negatywna FN
klasa rzeczywista nieprawdziwa	falszywie pozytywna FP	prawdziwie negatywna PN

Tabela 10.1: Macierz błędów klasyfikacji binarnej.

Wynik testu prawdziwie pozytywny (PP) oznacza poprawne zaklasyfikowanie osoby chorej. Analogicznie, rezultat prawdziwie negatywny (PN) jest to poprawna klasyfikacja osoby zdrowej. W przypadku kiedy przez klasyfikator zwracana jest kategoria oznaczająca chorobę dla ludzi zdrowych, wyniki oznaczane są jako falszywie pozytywne (FP). Jest to błąd I rodzaju lub inaczej nazywane falszywym alarmem. Błąd II rodzaju, równoznaczny z klasyfikacją falszywie negatywną (FN), występuje gdy, zaklasyfikowano ludzi chorych jako zdrowych.

Powszechnie stosowanymi miarami wartości diagnostycznej klasyfikatora są: czułość (wzór 10.1), dokładność (wzór 10.2), precyzja (wzór 10.3) oraz specyficzność (wzór 10.4). W celu ich wyznaczenia każdemu elementowi zbioru walidacyjnego należy przyporządkować jedną z czterech kategorii zawartej w macierzy błędów (tabela 10.1). Następnie zliczana jest liczba wystąpień tych kategorii i podstawiana do wzorów.

$$\text{czułość} = \frac{\text{PP}}{\text{PP} + \text{FN}} \quad (10.1)$$

$$\text{dokładność} = \frac{\text{PP} + \text{PN}}{\text{PP} + \text{PN} + \text{FP} + \text{FN}} \quad (10.2)$$

$$\text{precyzja} = \frac{\text{PP}}{\text{PP} + \text{FP}} \quad (10.3)$$

$$\text{specyficzność} = \frac{\text{PN}}{\text{PN} + \text{FP}} \quad (10.4)$$

10.2. Walidacja krzyżowa

U. A. ROMANIUK

Walidacja krzyżowa (ang. *cross-validation*) polega na stworzeniu K losowych podziałów zbioru danych wejściowych klasyfikatora. Każdy podział tworzy dwa rozłączne podzbiory nazywane uczącym i testowym. Dla każdego podziału tworzony jest klasyfikator uczony na zbiorze treningowym. Jego ocena przeprowadzana jest na odpowiadającym mu zbiorze walidacyjnym. Dla tak otrzymanego zbioru miar oceny klasyfikatorów obliczane są jego statystyki, takie jak średnia oraz odchylenie standardowe. Uśredniona miara wraz z oceną błędu jest bardziej wiarygodną oceną wyników klasyfikatora.

10.3. Testy statystyczne

K. A. FILIPIUK

W celu porównania ze sobą dwóch różnych klasyfikatorów powinno się zastosować odpowiednie testy statystyczne. Zbiory miar jakości otrzymane w wyniku walidacji krzyżowej stanowią jedynie próbę losową reprezentującą ogólną populację możliwych rezultatów. Stawiając hipotezę dotyczącą relacji średnich miar dwóch modeli, należy zbadać ją przy pomocy właściwej statystyki oraz rozkładu testowego.

Istnieją dwa rodzaje testów statystycznych. Należą do nich testy parametryczne i nieparametryczne.

Testy parametryczne charakteryzują się większą mocą – analizy dokonane przy ich pomocy są dokładniejsze, a wyniki łatwiejsze do interpretowania. Niestety, pewna liczba założeń musi być spełniona, aby można było je stosować. W większości przypadków założeniem tym jest normalność rozkładu, z którego pochodziły obserwacje.

Testy nieparametryczne nie wymagają spełnienia założenia o normalności rozkładu populacji, z której pochodzą obserwacje. Niestety, charakteryzują się mniejszą efektywnością w porównaniu do testów parametrycznych. Niewątpliwą ich zaletą jest odporność na obserwacje odstające, które nie posiadają test parametryczne.

Od prawidłowego wyboru testu statystycznego zależy może poprawność i dokładność wnioskowania. W niniejszej pracy zastosowane zostaną testy parametryczne, jednakże dopiero po upewnieniu się, że zostało spełnione założenie o normalności rozkładu, z którego pochodzi obserwacja. Hipoteza ta będzie potwierdzana przy użyciu testu Shapiro-Wilka. Docelową statystyką stosowaną do porównywania równości średnich miar jakości klasyfikatorów będzie test t-Studenta. Wymaga on jednakże spełnienia dodatkowego założenia o równości wariancji w porównywanych populacjach. W tym celu, poprzedzając test równości średnich, zastosowany zostanie test Fishera-Snedecora.

10.3.1. Test Shapiro-Wilka

Test Shapiro-Wilka (test S-W) został zaproponowany przez Samuela Shapiro oraz Martina Wilka w 1965 roku dla mało licznych grup [38], a następnie zaadoptowany dla grup o większej liczebności przez Roystona w 1992 roku [39]. W porównaniu do testu Kołmogorova-Smirnova pochodzącego z 1933 roku [40], test S-W wykazuje większą moc.

Test S-W służy do oceny podobieństwa badanego rozkładu do rozkładu normalnego. Hipotezą zerową omawianego testu jest pochodzenie wartości testowanej próby losowej z rozkładu Gaussa.

Statystyka testowa Shapiro-Wilka ma postać przedstawioną we wzorze 10.5, gdzie poprzez W oznaczono wartość statystyki testowej. Obserwacje X , których liczba wynosi n są uszeregowane w kolejności od najmniejszej do największej. Aby to podkreślić, indeksy dolne zostały umieszczone w nawiasach okrągłych. W konsekwencji tego, obserwacja z indeksem (1) stanowi minimum zbioru, a z indeksem (n) - maksimum zbioru. Zmienne i i j iterują po wszystkich parach występujących wartości. Wartość $a_i(n)$ nie zależy od wartości obserwacji i jest stablicowana. Symbol \bar{X} oznacza średnią z badanej próby.

$$W = \frac{(\sum_{i=1}^n a_i(n) X_{(i)})^2}{\sum_{j=1}^n (X_j - \bar{X})^2} \quad (10.5)$$

Statystyka testowa S-W ma rozkład W . Przy pomocy tablic tego rozkładu można odczytać wartość krytyczną i na tej podstawie przyjąć bądź odrzucić hipotezę zerową.

10.3.2. Test Fishera-Snedecora

Test Fishera-Snedecora powstał w 1989 roku, kiedy to George Snedecor opisał rozkład stosunku wariancji dwóch prób [41]. Pierwszy człon nazwy tego testu został dodany przez samego Snedecora, aby uhonorować pracę Ronaldiego Fishera, który w 1920 roku stworzył statystykę stosunku wariancji [42].

Hipotezą zerową tego testu jest równość wariancji w dwóch badanych populacjach. Aby statystyka zerowa pochodziła z rozkładu F , musi być spełnione założenie o normalności obu populacji.

Statystyka testowa F-Snedecora ma postać przedstawioną we wzorze 10.6, gdzie poprzez S_X^2 oznaczono wariancję z pierwszej próby o liczności n . Analogicznie S_Y^2 oznacza wariancję z drugiej próby, której liczność wynosi m .

$$F = \frac{S_X^2}{S_Y^2} \quad (10.6)$$

Tak skonstruowana statystyka ma rozkład F o $n - 1$ i $m - 1$ stopniach swobody. Obszar krytyczny dla tego testu jest dwustronny.

10.3.3. Test t-Studenta

Test t-Studenta został stworzony w 1908 roku przez Williama Gosseta. Co ciekawe, pracował on wtedy jako chemik dla browaru Guiness. To właśnie zakaz pracodawcy dotyczący opublikowania owoców pracy Gosseta sprawił, że artykuł został wydrukowany pod pseudonimem „Student” [43]. Litera T występuje w nazwie ze względu na spopularyzowanie statystyki Gosseta przez Fishera, który oznaczał ją właśnie literą T.

Test T jest jedną z najczęściej stosowanych metod statystycznych w celu porównania średnich z dwóch niezależnych od siebie grup. Hipoteza zerowa może być sformułowana jako

równość średnich pochodzących z dwóch populacji o rozkładzie normalnym i identycznych wariancjach. W tym przypadku zbiór krytyczny jest dwustronny. W niniejszej pracy testowana jest hipoteza o większej wartości średniej pochodzącej z próby drugiej X_2 w porównaniu ze średnią z grupy pierwszej X_1 . Przy tak przyjętej hipotezie zbiór krytyczny jest lewostronny.

Postać statystyki testowej przedstawiona jest poniżej (wzór 10.7). W niniejszym wzorze symbolami n_1 i n_2 oznaczono odpowiednio licznosć pierwszego i drugiego zbioru. Poprzez S_1^1 oraz S_2^2 wyrażono ich wariancje.

$$t = \frac{X_1 - X_2}{\sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2} \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}} \quad (10.7)$$

Statystyka testowa ma rozkład t-Studenta o liczbie stopni swobody równej sumie licznosci obu zbiorów pomniejszonej o 2. W przypadku lewostronnego zbioru krytycznego wartosc statystyki testowej poniżej wartosci krytycznej daje postawy do odrzucenia hipotezy zerowej.

Rozdział 11

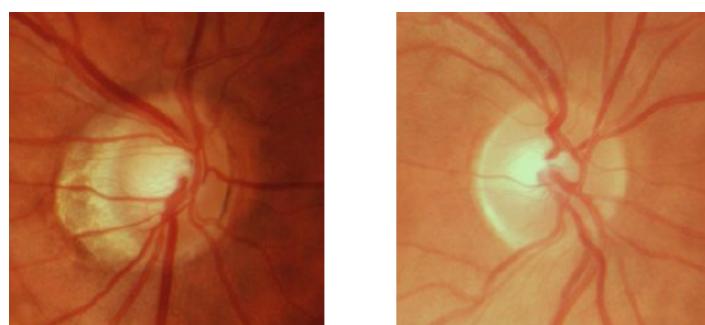
Część eksperimentalna

11.1. Dane eksperimentalne

U. A. ROMANIUK

11.1.1. Obrazy medyczne

Zdjęcia dna oka zostały wykonane i udostępnione przez Samodzielny Publiczny Kliniczny Szpital Okulistyczny w Warszawie i Centrum Mikrochirurgii Oka „Laser”. Obrazy zostały wykonane za pomocą funduskamer textitNidek RS 330-Retina Scan Duo, *DRS*, *Crystavue NFC-700*. Początkowo obrazy te były wykonane w formacie „.png” z rozdzielczością 4096×3072 . Dane medyczne zostały przypisane do jednej z dwóch klas niezależnie przez dwóch doświadczonych lekarzy okulistów. Otrzymany zbiór danych zawierał 283 zdjęcia oczu o prawidłowej morfologii tarczy nerwu wzrokowego oraz 418 zdjęć oczu o nieprawidłowym wyglądzie tej struktury. Przyjęto obrazy tak, aby widoczna była jedynie tarcza nerwu wzrokowego wraz z niewielkim marginesem. Rozmiar otrzymywanych zdjęć wynosił 256×256 pikseli.



Rysunek 11.1: Zdjęcia dna oka. Po lewej zdjęcie oka z nieprawidłową morfologią tarczy nerwu wzrokowego, a po prawej z prawidłową morfologią tej struktury.

11.1.2. Środowisko programowania

Wszystkie programy zostały napisane w języku Python 2.7. Do obsługi sieci użyto biblioteki caffe [44] udostępnionej na licencji BSD (ang. *Berkeley Software Distribution*).

11.2. Eksperyment przy użyciu caffe

U. A. ROMANIUK

Celem wykonywanego eksperymentu było stworzenie konwolucyjnej sieci neuronowej dającej jak najlepsze oceny klasyfikatora. Aby to osiągnąć, eksperyment podzielono na dwie części: *transfer learning* oraz *data augmentation*.

W części poświęconej *transfer learningowi* skupiono się na poszukiwaniu modelu oraz parametrów uczenia sieci neuronowej, dla których stworzony klasyfikator zwróciłby satysfakcyjne wyniki.

W drugiej części eksperymentu, która polegała na sztucznym zwiększaniu zbioru treningowego, poszukiwano przekształceń obrazu poprawiających wyniki klasyfikacji.

11.2.1. Konwolucyjna sieć neuronowa

Do stworzenia klasyfikatora użyto sieci AlexNet udostępnionej na licencji MIT (ang. *Massachusetts Institute of Technology*) [25]. Wybrano ją ze względu na powszechność jej występowania w zagadnieniach związanych z obrazami medycznymi [45]. Sieć zawiera pięć warstw konwolucyjnych i trzy warstwy w pełni połączone. Wejście sieci ma wymiar $227 \times 227 \times 3$. Warstwa wyjściowa zawiera 1000 neuronów, co odpowiada ilości klas zwracanych przez sieć.

11.2.2. Preprocessing

Wszystkie zdjęcia, przed przetwarzaniem przez sieć, zostały przeskalowane do wymiaru 227×227 (z niezmienioną głębokością obrazu).

W części eksperymentu poświęconej sztucznemu powiększaniu zbioru uczącego zastosowano walidację krzyżową. W tym celu stworzono 16 modeli z losowo przyporządkowanymi obrazami do grup testowych i treningowych. Na każdym z nich nauczono sieć o takich samych parametrach. Takie podejście pozwala na zastosowanie testów statystycznych stwierdzających istotność różnic pomiędzy wynikami klasyfikacji.

11.2.3. Transfer learning

K. A. FILIPIUK

Stworzenie klasyfikatora opierającego się na wiedzy innej sieci neuronowej wymaga dokładnego dopasowania zarówno architektury, jak i parametrów opisujących proces douczenia. Niestety, nie istnieje heurystyka pozwalająca stwierdzić, które parametry należy zmienić i w jaki sposób, aby uczenie doprowadziło do osiągnięcia optymalnych wartości wag. Z tego powodu podczas przeprowadzania eksperymentu stworzono ponad 30 modeli klasyfikatorów. Ze względu na bardzo długi czas uczenia, nie było możliwości poświęcenia zasobów na wykonanie walidacji krzyżowej dla każdego modelu. Jest to powód, dla którego w tej części pracy, oceny jakości klasyfikatora nie są podawane wraz z błędami.

Podczas każdego uczenia sieci neuronowej, również w czasie *transfer learningu* należy przede wszystkim zdefiniować podstawowy współczynnik uczenia (*base_lr*). W trakcie treningu sieci AlexNet wynosił on 0,01. W praktyce *transfer learningu* zmniejsza się początkowa szybkość uczenia sieci, aby częściowo zachować postać już wyuczonych filtrów.

Przy uczeniu sieci neuronowej stosuje się zmniejszanie współczynnika uczenia wraz ze zbliżaniem się do poszukiwanego minimum funkcji kosztu. W *transfer learningu* zmniejszanie to należy wykonywać z większą częstotliwością ze względu na zaawansowany poziom wytrenowania sieci. Częstotliwość ta przekazywana jest w postaci parametru o nazwie *stepsize*.

Ze względu na chęć modyfikacji wag warstw w różnym stopniu definiuje się mnożnik współczynnika uczenia (*lr_mult*) dla każdej warstwy. Pozwala on na przyspieszenie, zahamowanie lub uniemożliwienie zmian wartości wag w wybranej warstwie.

Przedstawienie wyników wszystkich stworzonych modeli w niniejszej pracy byłoby bezcelowe. Zaprezentowane zostaną 4 wybrane modele, które wyczerpująco opisują wykonaną przez eksperymentatorów pracę.

Modele te zostały nauczone na zbiorze testowym zawierającym 292 obrazy osób zdrowych, 198 obrazów osób chorych oraz 94 kopie losowych obrazów osób chorych, w celu wyrównania liczebności klas. Po zsumowaniu daje to 584 zdjęcia treningowe. Wyrównywanie grup nie jest stosowane do zbioru walidacyjnego, który składał się z 126 zdjęć dna oka bez zmian morfologicznych oraz 85 zdjęć dna oka z podejrzeniem jaskry, co daje 211 obrazów walidacyjnych.

Model 1 - uczenie ostatniej warstwy

Ideą pierwszego modelu była jak najmniejsza ingerencja w architekturę sieci AlexNet. Jedyną jej zmianą było utworzenie nowej, w pełni połączonej warstwy, w miejsce ostatniej warstwy. Jest to konieczne ze względu na potrzebę dopasowania wyjścia klasyfikatora do nowego zbioru danych.

W procesie uczenia tego modelu zmieniano jedynie losowo zainicjalizowane wagi podmiennej warstwy. Wartości wag na pozostałych warstwach były identyczne z wagami pierwotnego modelu. Prędkość uczenia douczanej warstwy, regulowana poprzez podstawowy współczynnik uczenia oraz jego mnożnik, była identyczna z parametrami oryginalnego AlexNetu i wynosiła 0,01 (jest to wynik mnożenia bazowego współczynnika uczenia oraz jego mnożnika dla tej warstwy). Tabela 11.1 przedstawia parametry przekazywane podczas uczenia sieci.

Model 1	
<i>base_lr</i>	= 0,001
<i>stepsize</i>	= 50 000
warstwa	<i>lr_mult</i>
conv1	0
conv2	0
conv3	0
conv4	0
conv5	0
fc6	0
fc7	0
fc8_kiu	10

Tabela 11.1: Parametry uczenia przekazywane podczas trenowania modelu 1.

Model 2 - uczenie 5 ostatnich warstw

Kolejnym krokiem w poszukiwaniach najlepszego klasyfikatora było pozwolenie na zmianę wag warstw wyższego rzędu. W oryginalnym modelu warstwy te poszukiwały cech obrazów

charakterystycznych dla pierwotnych klas, które między innymi stanowiły koty, psy czy parasy. Badany był postulat, że dopasowanie filtrów warstw konwolucyjnych oraz wag warstw w pełni połączonych do posiadanej zbioru danych spowoduje poprawienie działania klasyfikatora.

Aby osiągnąć wyżej opisane założenia, zmieniono mnożnik współczynnika uczenia dla pięciu ostatnich warstw. Począwszy od czwartej warstwy konwolucyjnej, a kończąc na siódmej warstwie sieci, zmieniono mnożnik współczynnika uczenia na 1. Mnożnik ten dla ostatniej warstwy sieci wynosił 10 (podsumowanie tych wartości znajduje się w tabeli 11.2). Różne wartości mnożników wynikają z faktu, że warstwy, w których ma on wartość jednostkową, inicjalizowane są wagami pochodząymi z AlexNetu, a ich drastyczna zmiana nie jest pożądana. Przed etapem uczenia, wagi ostatniej warstwy mają losowe wartości, stąd też większy mnożnik zapewniający szybsze uczenie.

Model 2	
<i>base_lr</i>	= 0,000 01
<i>stepsize</i>	= 50 000
warstwa	<i>lr_mult</i>
conv1	0
conv2	0
conv3	0
conv4	1
conv5	1
fc6	1
fc7	1
fc8_kiu	10

Tabela 11.2: Parametry uczenia przekazywane podczas trenowania modelu 2.

Model 3 - uczenie 5 ostatnich warstw z różną prędkością

Model 3 jest rozszerzeniem poprzedniego modelu. Mając na uwadze, że poziom skomplikowania wyszukiwanych cech zwiększa się wraz z numerem warstwy, postanowiono przetestować coraz silniejszą zmianę wag na kolejnych warstwach. Parametry uczenia realizujące tę ideę przedstawione są w tabeli 11.3.

Model 4 - dodatkowa warstwa sieci

W modelu 4 postanowiono sprawdzić, jak zmienią się wyniki sieci przy zastąpieniu ostatniej warstwy AlexNetu dwoma warstwami w pełni połączonymi. W tym przypadku nowo powstała ósma warstwa sieci posiada 64 neurony, a dodatkowa warstwa dziewiąta zawiera 2 neurony. Wartości parametrów uczenia tego modelu przedstawione są w tabeli 11.4.

Wyniki

Dla każdego z wyżej wymienionych modeli wyznaczono cztery miary oceny klasyfikatora opisane w rozdziale 10.1. Wartości czułości, dokładności, precyzji i specyficzności przedstawione zostały w tabeli 11.5.

Najczęściej przytaczaną miarą jest dokładność, jednakże w przypadku czterech zaprezentowanych modeli miara ta nie różnicuje ich między sobą w wystarczającym stopniu. Jedynie

Model 3	
<i>base_lr</i>	= 0,000 01
<i>stepsize</i>	= 50 000
warstwa	<i>lr_mult</i>
conv1	0
conv2	0
conv3	0
conv4	1
conv5	1
fc6	2
fc7	3
fc8_kiu	10

Tabela 11.3: Parametry uczenia przekazywane podczas trenowania modelu 3.

Model 4	
<i>base_lr</i>	= 0,000 005
<i>stepsize</i>	= 50 000
warstwa	<i>lr_mult</i>
conv1	0
conv2	0
conv3	0
conv4	0
conv5	0
fc6	0
fc7	0
fc8_kiu	10
fc9_kiu	10

Tabela 11.4: Parametry uczenia przekazywane podczas trenowania modelu 4.

	czułość	dokładność	precyzja	specyficzność
Model 1	0,89	0,64	0,56	0,43
Model 2	0,96	0,68	0,62	0,40
Model 3	0,92	0,71	0,65	0,50
Model 4	0,81	0,65	0,55	0,55

Tabela 11.5: Zestawienie wyników modeli.

model 4 cechuje się równowagą pomiędzy wartością czułości i specyficzności. Niższa wartość precyzji dla tego modelu nie jest alarmująca, gdy pod uwagę weźmie się fakt, iż klasyfikator ten tworzony jest na potrzeby badań przesiewowych. Oznacza to, że wartość czułości, która określa odsetek poprawnie wykrytych osób chorych, jest bardziej istotna niż precyzja. Precyzja mówi o procencie osób faktycznie chorych z grupy wszystkich osób zaklasyfikowanych pozytywnie.

Z wyżej wymienionych powodów do późniejszych obliczeń zastosowano model 4.

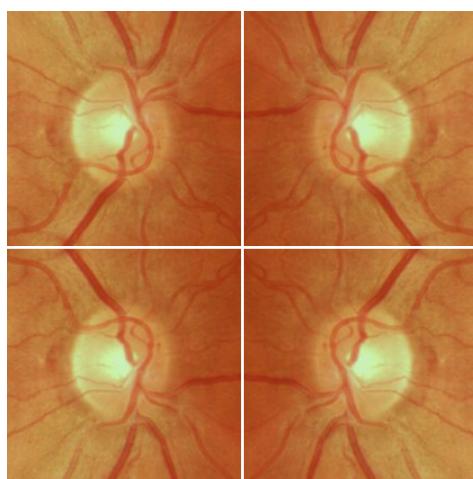
11.2.4. Sztuczne powiększanie zbioru uczącego

U . A . ROMANIUK

Do zwiększenia skuteczności sieć zastosowano sztuczne powiększanie danych uczących. Metoda ta polega na wytworzeniu kilku nowych obrazów z jednego oryginalnego. Zastosowano trzy różne podejścia.

Odbicia wzdłuż osi pionowej i poziomej

Stworzono 4 obrazy za pomocą odbić wzdłuż osi wertykalnej i horyzontalnej. Dla tej przemiany przyjęto nazwę odbicia. Na rysunku 11.2 pokazano opisaną transformację.



Rysunek 11.2: Przykładowe zdjęcie dna oka z odbiciami wzdłuż osi pionowej i poziomej.

Wycięcie 5 obrazów z jednego

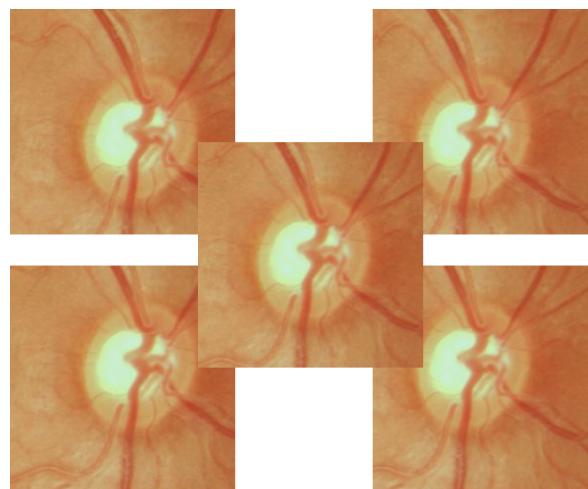
Zwiększo ilość obrazów uczących przy użyciu funkcji wycinającej. Dzięki temu z jednego obrazka otrzymano pięć obszarów, rogi i środek obrazu. W dalszej części pracy będzie używany skrót x5. Na rysunku 11.3 przedstawiono wycięcia obrazu.

Wycięcie 9 obrazów z jednego

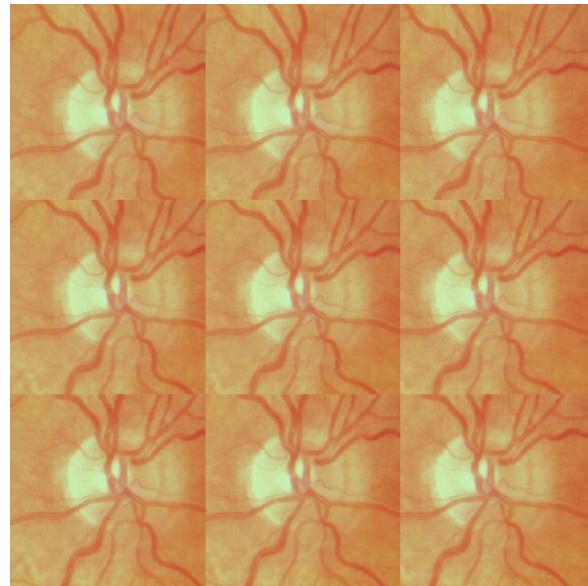
Zastosowano podobną metodologię z poprzedniego punktu do wycięcia 9 obrazów z jednego oryginalnego obrazka. Obszary te były równomiernie rozmiędzzone. Transformacja przedstawiona została na rysunku 11.4. Dla tego przekształcenia przyjęto nazwę x9.

Wyniki

Analogicznie jak w części poświęconej *transfer learningowi*, zbiór uczący zawierał 584 obrazy, a zbiór testowy - 211. Wyniki miar klasyfikatora obliczono dla danych bez powiększenia, odbić, x5 oraz x9. Uśrednione wyniki wraz z odchyleniem standardowym umieszczone w tabeli 11.6.



Rysunek 11.3: Przykład zastosowania wycinania 5 obszarów pochodzących z losowego zdjęcia dna oka.



Rysunek 11.4: Przykład zastosowania wycinania 9 obrazów z jednego losowego zdjęcia dna oka.

	czułość	dokładność	precyzja	specyficzność
brak powiększenia	$0,805 \pm 0,043$	$0,650 \pm 0,031$	$0,546 \pm 0,029$	$0,546 \pm 0,058$
odbicia	$0,713 \pm 0,064$	$0,662 \pm 0,034$	$0,567 \pm 0,037$	$0,627 \pm 0,084$
x5	$0,715 \pm 0,054$	$0,709 \pm 0,028$	$0,621 \pm 0,034$	$0,70 \pm 0,039$
x9	$0,760 \pm 0,054$	$0,690 \pm 0,044$	$0,595 \pm 0,043$	$0,642 \pm 0,084$

Tabela 11.6: Wartości miar oceny klasyfikacji zdjęć dna oka dla różnych sposobów sztucznego powiększania zbioru uczącego przy użyciu caffe.

Wyniki – powiększenie zbioru danych

Ze względu na niesatysfakcjonujące wyniki klasyfikatora podjęto próbę ich poprawy, zwiększając zbiór danych (wykonując dodatkowe zdjęcia), co w teorii powinno poprawić generalizację.

cję. Ostatecznie powstały 933 zdjęcia osób z prawidłową morfologią tarczy nerwu wzrokowego i 754 zdjęcia o nieprawidłowej morfologii tej struktury.

Metodologia powstania sztucznych obrazów została zachowana.

Zbiór uczący zawierał 1586 obrazów, a zbiór testowy zawierał 393 obrazy. Uśrednione wyniki wraz z odchyleniem standardowym umieszczone w tabeli 11.7.

	czułość	dokładność	precyzja	specyficzność
brak powiększenia	$0,820 \pm 0,040$	$0,673 \pm 0,032$	$0,599 \pm 0,030$	$0,550 \pm 0,052$
odbicia	$0,865 \pm 0,028$	$0,697 \pm 0,031$	$0,618 \pm 0,034$	$0,561 \pm 0,057$
x5	$0,804 \pm 0,029$	$0,717 \pm 0,035$	$0,651 \pm 0,043$	$0,646 \pm 0,059$
x9	$0,819 \pm 0,027$	$0,715 \pm 0,033$	$0,646 \pm 0,041$	$0,631 \pm 0,059$

Tabela 11.7: Wartości miar oceny klasyfikacji zdjęć dna oka dla różnych sposobów sztucznego powiększania zbioru uczącego przy użyciu caffe dla powiększonego zbioru danych.

Testy statystyczne

Poziomy istotności wszystkich testów stosowanych w tej pracy wynoszą 5%. Wykonano testy Shapiro-Wilka na normalność rozkładu. Następnie zbadano czy wariancje w obu próbach są równe. Zastosowano test t-Studenta, aby sprawdzić, czy sztuczne zwiększenie zbioru uczącego poprawiło skuteczność klasyfikatora. W przypadku testu F-Snedecora, jeśli wartość znajduje się w zbiorze krytycznym, to wartość zaznaczona jest na czerwono. Dla testu t-Studenta zielono zaznaczono poprawę klasyfikatora, a na czerwono pogorszenie skuteczności sieci. Wyniki znajdują się w tabelach 11.8, 11.10, 11.12 dla pierwszego zbioru i w tabelach 11.9, 11.11, 11.13 dla powiększonego zbioru danych.

	czułość	dokładność	precyzja	specyficzność
brak powiększenia	0,083	0,971	0,559	0,944
odbicia	0,955	0,380	0,869	0,241
x5	0,359	0,501	0,522	0,294
x9	0,973	0,890	0,217	0,477

Tabela 11.8: P-wartości testu Shapiro-Wilka dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w caffe. Przyjmowany poziom istotności testu: $p = 0,050$.

	czułość	dokładność	precyzja	specyficzność
brak powiększenia	0,091	0,548	0,571	0,406
odbicia	0,172	0,061	0,052	0,068
x5	0,888	0,400	0,060	0,055
x9	0,375	0,127	0,065	0,058

Tabela 11.9: P-wartości testu Shapiro-Wilka dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w caffe dla powiększonego zbioru danych. Przyjmowany poziom istotności testu: $p = 0,050$.

Test brak powiększenia z:	czułość	dokładność	precyzja	specyficzność
odbicia	2,2	1,2	1,6	2,1
x5	1,6	1,2	1,4	2,2
x9	1,6	2,0	2,2	2,1

Tabela 11.10: Wartości statystyki testowej F-Snedecora dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w caffe. Zbiór krytyczny: $\langle 2, 4, \infty \rangle$. Przyjmowany poziom istotności testu: $p = 0,050$.

Test brak powiększenia z:	czułość	dokładność	precyzja	specyficzność
odbicia	2,0	1,1	1,3	1,2
x5	1,9	1,2	2,1	1,3
x9	2,2	1,1	1,9	1,3

Tabela 11.11: Wartości statystyki testowej F-Snedecora dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w caffe dla powiększonego zbioru danych. Zbiór krytyczny: $\langle 2, 4, \infty \rangle$. Przyjmowany poziom istotności testu: $p = 0,050$.

Test brak powiększenia z:	czułość	dokładność	precyzja	specyficzność
odbicia	-4,77	1,04	1,79	3,17
x5	-5,22	5,65	6,71	9,04
x9	-2,61	2,97	3,78	3,76

Tabela 11.12: Wartości statystyki testowej t-Studenta dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w caffe. Zbiór krytyczny dla hipotezy alternatywnej w postaci poprawienia wyników: $\langle 1, 7, \infty \rangle$. Na zielono zaznaczono wartości w nim zawierające się. Zbiór krytyczny dla hipotezy alternatywnej w postaci pogorszenia wyników: $(\infty, -1, 7)$. Na czerwono zaznaczono wartości w nim zawierające się. Przyjmowany poziom istotności testu: $p = 0,050$.

Test brak powiększenia z:	czułość	dokładność	precyzja	specyficzność
odbicia	3,69	2,15	1,68	0,57
x5	-1,30	3,71	3,97	4,88
x9	-0,08	3,65	3,70	4,12

Tabela 11.13: Wartości statystyki testowej t-Studenta dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w caffe dla powiększonego zbioru danych. Zbiór krytyczny dla hipotezy alternatywnej w postaci poprawienia wyników: $\langle 1, 7, \infty \rangle$. Na zielono zaznaczono wartości w nim zawierające się. Zbiór krytyczny dla hipotezy alternatywnej w postaci pogorszenia wyników: $(\infty, -1, 7)$. Na czerwono zaznaczono wartości w nim zawierające się. Przyjmowany poziom istotności testu: $p = 0,050$.

Porównanie zbiorów o różnej liczebności

Zwiększo zbiór uczący ze względu na niesatysfakcyjujące wyniki klasyfikatora uzyskane w pierwszej części eksperymentu. Sprawdzono, czy statystycznie wyniki uzyskane dla większe-

go zbioru danych (tabela 11.6) są rzeczywiście lepsze od wyników uzyskanych na mniejszym zbiorze (tabela 11.7). Zbadano równość wariancji w obu próbach (tabela 11.14). Sprawdzono statystyczną poprawę klasyfikatorów testem t-Studenta (tabela 11.15).

	czułość	dokładność	precyzja	specyficzność
brak powiększenia	1,2	1,1	1,1	1,2
odbicia	12,3	1,2	1,2	2,2
x5	3,5	1,6	1,6	2,3
x9	4,0	1,8	1,8	2,0

Tabela 11.14: Wartości statystyki testowej F-Snedecora dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w caffe. Zbiór krytyczny: $\langle 2, 4, \infty \rangle$. Na czerwono zaznaczono wartości znajdujące się w zbiorze krytycznym. Przyjmowany poziom istotności testu: $p = 0,050$.

	czułość	dokładność	precyzja	specyficzność
brak powiększenia	1,02	2,06	5,08	0,21
odbicia	6,96	3,04	4,06	-2,60
x5	5,81	0,71	2,19	-3,28
x9	3,91	1,82	2,97	-0,43

Tabela 11.15: Wartości statystyki testowej t-Studenta dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w caffe dla powiększonego zbioru danych. Zbiór krytyczny dla hipotezy alternatywnej w postaci poprawienia wyników: $\langle 1, 7, \infty \rangle$. Na zielono zaznaczono wartości w nim zawierające się. Zbiór krytyczny dla hipotezy alternatywnej w postaci pogorszenia wyników: $\langle \infty, -1, 7 \rangle$. Na czerwono zaznaczono wartości w nim zawierające się. Przyjmowany poziom istotności testu: $p = 0,050$.

11.2.5. Podsumowanie

I. M. SZOPA

W pierwszej części eksperymentu zastosowano *transfer learning* do otrzymanego zbioru danych. Testowano zmianę różnych parametrów oraz przekształcenia warstw w sieci. Metodą prób i błędów zmieniano po kolejne każdy z parametrów, który uważano, że może mieć znaczenie w poprawieniu klasyfikacji sieci. Najbardziej istotnym czynnikiem w momencie trenowania dużej ilości modeli sieci neuronowych jest czas. Liczenie jednego modelu może zająć nawet kilka dni w zależności od posiadanej sprzęt. Po otrzymaniu satysfakcyjnych wyników klasyfikatora zakończono poszukiwanie zmian w parametrach architektury sieci AlexNet.

W drugiej części eksperymentu skupiono się na dalszej poprawie otrzymanych już wyników klasyfikacji sieci. Wykorzystano do tego sztuczne zwiększenie zbioru uczącego. Przetestowano trzy różne przekształcenia obrazu wejściowego, odbicie wzduż osi pionowej i poziomej, wycięcie pięciu oraz dziewięciu równej rozmiarów obrazów z bazowego obrazu. Zastosowanie tego podejścia we wszystkich przypadkach poprawiło wynik klasyfikatora.

Spośród wszystkich wyników otrzymanej w tej części eksperymentu wybrano model 4 (opisany w rozdziale 11.2.3), do którego uczenia zastosowano metodę wycięcia 9 obrazów z jednego (rozdział 11.2.4).

11.3. Eksperyment przy użyciu fastai

K. A. FILIPIUK

Fast.ai jest instytutem badawczym założonym w 2015 roku przez Jeremy'ego Howarda i Rachel Thomas. Idea przyświecającą instytutowi jest popularyzacja sieci neuronowych, która ma za zadanie ukrócić dominację niewielkiej liczby najbogatszych korporacji w dziedzinie uczenia maszynowego [46].

W październiku 2018 roku Fast.ai udostępniło narzędzie fastai do tworzenia sieci neuronowych na licencji Apache License 2.0. Oprogramowanie fastai stworzone zostało w interpretowanym języku programowania Python. Oprogramowanie to jest w rzeczywistości biblioteką, czyli zbiorem metod rozszerzających podstawową użyteczność. Fastai korzysta z biblioteki PyTorch, wspieranej przez firmy technologiczne takie jak Facebook oraz Uber, która została udostępniona w 2016 roku udostępniona na licencji BSD [47].

Najistotniejszymi funkcjami uwzględnionymi w bibliotece fastai są między innymi automatyzacja wytwarzania sztucznych obrazów uczących oraz implementacja algorytmu cyklicznego współczynnika uczenia.

Algorytm cyklicznego współczynnika uczenia jest stosunkowo nową metodą, zaproponowaną w kwietniu 2018 roku przez Leslie Smitha [48]. Ideą przyświecającą metodzie Smitha jest rozpoczęcie cyklu uczenia przy bardzo małej wartości współczynnika uczenia. Wraz z kolejnymi iteracjami współczynnik ten rośnie aż do zdefiniowanej maksymalnej wartości, aby potem ponownie go zmniejszać. Okazuje się, że takie podejście nie tylko umożliwia sieci lepszą generalizację, ze względu na większe prawdopodobieństwo odnalezienia globalnego minimum funkcji kosztu, ale także na zwiększoną szybkość trenowania sieci, która jest efektem stosowania wysokiego współczynnika uczenia.

W niniejszej części pracy stosowana jest wyłącznie metoda cyklicznego współczynnika uczenia. Pozwala ona na osiągnięcie minimum funkcji kosztu sieci konwolucyjnej już w kilku epokach uczenia.

Poniżej zostanie przedstawiony i omówiony szkic kodu programu napisanego przy użyciu biblioteki fastai, którego celem jest stworzenie klasyfikatora w postaci głębokiej sieci konwolucyjnej.

11.3.1. Tworzenie klasyfikatora obrazów przy użyciu fastai

Uczenie sieci neuronowych jest kosztowne obliczeniowo. Komputery stacjonarne bądź laptopy nie są przystosowane do tego typu zadań. Z tego względu planując stworzenie klasyfikatora opartego na sieci neuronowej, którego przykładem jest konwolucyjna sieć neuronowa, należy rozważyć skorzystanie z innych, silniejszych jednostek obliczeniowych.

Optymalnymi jednostkami do wykonywania obliczeń przy implementacji algorytmów uczenia maszynowego są karty graficzne. Pomimo szybszego wykonywania obliczeń przez procesor, jednostki graficzne wykonują wiele obliczeń jednocześnie, co przekłada się na krótszy czas uczenia klasyfikatorów. Ze względu na powszechnie zastosowanie sieci neuronowych w wielu dziedzinach (sugestie produktów w sklepach internetowych, rozpoznanie osób na zdjęciach w sieciach społecznościowych), powstały jednostki obliczeniowe (będące w teorii kartami graficznymi) dedykowane uczeniu maszynowemu.

Na rynku dostępnych jest wiele odpłatnych serwisów, oferujących usługę wykorzystywania serwerów wyposażonych w karty graficzne do zastosowań uczenia maszynowego. Dwa największe z nich – Google Cloud Platform i Azure – oferują pełne wsparcie dla fastai. W praktyce

oznacza to prostotę korzystania z tego rozwiązania, ponieważ wszystkie niezbędne komponenty, są preinstalowane na serwerze.

Posiadając zasoby w postaci serwera z kartą graficzną, należy zastanowić się nad wyborem środowiska graficznego do programowania. Jego posiadanie nie jest konieczne, jednakże pisanie kodu w graficznym edytorze jest znaczącym ułatwieniem, w szczególności dla niedoswiadczonego programisty. Ze względu na interpretowalność języka programowania Python, zalecanym wyborem edytora jest Jupyter. Jest to interaktywne środowisko działające w przeglądarce internetowej, stworzone do wykonywania oraz wizualizacji działania kodu pythonowego. Jego główną zaletą jest możliwość wykonywania skryptu linijka po linijce, ilustrując działanie programu, pomiędzy kolejnymi wywołaniami.

Deklaracja używanych bibliotek

Jedyną biblioteką, którą należy zainportować w celu korzystania z modułów stworzonych przez omawiany w tym rozdziale instytut badawczy, jest fastai. Nie jest wymagany import bibliotek, od których fastai zależy. Zostanie to wykonane niejawnie.

W poniższym przykładzie funkcje dostępne w omawianej bibliotece są bezpośrednio importowane do przestrzeni nazw. Standard programowania stanowczo odradza stosowanie takiej praktyki, ze względu na możliwe występowanie konfliktów. Jednakże ze względu na ideę jak największego uproszczenia pracy z kodem, Howard i Thomas zwróciли znaczną uwagę na uniknięcie takich kolizji.

```
from fastai import *
from fastai.vision import *
```

Tworzenie zbioru danych

W celu stworzenia klasyfikatora danych niezbędne jest posiadanie zbioru zdjęć, wraz z ich poprawnym przypisaniem do klas. Istnieją różne metody przekazania tych informacji do programu. Najczęściej spotykanym sposobem na kodowanie klas zdjęć jest przechowywanie ich w katalogach o nazwie odpowiadającej nazwie grupy. Przykład takiej hierarchii obrazów jest przedstawiony poniżej.

```
katalog_nadrzedny
    train
        klasa1
        klasa2
        klasa3
        ...
        klasaN
    valid
        klasa1
        klasa2
        klasa3
        ...
        klasaN
```

W tym przykładzie w katalogu nadzrzednym znajdują się dwa katalogi: train i valid. W każdym z tych folderów znajduje się taka sama liczba podfolderów, których nazwy odpowiadają nazwom klas.

Poniżej znajduje się kod, służący wczytaniu tak skatalogowanych obrazów. Pod zmienną `data` powstanie obiekt klasy `ImageDataBunch` zawierający odpowiednio spreparowane dane (to znaczy podzielone na zbiory treningowe i walidacyjne), przekazane przez użytkownika.

```
data = ImageDataBunch . from_folder (
    path='gdzie_ma_utworzyc_dataset',
    train='gdzie_najduja_sie_obrazy_treningowe',
    valid='gdzie_najduja_sie_obrazy_walidacyjne')
```

Sztuczne powiększanie zbioru treningowego

Obiekt `data` zawiera wiele zmiennych, w tym także zmienną `ds_tfms` odpowiedzialną za stosowanie transformacji na zbiorach obrazów. Należy jej przekazać krotkę zawierającą dwie listy transformacji dla zbioru uczącego i walidacyjnego.

Samodzielne tworzenie krotek, celem przekazania ich zmiennej `ds_tfms` byłoby bardzo żmudne. Z tego też względu w bibliotece fastai uwzględniona została funkcja zwracająca wymaganą krotkę. Poniżej przedstawiona jest jej sygnatura.

```
get_transforms( do_flip : bool=True ,
                flip_vert : bool=False ,
                max_rotate : float = 10.0 ,
                max_zoom : float = 1.1 ,
                max_lighting : float = 0.2 ,
                max_warp : float = 0.2 ,
                p_affine : float = 0.75 ,
                p_lighting : float = 0.75 ,
                xtra_tfms : Optional[ Collection[ Transform ] ] = None )
```

Jak zostało to pokazane, wszystkie argumenty tej funkcji mają wartości domyślne. Wartości te nie są przypadkowe. Zostały one dobrane tak, aby poprawiały wyniki sieci przy większości zastosowań.

Reasumując, aby zastosować *data augmentation* do swojego zbioru treningowego, wystarczy poniższa linijka kodu.

```
ds_tfms = get_transforms()
```

Standaryzacja danych

Tak jak wszystkie klasyfikatory, również sieci neuronowe wymagają, aby dane im przekazywane były skupione wokół zera. Standaryzacja wariancji do wartości jednostkowej nie jest tak ścisłym wymogiem, ale niektóre gotowe architektury stosowane w *transfer learningu* tego wymagają.

Przy sieciach już nauczonych na pewnym zbiorze danych, do standaryzacji nowego zbioru uczącego stosuje się zwykle średnią oraz odchylenie standardowe zadane przez zbiór uczący z poprzednio wyuczonego zestawu danych.

Prawie wszystkie powszechnie dostępne modele sieci neuronowych w bibliotece PyTorch pochodzą z przeprowadzanego co roku konkursu ILSVRC (ang. *ImageNet Large Scale Visual Recognition Competition*). Ponieważ wszystkie te modele wytrenowane zostały na tym samym zbiorze danych, wszystkie również wymagają takiej samej normalizacji danych. Wartości średnich i odchylenia standardowego dla zbioru uczącego pochodzącego z przytoczonego

konkursu są udostępnione w Internecie. Jednakże, aby nie wpisywać ich ręcznie, wartości te przechowywane są w module `imagenet_stats`.

Aby wykonać normalizację obrazów za pomocą statystyk zbioru ImageNet, przechowywanych w obiekcie `data`, wystarczy w kodzie uwzględnić poniższą linijkę.

```
data.normalize(imagenet_stats)
```

Tworzenie modelu sieci

Posiadając już obiekt zawierający zbiór treningowy i walidacyjny, po odpowiednich przekształceniach, można stworzyć model sieci. W tym celu należy wybrać odpowiednią architekturę.

Fastai zawiera w sobie wszystkie modele zaimplementowane w bibliotece PyTorch, takie jak AlexNet, VGG, ResNet, DenseNet, SqueezeNet oraz wiele innych. Aby utworzyć klasyfikator opierający się na którejś z tych architektur, należy podać funkcji `cnn_learner` danych, na których trenowana będzie sieć, oraz nazwy wybranego modelu tak jak w poniższym przykładzie. Utworzona sieć zostanie automatycznie dopasowana do przekazanych danych pod względem mocy zbioru klas oraz zainicjowana wagami nauczonymi na zbiorze ImageNet.

```
learner = cnn_learner(data, models.alexnet)
```

Uczenie klasyfikatora

Ostatnią czynnością niezbędną do utworzenia funkcjonalnego klasyfikatora, będącego konwolucyjną siecią neuronową, jest douczenie stworzonej w poprzednim kroku sieci. W tym celu wywołana zostanie metoda obiektu `learner` o nazwie `fit_one_cycle`. Parametrem przekazywanym tej metodzie jest liczba epok do wykonania. Jeżeli wartość współczynnika uczenia nie zostanie podana explicit, zostanie zastosowana wartość domyślna wynosząca 0,003.

```
learner.fit_one_cycle(5)
```

Ocena klasyfikatora

Istnieje wiele miar pozwalających na ocenę klasyfikatora. Jedną z bardziej obrazowych metod jest przedstawienie macierzy błędów. Stosując fastai można ją otrzymać w dwóch poniższych linijkach kodu.

```
interpretation = ClassificationInterpretation.from_learner(learner)
interpretation.plot_confusion_matrix()
```

Predykcja klasy obrazu

Kulminacyjnym momentem tworzenia klasyfikatora jest zastosowanie go w praktyce. Aby otrzymać przewidywaną przez sieć konwolucyjną klasę, należy zastosować metodę `predict` obiektu `learner`, przekazując jej jako parametr obraz. W tym celu obraz zostaje najpierw wczytany za pomocą funkcji `open_image`.

```
img = open_image('sciezka_do_obrazu')
learner.predict(img)
```

Zestawienie kodu

Poniżej przedstawiono w jednym miejscu elementy kodu uprzednio omówione. Dopiero w tej postaci można dostrzec, jak krótki jest kod pozwalający na korzystanie z konwolucyjnych sieci neuronowych. Jego trywialność pozwala na tworzenie znakomitych klasyfikatorów nawet osobom, które nie miały wcześniej doświadczenia z programowaniem i sieciami konwolucyjnymi.

```
from fastai import *
from fastai.vision import *

data = ImageDataBunch.from_folder(
    path='gdzie_ma_utworzyc_dataset',
    train='gdzie_znajduja_sie_obrazy_treningowe',
    valid='gdzie_znajduja_sie_obrazy_walidacyjne',
    ds_tfms=get_transforms()).normalize(imagenet_stats)
learner = cnn_learner(data, models.alexnet)
learner.fit_one_cycle(5)

interpretation = ClassificationInterpretation.from_learner(learner)
interpretation.plot_confusion_matrix()

img = open_image('sciezka_do_obrazu')
learner.predict(img)
```

11.3.2. Specyfikacja techniczna

Poniższa część eksperymentalna wykonana została przy użyciu maszyny wirtualnej utworzonej na serwerach Google Cloud Platform. Maszyna ta miała dostęp do 8 wirtualnych procesorów, opartych na jednostkach Intel Skylake, 52 GB pamięci dyskowej oraz jednej karty graficznej Nvidia Tesla P100.

11.3.3. Charakterystyka danych wejściowych

Do treningu klasyfikatora oraz jego oceny zostało użytych łącznie 1687 obrazów. Liczba zdjęć osób bez cech świadczących o degeneracji nerwu wzrokowego wynosiła 933, a liczba zdjęć osób ze znamionami jaskry – 754.

Wszystkie zdjęcia podzielono losowo na dwa zbiorы: uczący i walidacyjny. W zbiorze uczącym znajdowało się 80% obrazów bez objawów chorobowych oraz 80% obrazów zawierających objawy jaskry. Pozostałe dane zostały przydzielone do zbioru walidacyjnego.

Powyższy podział powtórzono 16 razy. W rezultacie otrzymano 16 zbiorów uczących oraz 16 odpowiadających im zbiorów walidacyjnych. Analogicznie do poprzedniego eksperymentu, czynność tę wykonano w celu wnioskowania statystycznego dotyczącego budowanych klasyfikatorów.

11.3.4. Wybór modelu konwolucyjnej sieci neuronowej

AlexNet jest bardzo często pierwszym wyborem wielu osób stosujących *transfer learning*, ze względu na mało skomplikowaną strukturę. Jego architektura jest jednokierunkowa,

o 5 warstwach konwolucyjnych oraz 3 warstwach w pełni połączonych. Dokładna budowa AlexNetu została omówiona w rozdziale 11.2.1. Pomimo prostoty tej sieci, charakteryzuje się ona dobrymi wynikami pod względem klasyfikacji zbioru ImageNet. Dokładność AlexNetu przy porządkowywania obrazów do jednej z 1000 klas wynosiła 56%. Jednakże, od czasu postania omawianego modelu, powstało znacznie więcej ogólnodostępnych konwolucyjnych sieci neuronowych, które zdeterminizowały AlexNet, jako zwycięzcę ILSVRC z 2012 roku. Ich stosunkowo niską popularność w zastosowaniach *transfer learningu* można tłumaczyć faktem wyższego skomplikowania oraz większej liczby parametrów, co prowadziło do przymusu zgromadzenia większej ilości danych.

Użycie biblioteki fastai zdejmuję z eksperymentatora odpowiedzialność za dostosowanie wybranego modelu sieci. Jak zostało to pokazane w 11.3.1, za utworzenie dopasowanego modelu sieci do posiadanych danych odpowiedzialna jest jedna funkcja. Przekazywany jej jest gotowy model sieci, na którym tworzony klasyfikator ma się opierać. Umożliwia to użytkownikowi na zastosowanie modeli o bardzo złożonych architekturach, których samodzielna zmiana i dopasowywanie mogłoby okazać się zbyt pracochłonne.

Na potrzeby tego eksperymentu zdecydowano się na porównanie trzech modeli sieci konwolucyjnych, jakimi są AlexNet, VGG oraz ResNet. Zarówno VGG, jak i ResNet posiadają kilka implementacji różniących się liczbą warstw, a co za tym idzie – liczbą parametrów. Zwiększenie liczby warstw w tych modelach prowadziło do poprawienia się dokładności klasyfikacji oryginalnego zbioru, na którym były one uczone. Sieć VGG w porównaniu do ResNet zawiera więcej parametrów, co z punktu widzenia prędkości uczenia jest wadą. Niestety w fastai dostępna jest tylko jedna wersja modelu VGG, zawierająca 16 warstw. ResNet jest dostępny w wielu wersjach. W tej pracy skupiono się na wersji 34- i 50-warstwowej.

W tabeli 11.16 znajduje się porównanie czterech wybranych miar oceny klasyfikacji, jakimi są czułość, dokładność, precyzja i specyficzność.

	czułość	dokładność	precyzja	specyficzność
AlexNet	$0,646 \pm 0,043$	$0,726 \pm 0,019$	$0,711 \pm 0,053$	$0,790 \pm 0,048$
VGG16	$0,730 \pm 0,047$	$0,770 \pm 0,023$	$0,746 \pm 0,038$	$0,802 \pm 0,036$
ResNet34	$0,736 \pm 0,039$	$0,767 \pm 0,022$	$0,739 \pm 0,048$	$0,793 \pm 0,039$
ResNet50	$0,749 \pm 0,042$	$0,779 \pm 0,026$	$0,752 \pm 0,041$	$0,803 \pm 0,035$

Tabela 11.16: Wartości miar oceny klasyfikacji zdjęć dna oka dla czterech modeli konwolucyjnych sieci neuronowych stworzonych przy użyciu fastai. Oszacowanie błędu otrzymano w wyniku walidacji krzyżowej.

Dla każdej miary oceny klasyfikatora zbadano testem Shapiro-Wilka hipotezę dotyczącą normalności rozkładu 16 wartości otrzymanych w wyniku walidacji krzyżowej. W tabeli 11.17 przedstawiono p-wartości tego testu. Przyjmując poziom istotności testu na poziomie 5%, nie mamy podstaw do odrzucenia hipotezy zerowej dla wszystkich przedstawianych przypadków, z wyjątkiem rozkładu czułości dla sieci VGG16. Nie zdecydowano się jednak na zastosowanie testu nieparametrycznego, ze względu na jego mniejszą moc. Należy mieć na uwadze, że poziom istotności oznacza prawdopodobieństwo popełnienia błędu typu I, który mówi o odrzuceniu hipotezy zerowej pomimo jej poprawności. Założono, że w tym przypadku niska wartość p-wartości jest efektem właśnie popełnienia tego typu błędu.

W celu wnioskowania statystycznego dotyczącego poprawy czterech wyznaczanych miar przy zmianie modelu sieci należy zastosować test t-Studenta. Ze względu na jego założenia, najpierw zbadano równość wariancji testem F-Snedecora pomiędzy interesującymi modelami. Wartości statystyki testowej tego testu przedstawione są w tabeli 11.18. Prawostronny obszar

	czułość	dokładność	precyzja	specyficzność
AlexNet	0,772	0,118	0,806	0,665
VGG16	0,017	0,779	0,605	0,625
ResNet34	0,215	0,125	0,197	0,115
ResNet50	0,928	0,477	0,825	0,096

Tabela 11.17: P-wartości testu Shapiro-Wilka dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w fastai. Przyjmowany poziom istotności testu: $p = 0,050$. Na czerwono zaznaczono wartości poniżej poziomu istotności.

krytyczny dla poziomu istotności 5% wynosi $\langle 2, 4, \infty \rangle$.

Model 1	Model 2	czułość	dokładność	precyzja	specyficzność
AlexNet	VGG16	1,1	1,2	1,4	1,3
AlexNet	Resnet34	1,1	1,1	1,1	1,2
AlexNet	ResNet50	1,0	1,3	1,3	1,3
VGG16	ResNet34	1,2	1,1	1,3	1,1
VGG16	ResNet50	1,1	1,0	1,1	1,0
Resnet34	ResNet50	1,1	1,2	1,1	1,1

Tabela 11.18: Wartości statystyki testowej F-Snedecora dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w fastai. Zbiór krytyczny: $\langle 2, 4, \infty \rangle$. Przyjmowany poziom istotności testu: $p = 0,050$.

Wiedząc, że są spełnione założenie testu t, wykonano go dla par modeli sieci neuronowych. Badano hipotezę o mniejszej wartości średniej miary oceny pochodzącej z modelu z pierwszej kolumny tabeli w porównaniu do wartości tej średniej obliczonej na podstawie modelu z drugiej kolumny. Wartości statystyki testowej t-Studenta znajdują się w tabeli 11.19. Prawostronny obszar krytyczny dla tej hipotezy na poziomie istotności 5% wynosi $\langle 1, 7, \infty \rangle$.

Model 1	Model 2	czułość	dokładność	precyzja	specyficzność
AlexNet	VGG16	5,3	6,0	2,1	0,8
AlexNet	Resnet34	6,2	5,8	1,5	0,2
AlexNet	ResNet50	6,8	6,7	2,4	0,9
VGG16	ResNet34	0,4	-0,4	-0,5	-0,7
VGG16	ResNet50	1,2	1,0	0,4	0,1
Resnet34	ResNet50	0,9	1,4	0,8	0,8

Tabela 11.19: Wartości statystyki testowej t-Studenta dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w fastai. Zbiór krytyczny dla hipotezy alternatywnej w postaci poprawienia wyników: $\langle 1, 7, \infty \rangle$. Na zielono zaznaczono wartości w nim zawierające się. Przyjmowany poziom istotności testu: $p = 0,050$.

Jak wynika z powyższej tabeli, oba modele ResNetów oraz VGG16 są lepsze od AlexNetu pod względem czułości i dokładności, a różnica ta jest istotna statystycznie. Analizując pary modeli VGG16-ResNet34, VGG16-ResNet50 oraz ResNet34-ResNet50 nie stwierdzono istotnych statystycznie różnic między nimi. Bazując na wartości statystyki, jako model o najlepszych możliwościach klasyfikacyjnych wybrano ResNet50.

11.3.5. Sztuczne powiększanie zbioru uczącego

Sztuczne powiększanie zbioru danych jest powszechnie stosowaną praktyką podczas tworzenia klasyfikatorów. Główną ideą tego podejścia jest stworzenie z jednego obrazu jego kilku wersji, które są tożsame dla zadanego problemu, jednak przez system komputerowy odbierane jako różne zdjęcia. Powiększanie zbioru uczącego szerzej opisane jest w rozdziale 8.

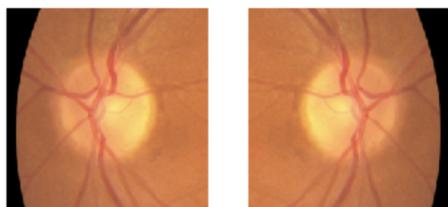
W tym rozdziale zastosowane zostaną różne sposoby powiększania zbioru danych oraz przetestowany zostanie ich wpływ na wybrany w poprzednim rozdziale model klasyfikatora ResNet50. Omówione transformacje różnią się od tych, stosowanych w poprzednim eksperymencie, gdzie przekształcenia były dokonywanie za pomocą własnoręcznie stworzonego kodu. Poniższe sposoby są wewnętrznymi metodami biblioteki fastai.

W czasie klasyfikacji obrazu przez sieć nauczoną na powiększonym zbiorze danych można również stosować transformacje. Stosuje się wtedy uśrednianie wyników otrzymywanych na oryginalnym zdjęciu, jak i również na zdjęciach przetworzonych. W tym eksperymencie zastosowano średnią ważoną, w której wynik pochodzący z niezmienionego obrazu posiadał wagę 0,6. Nazwa tego sposobu nie posiada polskiego odpowiednika, dlatego zostanie zastosowana nazwa angielska *test time augmentation* lub TTA.

Odbicie wzdłuż osi pionowej

Najpopularniejszym sposobem powiększania zbioru uczącego jest stworzenie lustrzanego odbicia wszystkich obrazów wzdłuż osi pionowej. Przed jego zastosowaniem należy jednakże zastanowić się, czy to podejście jest zasadne. Dobrym przykładem, który ujmie, jak niefajne może być to podejście, jest klasyfikator pisma. Tworząc lustrzane odbicia wszystkich liter, powstają twory, które przeszkadzają w poprawnej generalizacji liter.

Na rysunku 11.5 zwizualizowane zostało omawiane przekształcenie. Metoda ta jest analogiczna do metody użytej w pierwszej części eksperimentu (rozdział 11.2.4).

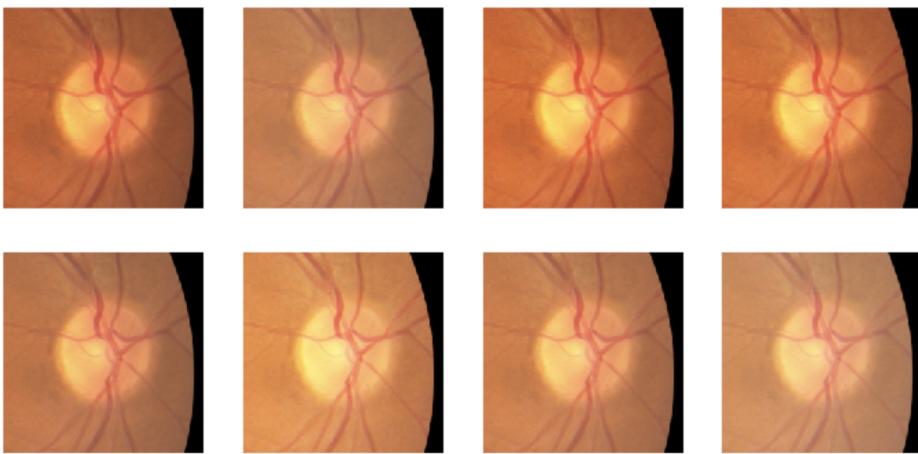


Rysunek 11.5: Przykład zastosowania odbicia wzdłuż osi pionowej dla losowego zdjęcia ze zbioru uczącego.

Dla zbioru danych zawierającego zdjęcie dna oka, stworzenie ich odbicia nie wydaje się błędem. W przypadku zdjęcia oka prawego, przekształcenie go wzdłuż osi pionowej powinno zasymulować zdjęcie oka lewego.

Zmiana jasności i kontrastu

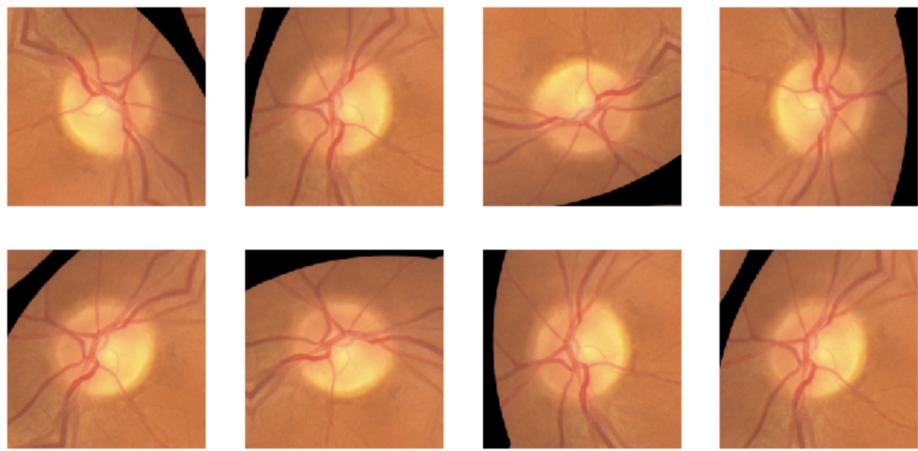
Transformacją, która nie ingeruje w pozycję struktur dna oka na zdjęciu, jest zmiana jasności oraz kontrastu zdjęcia. Dla ludzkiego mózgu omawiane przekształcenie, którego efekty przedstawione są na rysunku 11.6, jest praktycznie niezauważalne, jednakże ze względu na zmianę wartości pikseli, stanowi całkowicie nowy obraz dla systemu komputerowego.



Rysunek 11.6: Przykład zastosowania losowej zmiany jasności i kontrastu dla losowego zdjęcia ze zbioru uczącego.

Losowa rotacja

Następnym przekształceniem konstruującym nowe dane jest rotacja obrazu. W zależności od zagadnienia można zadecydować, o jaką wartość kąta przekrećać zdjęcie. Na rysunku 11.7 przedstawiono działanie programu obracającego wejściowe zdjęcie o losową wartość z przedziału $[0, 360)$.

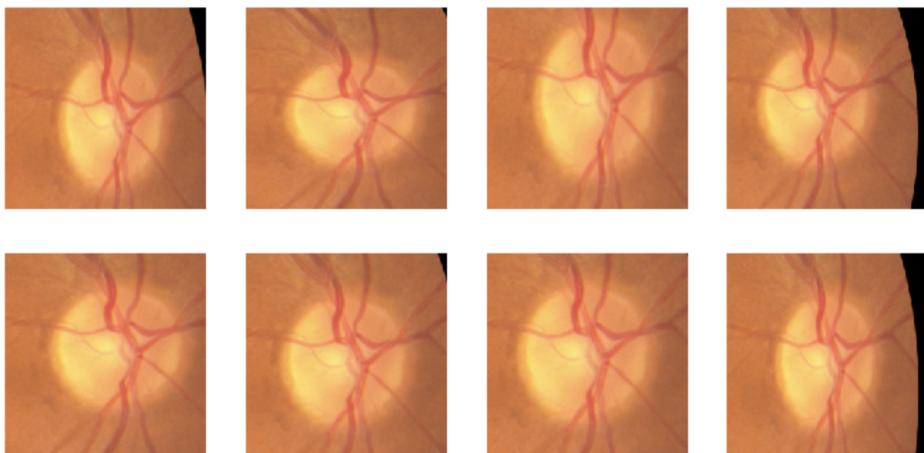


Rysunek 11.7: Przykład zastosowania rotacji o kąt z przedziału $[0, 360)$ dla losowego zdjęcia ze zbioru uczącego.

Znieksztalcenia

Znieksztalcenia danych wejściowych skalują obrazy wzdłuż losowej osi. Oznacza to zmianę kształtów zawartych na obrazie. Nie jest jasne, czy taka transformacja będzie skutecznie poprawiać wyniki klasyfikatora wybranego problemu.

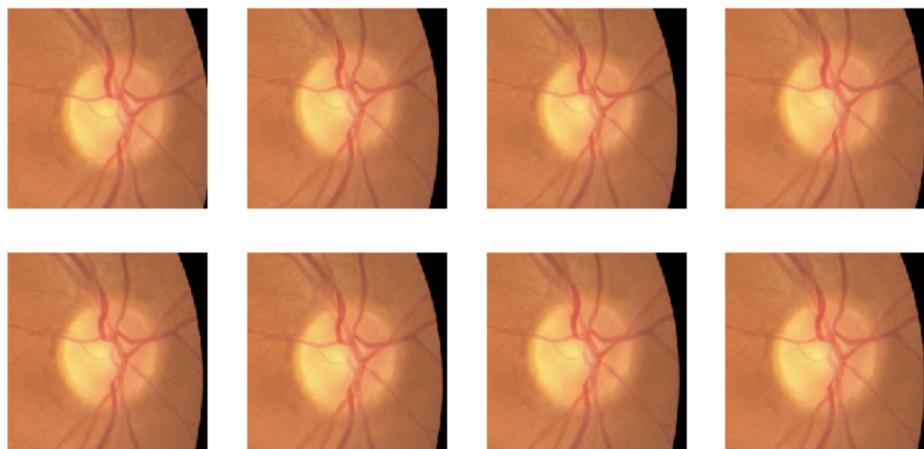
Przykłady omawianego przekształcenia przedstawione są na rysunku 11.8.



Rysunek 11.8: Przykład zastosowania zniekształceń dla losowego zdjęcia ze zbioru uczącego.

Przybliżanie

W tej metodzie zdjęcie było przybliżane o losową wartość pomiędzy 1-krotnym przybliżeniem (będące w rzeczywistości jego brakiem) a 1,1-krotnym zoomem. Tak przetworzone obrazy przedstawione są na rysunku 11.9.



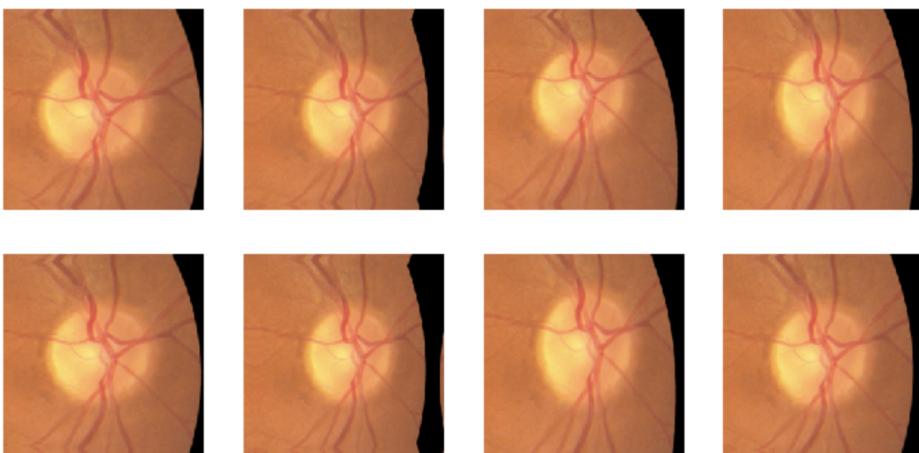
Rysunek 11.9: Przykład zastosowania przybliżeń dla losowego zdjęcia ze zbioru uczącego.

Losowe wycinanie

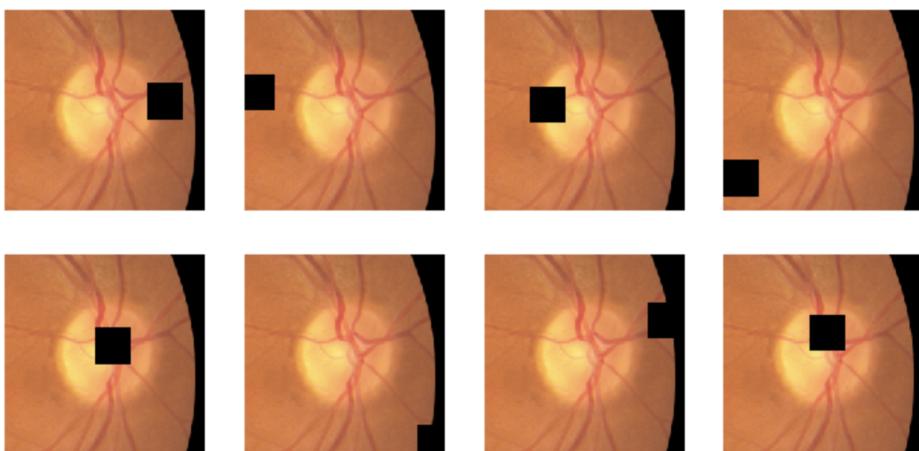
Metodą zwiększającą liczbę obrazów uczących, analogiczną do wycięcia 5 lub 9 obrazów z jednego (rozdział 11.2.4), jest losowy wybór części zdjęcia. Przy stosowanej implementacji transformacji dokładowane jest także skalowanie zdjęcia przez wartości z przedziału [1, 2]. Przykładowe, tak utworzone obrazy widoczne są na rysunku 11.10.

Nakładanie maski

Ostatnią przedstawianą w tej pracy transformacją jest nakładanie czarnego kwadratu w losowym miejscu obrazu. Wizualizacja tej metody przedstawiona na rysunku 11.11



Rysunek 11.10: Przykład zastosowania losowego wycinania dla losowego zdjęcia ze zbioru uczącego.



Rysunek 11.11: Przykład zastosowania nakładania maski dla losowego zdjęcia ze zbioru uczącego.

Wyniki

W tabeli 11.20 przedstawiono wyniki miar oceny klasyfikatorów, opartych na różnych sposobach powiększenia zbioru danych uczących. Podczas klasyfikacji zdjęć stosowano TTA.

Testy statystyczne

Tabele 11.21 i 11.22 przedstawiają odpowiednio p-wartości testu Shapiro-Wilka oraz wartości statystyki testowej F-Snedecora.

Poziomem odniesienia dla wszystkich klasyfikatorów tworzonych na powiększonych zbiorach uczących są wartości miar klasyfikacji modelu ResNet50 wytrenowanego na oryginalnych obrazach.

W tabeli 11.23, zawierającej wartości statystyki testu t-Studenta, zaznaczono kolorem zielonym wartości zawierające się w prawostronnym zbiorze krytycznym. Oznacza to istotną statystycznie poprawę wyników klasyfikatora. Analogicznie, kolorem czerwonym oznaczono wyniki, które wykazują pogorszenie w porównaniu do pierwotnego klasyfikatora.

	czułość	dokładność	precyzja	specyficzność
Brak powiększenia	0,749 ± 0,042	0,779 ± 0,026	0,752 ± 0,041	0,803 ± 0,045
Odbicie	0,739 ± 0,030	0,777 ± 0,019	0,754 ± 0,043	0,808 ± 0,034
Jasność i kontrast	0,784 ± 0,047	0,774 ± 0,036	0,732 ± 0,059	0,766 ± 0,079
Rotacja 10°	0,777 ± 0,057	0,781 ± 0,025	0,744 ± 0,054	0,786 ± 0,057
Rotacja 360°	0,797 ± 0,059	0,777 ± 0,030	0,728 ± 0,049	0,761 ± 0,059
Zniekształcenie	0,769 ± 0,069	0,777 ± 0,029	0,740 ± 0,049	0,785 ± 0,048
Przybliżenie	0,780 ± 0,056	0,777 ± 0,019	0,735 ± 0,040	0,774 ± 0,054
Losowe wycięcie	0,779 ± 0,050	0,773 ± 0,029	0,732 ± 0,063	0,773 ± 0,068
Nałożenie maski	0,781 ± 0,064	0,787 ± 0,019	0,752 ± 0,051	0,792 ± 0,060

Tabela 11.20: Wartości miar oceny klasyfikacji zdjęć dna oka dla różnych sposobów sztucznego powiększenia zbioru uczącego przy użyciu fastai. Oszacowanie błędu otrzymano w wyniku walidacji krzyżowej.

	czułość	dokładność	precyzja	specyficzność
Brak powiększenia	0,928	0,477	0,825	0,096
Odbicie	0,330	0,488	0,877	0,560
Jasność i kontrast	0,955	0,879	0,841	0,944
Rotacja 10°	0,661	0,059	0,074	0,276
Rotacja 360°	0,939	0,678	0,176	0,713
Zniekształcenie	0,912	0,725	0,357	0,242
Przybliżenie	0,419	0,221	0,192	0,012
Losowe wycięcie	0,390	0,653	0,402	0,578
Nałożenie maski	0,419	0,221	0,192	0,012

Tabela 11.21: P-wartości testu Shapiro-Wilka dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w fastai. Przyjmowany poziom istotności testu: $p = 0,050$. Na czerwono zaznaczono wartości poniżej poziomu istotności.

	czułość	dokładność	precyzja	specyficzność
Odbicie	2,0	1,8	1,1	1,8
Jasność i kontrast	1,2	1,9	2,0	3,0
Rotacja 10°	1,8	1,1	1,7	1,6
Rotacja 360°	1,9	1,4	1,4	1,7
Zniekształcenie	2,7	1,3	1,4	1,1
Przybliżenie	1,8	1,8	1,1	1,4
Losowe wycięcie	1,4	1,3	2,3	2,2
Nałożenie maski	2,3	1,8	1,5	1,7

Tabela 11.22: Wartości statystyki testowej F-Snedecora dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w fastai. Zbiór krytyczny: $(2, 4, \infty)$. Na czerwono zaznaczono wartości znajdujące się w zbiorze krytycznym. Przyjmowany poziom istotności testu: $p = 0,050$.

Zastosowanie kilku metod sztucznego powiększenia zbioru uczącego jednocześnie

Na podstawie otrzymanych wyników sprawdzono, jak zmienią się oceny klasyfikacji przy zastosowaniu kilku transformacji jednocześnie. W tym celu stworzono zbiór treningowy, w któ-

	czułość	dokładność	precyzja	specyficzność
Odbicie	-0,72	-0,19	0,13	0,31
Jasność i kontrast	2,26	-0,46	-1,07	-1,64
Rotacja 10°	1,58	0,25	-0,44	-0,97
Rotacja 360°	2,65	-0,15	-1,45	-2,27
Zniekształcenie	1,01	-0,17	-0,73	-1,14
Przybliżenie	1,77	-0,23	-1,16	-1,65
Losowe wycięcie	1,83	-0,65	-1,04	-1,64
Nałożenie maski	1,66	0,95	0,03	-0,60

Tabela 11.23: Wartości statystyki testowej t-Studenta dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w fastai. Zbiór krytyczny dla hipotezy alternatywnej w postaci poprawienia wyników: $\langle 1, 7, \infty \rangle$. Na zielono zaznaczono wartości w nim zawierające się. Zbiór krytyczny dla hipotezy alternatywnej w postaci pogorszenia wyników: $(\infty, -1, 7)$. Na czerwono zaznaczono wartości w nim zawierające się. Przyjmowany poziom istotności testu: $p = 0,050$.

rym zastosowano odbicie, zmianę jasności i kontrastu, losowy obrót o kąt z przedziału $[0^\circ, 360^\circ]$, przybliżanie oraz losowe wycinanie. Dla tego zbioru transformacji użyto nazwy „5 transformacji”.

Sprawdzono również jak wpłynie na klasyfikację zastosowanie zbioru domyślnych przekształceń zaimplementowanych w bibliotece fastai (w tabeli oznaczonych jako „domyślne”).

Sprawdzono, czy dane pochodzące z walidacji krzyżowej wyżej wymienionych modeli mają rozkład normalny za pomocą testu Shapiro-Wilka (tabela 11.25) oraz czy ich wariancje są homogenicznie (test F-Snedecora, tabela 11.26). Następnie użyto testu t-Studenta, aby przeprowadzić wnioskowanie statystyczne dotyczące poprawy wyników klasyfikacji (tabela 11.27).

Analogicznie do poprzednich przypadków, zastosowano zielony kolor do oznaczenia statystycznie istotnej poprawy wyników, a czerwonego dla pogorszenia.

	czułość	dokładność	precyzja	specyficzność
5 transformacji	$0,699 \pm 0,064$	$0,796 \pm 0,052$	$0,735 \pm 0,038$	$0,753 \pm 0,015$
Domyślne	$0,768 \pm 0,050$	$0,785 \pm 0,25$	$0,753 \pm 0,042$	$0,799 \pm 0,041$

Tabela 11.24: Wartości miar oceny klasyfikacji zdjęć dna oka dla dwóch zbiorów transformacji danych wejściowych przy użyciu fastai. Oszacowanie błędu otrzymano w wyniku walidacji krzyżowej.

	czułość	dokładność	precyzja	specyficzność
5 transformacji	0,677	0,661	0,818	0,317
Domyślne	0,003	0,079	0,319	0,160

Tabela 11.25: P-wartości testu Shapiro-Wilka dla prób losowych otrzymanych w wyniku walidacji krzyżowej modeli stworzonych w fastai. Przyjmowany poziom istotności testu: $p = 0,050$. Na czerwono zaznaczono wartości poniżej poziomu istotności.

	czułość	dokładność	precyzja	specyficzność
5 transformacji	2,3	4,1	1,2	8,9
Domyślne	1,4	1,1	1,0	1,2

Tabela 11.26: Wartości statystyki testowej F-Snedecora dla prób losowych otrzymanych w wyniku walidacji krzyżowej dla fastai. Zbiór krytyczny: $(2, 4, \infty)$. Na czerwono zaznaczono wartości znajdujące się w zbiorze krytycznym. Przyjmowany poziom istotności testu: $p = 0,050$.

	czułość	dokładność	precyzja	specyficzność
5 transformacji	-2,61	1,18	-1,18	-4,20
Domyślne	1,16	0,65	0,07	-0,30

Tabela 11.27: Wartości statystyki testowej t-Studenta dla prób losowych otrzymanych w wyniku walidacji krzyżowej dla fastai. Zbiór krytyczny dla hipotezy alternatywnej w postaci pogorszenia wyników: $(\infty, -1, 7)$. Na czerwono zaznaczono wartości w nim zawierające się. Przyjmowany poziom istotności testu: $p = 0,050$.

11.3.6. Podsumowanie

Sposobami powiększania zbioru obrazów dającymi statystycznie istotną poprawę wyników czułości są: zmiana jasności i kontrastu, rotacja o losowy kąt z zakresu $[0^\circ, 360]$, przybliżanie oraz losowe wycinki. Należy jednakże zwrócić uwagę, że poprawa ta wiąże się z pogorszeniem pozostały miar, szczególnie w przypadku rotacji, gdzie zmniejszenie się specyficzności klasyfikatora jest istotne statystycznie.

Kombinacje kilku metod sztucznego powiększania zbioru uczącego nie dały satysfakcjonujących rezultatów. Połączenie metod, które osobno poprawiały czułość klasyfikatora, ostatecznie pogorszyło wyniki. Przy skorzystaniu ze zbioru metod opracowanego przez autorów fastai otrzymano poprawę rezultatów, ale niebędącą istotną statystycznie.

Na podstawie uzyskanych rezultatów, wybór modelu, będącego najlepszym klasyfikatorem danych wejściowych, nie był prosty. Niestety, stosując metody zwiększania ilości danych wejściowych poprawiające czułość klasyfikatora, zmniejszały się równocześnie pozostałe parametry. Pomimo nieistotności wyników na zadanym poziomie 5%, wybrany został model, w którym zastosowano nakładanie maski. Charakteryzuje się on zwiększeniem trzech z czterech analizowanych miar.

Kontynuując badania nad klasyfikatorem, należały sprawdzić wpływ wielkości oraz liczby nakładanych na obraz masek. Być może poprzez lepsze dopasowanie parametrów tej transformacji, osiągnięto by statystycznie istotną poprawę wyników.

11.4. Porównanie eksperymentów

U. A. ROMANIUK

Ostatnią fazą przeprowadzania eksperymentu było porównanie wyników otrzymanych w obu jego częściach.

Na początku sprawdzono jaki wpływ na wyniki tej samej sieci ma biblioteka, która została użyta do wykonania eksperymentu. W tym celu przytoczono wyniki sieci AlexNet bez użycia powiększeń zbioru uczącego dla obu eksperymentów (tabela 11.28).

AlexNet	czułość	dokładność	precyza	specyficzność
eksperyment przy użyciu caffe	$0,820 \pm 0,040$	$0,673 \pm 0,032$	$0,599 \pm 0,030$	$0,550 \pm 0,052$
eksperyment przy użyciu fastai	$0,646 \pm 0,043$	$0,726 \pm 0,019$	$0,711 \pm 0,053$	$0,790 \pm 0,048$

Tabela 11.28: Zestawienie wyników miar oceny klasyfikatorów otrzymanych przy użyciu caffe oraz fastai. Ocenę błędu otrzymano w wyniku walidacji krzyżowej.

Sprawdzono, czy uzyskane wyniki z eksperymentu z fastai są statystycznie lepsze od tych uzyskanych w caffe. Wykonano test F-Snedecora 11.29 oraz test t-Studenta (tabela 11.30).

testy	czułość	dokładność	precyza	specyficzność
F-Snedecora	1,2	2,8	3,1	1,2

Tabela 11.29: Porównanie AlexNet caffe i fastai. Wartości statystyki testowej F-Snedecora dla prób losowych otrzymanych w wyniku walidacji krzyżowej. Zbiór krytyczny: $\langle 2, 4, \infty \rangle$. Na czerwono zaznaczono wartości znajdujące się w zbiorze krytycznym. Przyjmowany poziom istotności testu: $p = 0,050$.

testy	czułość	dokładność	precyza	specyficzność
t-Studenta	-11,9	5,7	7,4	13,6

Tabela 11.30: Porównanie AlexNet caffe i fastai. Wartości statystyki testowej t-Studenta dla prób losowych otrzymanych w wyniku walidacji krzyżowej. Zbiór krytyczny dla hipotezy alternatywnej w postaci poprawienia wyników: $\langle 1, 7, \infty \rangle$. Na zielono zaznaczono wartości w nim zawierające się. Zbiór krytyczny dla hipotezy alternatywnej w postaci pogorszenia wyników: $\langle \infty, -1, 7 \rangle$. Na czerwono zaznaczono wartości w nim zawierające się. Przyjmowany poziom istotności testu: $p = 0,050$.

Następnie porównano ze sobą najlepsze uzyskane wyniki modeli przez obie biblioteki.

Najlepszym modelem stworzonym za pomocą caffe okazała się architektura AlexNet z dwoma dodatkowymi warstwami połączonymi zamiast jednej, z zastosowaniem sztucznego powiększenia zbioru danych w postaci 9 wyciętych obrazów z jednego.

Przy użyciu fastai najlepszą architekturą okazała się sieć ResNet50, gdzie zastosowano nałożenie maski.

Wyniki dla obu sieci znajdują się w tabeli 11.31.

Najlepsza wybrana sieć	czułość	dokładność	precyzja	specyficzność
eksperyment przy użyciu caffe	$0,819 \pm 0,027$	$0,715 \pm 0,033$	$0,646 \pm 0,041$	$0,631 \pm 0,059$
eksperyment przy użyciu fastai	$0,781 \pm 0,064$	$0,787 \pm 0,019$	$0,752 \pm 0,051$	$0,792 \pm 0,060$

Tabela 11.31: Zestawienie wyników miar oceny klasyfikatorów otrzymanych przy użyciu caffe oraz fastai. Ocenę błędu otrzymano w wyniku walidacji krzyżowej.

Sprawdzono także statystyczną poprawę wyników uzyskanych dla modelu uzyskanego w fastai od wyników uzyskanych dla modelu w caffe (tabela 11.32, 11.33).

testy	czułość	dokładność	precyzja	specyficzność
F-Snedecora	5,6	3,0	1,5	1,0

Tabela 11.32: Porównanie caffe i fastai. Wartości statystyki testowej F-Snedecora dla prób losowych otrzymanych w wyniku walidacji krzyżowej. Zbiór krytyczny: $\langle 2, 4, \infty \rangle$. Na czerwono zaznaczono wartości znajdujące się w zbiorze krytycznym. Przyjmowany poziom istotności testu: $p = 0,050$.

testy	czułość	dokładność	precyzja	specyficzność
t-Studenta	-2,2	7,5	6,5	7,7

Tabela 11.33: Porównanie caffe i fastai. Wartości statystyki testowej t-Studenta dla prób losowych otrzymanych w wyniku walidacji krzyżowej. Zbiór krytyczny dla hipotezy alternatywnej w postaci poprawienia wyników: $\langle 1, 7, \infty \rangle$. Na zielono zaznaczono wartości w nim zawierające się. Zbiór krytyczny dla hipotezy alternatywnej w postaci pogorszenia wyników: $(\infty, -1, 7)$. Na czerwono zaznaczono wartości w nim zawierające się. Przyjmowany poziom istotności testu: $p = 0,050$.

Miary oceny klasyfikatora uzyskane przy użyciu biblioteki fastai okazały się lepsze od wyników uzyskanych przy użyciu biblioteki caffe.

Rozdział 12

Dyskusja

Przeprowadzone eksperymenty doprowadziły do stworzenia dwóch klasyfikatorów o satysfakcjonujących wynikach. Pomimo posiadania niewielkiego zbioru obrazów, udało się stworzyć poprawnie wyuczone głębokie sieci neuronowe. Osiągnięto to przy pomocy *transfer learningu* oraz sztucznego powiększania posiadanych danych.

Otrzymane wyniki mogą być niewystarczające do stworzenia testu przesiewowego. Pomimo dołożonych starań nie udało się istotnie zwiększyć miar oceny klasyfikatorów.

Jedną z przyczyn braku dalszej poprawy wyników, mogą być błędne klasyfikacje otrzymanych danych. Ze względu na charakterystykę uczenia się sieci neuronowych, w zbiorze uczącym powinny znajdować się jedynie zdjęcia poprawnie przypisane do klas. Obrazów, co do których występują wątpliwości do ich przydziału, nie należy uwzględnić podczas uczenia klasyfikatora.

Interesujące są źródła różnic w wynikach pochodzących z różnych części eksperymentu. Różnice te mogą wynikać z zastosowania dwóch modeli konwolucyjnych sieci neuronowych, jakimi są AlexNet i ResNet.

W pierwszej części eksperymentu zastosowanie innej sieci niż AlexNet byłoby bardzo trudne. Wynika to z potrzeby ręcznego dopasowywania architektury sieci. Kolejnym czynnikiem jest manualny dobór parametrów uczenia, których wpływ na wyniki pozostaje nieokreślony do momentu wytrenowania klasyfikatora. Następnym argumentem za wyborem tego modelu sieci, jest fakt posiadania niewielkich zasobów obliczeniowych. AlexNet, w odróżnieniu od sieci ResNet50, zawiera mniejszą liczbę warstw, co przekłada się na krótszy czas uczenia.

Ostatecznie druga część eksperymentu używała sieci ResNet, jednakże w trakcie pracy nauczono klasyfikator także na podstawie sieci AlexNet. Pozwala to dokładnie porównać oba sposoby. To porównanie pokazuje wyższość drugiego podejścia w stosunku do tego, użytego w pierwszej części eksperymentu. Może to być powodowane przez zastosowanie algorytmu cyklicznego współczynnika uczenia przez bibliotekę.

Kolejną istotną różnicą jest sposób doboru parametrów uczenia. W pierwszej części eksperymentu wszystkie parametry ustawiane były ręcznie. Natomiast w drugiej części proces ten był wspomagany poprzez funkcje zaimplementowane w bibliotece fastai.

Rozdział 13

Wnioski

Podczas tworzenia zbioru zdjęć ważnym elementem jest odpowiednie przypisanie ich do klas. Klasyfikacja zbioru danych może być przeprowadzona przez wielu lekarzy niezależnie. Pozwoli to wykluczyć obrazy wątpliwe, czyli takie, których ocena przez lekarzy znacząco różniła się między sobą.

Współpraca z lekarzem istotna jest także w trakcie uczenia sieci neuronowej. Po wycze- niu klasyfikatora powinno się poddać ponownej ocenie zdjęcia ze zbioru treningowego, których prawdopodobieństwo przynależności do ich prawdziwej klasy było najmniejsze. W przypadku stwierdzenia przez lekarza wątpliwości co do takiego obrazu należy usunąć go ze zbioru uczącego, a następnie ponownie wytrenować klasyfikator.

Z technicznego punktu widzenia trenowanie sieci neuronowej jest znacznie lepiej przepro- wadzać przy użyciu biblioteki fastai. Pozwala to oszczędzić czas pracy eksperymentatora oraz zmniejszyć występowanie błędu ludzkiego w wykonywanej pracy.

Dużym atutem omawianej biblioteki jest umożliwienie prostej obsługi nawet skomplikowanych głębokich sieci neuronowych. Jest to istotne ze względu na rosnącą z roku na rok złożoność architektur najskuteczniejszych sieci.

Największym zyskiem stosowania fastai jest możliwość uczenia klasyfikatora za pomocą metody cyklicznego współczynnika uczenia. Klasyfikatory trenowane przy użyciu tej metody osiągają znacznie lepszą generalizację danych uczących.

Możliwością poprawy klasyfikacji obrazów jest wytrenowanie kilku, różnych klasyfikatorów, a następnie uśrednienie tak otrzymanych wyników.

Bibliografia

- [1] Life expectancy at birth, total (years). https://data.worldbank.org/indicator/sp_dyn_le00_in?end=2017&start=1960&view=chart. Accessed: 2019-06-01.
- [2] GBD Compare — Viz Hub. <https://vizhub.healthdata.org/gbd-compare/>. Accessed: 2019-06-01.
- [3] Sharon Kingman. Glaucoma is second leading cause of blindness globally. *Bulletin of the World Health Organization*, 82:887–888, 2004.
- [4] Alicja R Rudnicka, Shahrul Mt-Isa, Christopher G Owen, Derek G Cook, and Deborah Ashby. Variations in primary open-angle glaucoma prevalence by age, gender, and race: a bayesian meta-analysis. *Investigative ophthalmology & visual science*, 47(10):4254–4261, 2006.
- [5] Roger CW Wolfs, Petra H Borger, Raan S Ramrattan, Caroline CW Klaver, Caroline AA Hulsman, Albert Hofman, Johannes R Vingerling, Roger A Hitchings, and Paulus TVM de Jong. Changing views on open-angle glaucoma: definitions and prevalences—the rotterdam study. *Investigative ophthalmology & visual science*, 41(11):3309–3321, 2000.
- [6] European glaucoma society terminology and guidelines for glaucoma, 4th edition - chapter 3: Treatment principles and optionssupported by the egs foundation. *British Journal of Ophthalmology*, 101(6):130–195, 2017.
- [7] Harsha L Rao, Linda M Zangwill, Robert N Weinreb, Pamela A Sample, Luciana M Alencar, and Felipe A Medeiros. Comparison of different spectral domain optical coherence tomography scanning areas for glaucoma diagnosis. *Ophthalmology*, 117(9):1692–1699, 2010.
- [8] Ricardo Y Abe, Carolina PB Gracitelli, and Felipe A Medeiros. The use of spectral-domain optical coherence tomography to detect glaucoma progression. *The open ophthalmology journal*, 9:78, 2015.
- [9] Karl U Bartz-Schmidt, Gabriele Thumann, Christian P Jonescu-Cuypers, and Gunther K Krieglstein. Quantitative morphologic and functional evaluation of the optic nerve head in chronic open-angle glaucoma. *Survey of ophthalmology*, 44:S41–S53, 1999.
- [10] Stephen Marsland. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2011.
- [11] Th Bayes. An essay towards solving a problem in the doctrine of chances. 1763. *MD computing: computers in medical practice*, 8(3):157, 1991.

- [12] Sieci neuronowe - wprowadzenie. <http://zsi.tech.us.edu.pl/~nowak/asi/w5.pdf>. Accessed: 2019-05-20.
- [13] Neural network. https://pl.wikipedia.org/wiki/Sie%C4%87_neuronowa#/media/File:Neuralnetwork.png. Accessed: 2019-05-20.
- [14] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.
- [15] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. *arXiv preprint arXiv:1507.06947*, 2015.
- [16] Paulo Henrique Siqueira, Maria Teresinha Arns Steiner, and Sérgio Scheer. A new approach to solve the traveling salesman problem. *Neurocomputing*, 70(4-6):1013–1021, 2007.
- [17] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.
- [18] Chun Hung Li and CK Lee. Minimum cross entropy thresholding. *Pattern recognition*, 26(4):617–625, 1993.
- [19] Jarosław Michalkiewicz. Modified kolmogorov’s theorem. *Mathematica Applicanda*, 37(51/10), 2009.
- [20] Honglak Lee. Deep learning methods for vision. https://www.slideshare.net/zukun/p04-restricted-boltzmann-machines-cvpr2012-deep-learning-methods-for-vision?from_action=save. Accessed: 2019-05-25.
- [21] Sumit Saha. A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Accessed: 2019-05-27.
- [22] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.
- [23] Krzysztof Odrzywołek. Wykorzystanie głębszych sieci neuronowych w weryfikacji mówcy. Praca magisterska, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej, Akademia Górnictwa-Hutnicza im. Stanisława Staszica w Krakowie, 2016.
- [24] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.
- [25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

- [26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [27] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- [28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.
- [30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [31] Trend wyszukiwań hasła transfer learning według google. <https://trends.google.com/trends/explore?date=today%205-y&q=transfer%20learning>. Accessed: 2019-05-28.
- [32] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.
- [33] Ashnil Kumar, Jinman Kim, David Lyndon, Michael Fulham, and Dagan Feng. An ensemble of fine-tuned convolutional neural networks for medical image classification. *IEEE journal of biomedical and health informatics*, 21(1):31–40, 2016.
- [34] Mostafa Mehdipour Ghazi, Berrin Yanikoglu, and Erchan Aptoula. Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing*, 235:228–235, 2017.
- [35] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.
- [36] Zeshan Hussain, Francisco Gimenez, Darvin Yi, and Daniel Rubin. Differential data augmentation techniques for medical imaging classification tasks. In *AMIA Annual Symposium Proceedings*, volume 2017, page 979. American Medical Informatics Association, 2017.
- [37] Yi Zhen, Lei Wang, Han Liu, Jian Zhang, and Jiantao Pu. Performance assessment of the deep learning technologies in grading glaucoma severity. *arXiv preprint arXiv:1810.13376*, 2018.
- [38] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

- [39] Patrick Royston. Approximating the shapiro-wilk w-test for non-normality. *Statistics and computing*, 2(3):117–119, 1992.
- [40] Andrey Kolmogorov. Sulla determinazione empirica di una lgge di distribuzione. *Inst. Ital. Attuari, Giorn.*, 4:83–91, 1933.
- [41] George W Snedecor and William G Cochran. Statistical methods, eight edition. *Iowa state University press, Ames, Iowa*, 1989.
- [42] Richard G Lomax and Debbie L Hahs-Vaughn. *Statistical concepts: A second course*. Routledge, 2013.
- [43] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.
- [44] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [45] Phillip M Cheng and Harshawn S Malhi. Transfer learning with convolutional neural networks for classification of abdominal ultrasound images. *Journal of digital imaging*, 30(2):234–243, 2017.
- [46] fast.ai.
- [47] PyTorch. <https://pytorch.org/>. Accessed: 2019-06-01.
- [48] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

Dodatek A.

Kod używany podczas eksperymentu w fastai

K. A. FILIPIUK

Poniżej przytoczono kod napisany w języku Python przy użyciu biblioteki fastai, użyty podczas przeprowadzania eksperymentu.

```
from fastai import *
from fastai.vision import *

path = Path('/home/jupyter/data/')
classes = ['healthy', 'glaucoma']
for c in classes:
    print(c)
    verify_images(path/c, delete=True, max_workers=8)

np.random.seed(42)

data_do_flip = []
learner_do_flip = []
tfms = get_transforms(do_flip=True, flip_vert=False)
for i in range(16):
    print(i)
    tmp_data = ImageDataBunch.from_folder(
        path,
        train=".",
        valid_pct=0.2,
        ds_tfms=tfms,
        size=256,
        num_workers=4).normalize(imagenet_stats)
    data_do_flip.append(tmp_data)
    tmp_learner = cnn_learner(tmp_data,
        models.resnet50,
        pretrained=True,
        metrics=accuracy)
    learner_do_flip.append(tmp_learner)
#tmp_learner.lr_find()
```

```

for i in range(16):
    learner_do_flip[i].recorder.plot()

learning_rate = 1e-02
for i in range(16):
    print(i)
    learner_do_flip[i].fit_one_cycle(5, learning_rate)

for i in range(16):
    print(i)
    learner_do_flip[i].unfreeze()
    #learner_do_flip[i].lr_find()

for i in range(16):
    learner_do_flip[i].recorder.plot()

learning_rates = slice(0, 1e-05)

for i in range(16):
    print(i)
    learner_do_flip[i].fit_one_cycle(20, learning_rates)

sen = np.zeros(16)
spe = np.zeros(16)
pre = np.zeros(16)
acc = np.zeros(16)

for i in range(16):
    final_preds, y = learner_do_flip[i].TTA()
    tp = 0
    tn = 0
    fp = 0
    fn = 0
    threshold = 0.5
    for final_preds, label in zip(final_preds, y):
        prob_of_glaucoma = final_preds[0]
        if prob_of_glaucoma > threshold and label == 0:
            tp += 1
        if prob_of_glaucoma > threshold and label == 1:
            fp += 1
        if prob_of_glaucoma < threshold and label == 0:
            fn += 1
        if prob_of_glaucoma < threshold and label == 1:
            tn += 1
    print(tp, tn, fp, fn)
    sen[i] = 1.0 * tp / (tp + fn)
    spe[i] = 1.0 * tn / (tn + fp)
    pre[i] = 1.0 * tp / (tp + fp)

```

```

acc[ i ] = 1.0 * ( tp + tn ) / ( tp + tn + fp + fn )

from scipy.stats import shapiro

file = open("resnet50_all_suggested.txt", "w")

file.write("sensitivity:\t"
+ str(np.mean(sen))
+ "\t±\t"
+ str(np.std(sen, ddof=1))
+ "\tnorm_p-value:\t"
+ str(shapiro(sen)[1]) + "\n")

file.write("specificity:\t"
+ str(np.mean(spe))
+ "\t±\t"
+ str(np.std(spe, ddof=1))
+ "\tnorm_p-value:\t"
+ str(shapiro(spe)[1]) + "\n")

file.write("precision:\t\t"
+ str(np.mean(pre))
+ "\t±\t"
+ str(np.std(pre, ddof=1))
+ "\tnorm_p-value:\t"
+ str(shapiro(pre)[1]) + "\n")

file.write("accuracy:\t\t"
+ str(np.mean(acc))
+ "\t±\t"
+ str(np.std(acc, ddof=1))
+ "\tnorm_p-value:\t"
+ str(shapiro(acc)[1]) + "\n")

file.close()

```