Polska Akademia Nauk

Instytut Biocybernetyki i Inżynierii Biomedycznej

Praca doktorska

Proces gojenia ścięgna Achillesa oceniany przez fuzję danych z wykorzystaniem głębokich sieci neuronowych

Autor: mgr inż. Norbert Kapiński

Kierujący pracą: dr hab. inż. Antoni Grzanka

Promotor pomocniczy: dr Jakub Zieliński

Warszawa, wrzesień 2018

Streszczenie

The abstract will go here....
W tym miejscu można umieścić abstrakt pracy. W przeciwnym wypadku należy usunąć/zakomentować ninijeszy fragment kodu.

Spis treści

Spis rysunków

Spis tabel

Rozdział 1

Wstęp

Logistic regression — 1958

Hidden Markov Model — 1960

Stochastic gradient descent — 1960

Support Vector Machine — 1963

k-nearest neighbors — 1967

Artificial Neural Networks — 1975

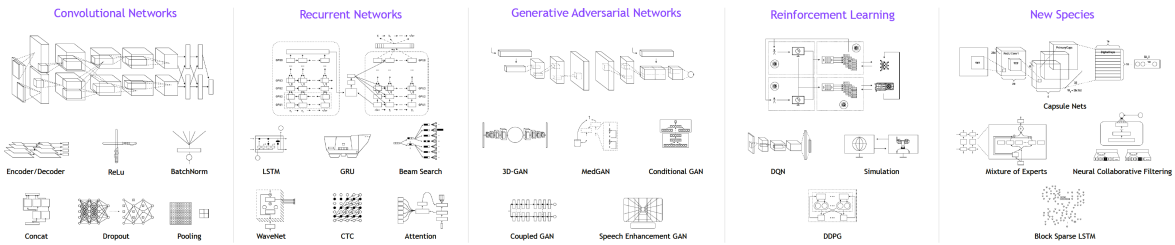
Expectation Maximization — 1977

Decision tree — 1986

Q-learning — 1989

Random forest — 1995

4% badań dotyczy oceny postępu w leczeniu [NVIDIA]



Rysunek 1.1: Podział przedstawiający różne rodzaje współczesnych głębokich sieci neuronowych.

Rozdział 2

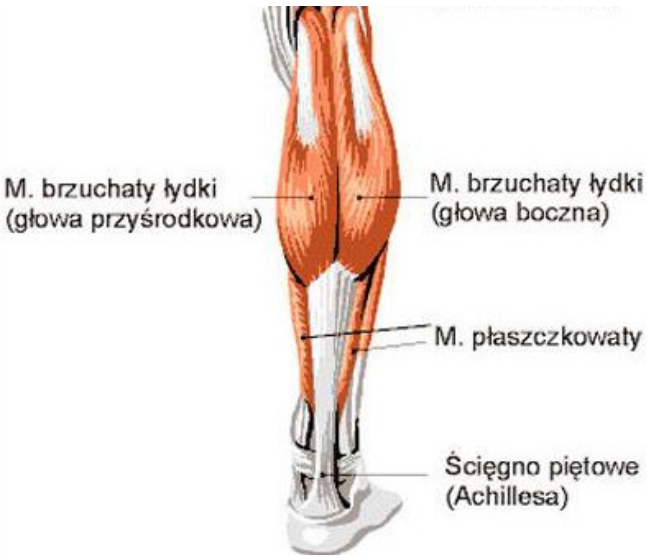
Cel i przebieg pracy

Rozdział 3

Monitorowanie procesu gojenia ścięgna Achillesa

3.1 Ścięgno Achillesa

Ścięgno Achillesa, nazywane również ścięgnem piętowym, jest największym i najsilniejszym ścięgnem występującym w ciele ludzkim. Stanowi wspólne zakończenie mięśnia trójgłowego łydki, w którego skład wchodzą dwie głowy mięśnia brzuchatego i mięsień płaszczkowaty. Całość struktury zlokalizowana jest w tylnym, powierzchownym przedziale łydki, co zostało przedstawione na Rysunku 3.1. Z obu głów (brzuścców)



Rysunek 3.1: Lokalizacja mięśnia trójgłowego łydki wraz ze ścięgnem Achillesa.

mięśnia brzuchatego łydki wyrasta jedno szerokie, płaskie ścięgno, które jest początkiem części brzuchatej ścięgna Achillesa. Następnie ścięgno to łączy się z włóknami pochodzącymi od mięśnia płaszczkowatego, które układają się stycznie do wcześniej powstałej struktury. Wówczas kształt ulega stopniowemu zwężeniu i zaokrągleniu, aż do punktu o minimalnej szerokości (około 4 cm nad przyczepem dolnym [1]). W rejonie samego przyczepu dolnego znajdującego się na tylnej powierzchnia kości piętowej, ścięgno ponownie jest płaskie i szerokie.

W kolejnych podsekcjach szczegółowo omówiona została anatomia ścięgna Achillesa, jego biomechanika, potencjalne urazy wraz z czynnikami im sprzyjającymi oraz proces gojenia i możliwości jego wspomagania. Wszystkie te aspekty są istotne z uwagi na możliwości monitorowania procesów fizjologicznych występujących w ścięgnie.

3.1.1 Anatomia

Średnia długość ścięgna Achillesa to 15 cm (11 - 26 cm). Średnia szerokość w rejonie początku wynosi 6.8 cm (4,5 - 8, 6 cm). Następnie, stopniowo ścięgno ulega zwężeniu do punktu o minimalnej szerokości 1.8 cm (1,2 - 2,6 cm). W rejonie samego przyczepu struktura ponownie się rozszerza i jej szerokość wynosi średnio 3.4 cm (2,0 - 4,8 cm) [2-3]. Zewnętrzną część ścięgna Achillesa stanowi ościęgno utworzone z tkanki łącznej włóknistej. Achil -Histologia -Unaczynienie (krew, nerwy)

3.1.2 Biomechanika

Zadaniem ścięgien jest transfer siły mięśniowej do układu szkieletowego.

3.1.3 Urazy i czynniki im sprzyjające

3.1.4 Leczenie, fazy gojenia i rehabilitacja

3.2 Zastosowanie rezonansu magnetycznego

3.3 Zastosowanie ultrasonografii

3.4 Zastosowanie badań biomechanicznych

3.5 Inne metody

Rozdział 4

Konwolucyjne sieci neuronowe

Konwolucyjne sieci neuronowe (ang. *Convolutional Neural Networks*, CNN) są biologicznie inspirowanymi sztucznymi sieciami neuronowymi. Należą do zbioru *głębokich sieci neuronowych* (ang. *Deep Neural Networks*, DNN), które z kolei są podzbiorem *systemów uczących się* (ang. *Machine Learning Systems*) tj. algorytmów nie wymagających eksplicytne programowania do ustalenia swoich parametrów.

Pierwsze matematyczne formalizmy dotyczące CNN zostały zaproponowane już w latach 40-tych XX wieku, natomiast fundamentalne inspiracje biologiczne dały badania nad korą wzrokową kotów Hubel'a i Wiesel'a z lat 50-tych i 60-tych (zob. [73]). M.in. dzięki pracom tych neurofizjologów ustalono, że kora wzrokowa zawiera złożone układy komórek, które odpowiadają za przetwarzanie informacji z wybranych regionów pola widzenia, sumarycznie pokrywając je w całości. Komórki kory wzrokowej działają zatem jak lokalne filtry przestrzeni wejściowej zaprojektowane, tak aby wydobyć istotne informacje z naturalnych obrazów. Dla przykładu reagują na orientację linii, kształty i kolory.

W odróżnieniu od naturalnych struktur biologicznych, sieci konwolucyjne operują przeważnie na reprezentacji cyfrowej obrazu. Zapisywana jest ona w postaci tablicy liczb tzw. *macierzy* o współrzędnych (x,y), gdzie x to kolumna macierzy, a y wiersz. W przypadku obrazów trójwymiarowych dochodzi jeszcze składowa z, a zamiast macierzy użyty jest wówczas *tensor*, czyli wielowymiarowe uogólnienie.

W obrazach cyfrowych dla każdego punktu (x,y), tzw. *piksela*, kodowana jest informacja o wartości funkcji obrazowej $I(x,y)$. Na tej podstawie dzielimy obrazy na:

- binarne – gdzie kodowane są jedynie dwie możliwe wartości I (0 lub 1);

- monochromatyczne – gdzie kodowana jest informacja o natężeniu jednej barwy (najczęściej są to odcienie szarości lub brązu tzw. *sepia*);

- kolorowe – gdzie kodowane są wartości natężenia składowych.

Do zakodowania informacji o wartości funkcji I w obrazie binarnym potrzebny jest jeden bit na punkt. Odcienie obrazu monochromatycznego lub pojedyncze składowe punktu obrazu kolorowego kodowane są na 8–16 bitach, co zależy od zakresu wartości występujących w danych lub precyzji wymaganej do obliczeń. Przykładowo, obrazy medyczne przetwarzane w tej pracy należą do grupy obrazów monochromatycznych i są zapisywane w reprezentacji 16 bitowe z uwagi na zakres wartości występujących w sygnale MR.

W sieciach konwolucyjnych procesy zachodzące w korze wzrokowej są modelowane przy użyciu odpowiednio dobranych parametrów i funkcji służących do ekstrakcji istotnych informacji obrazowych i ich przetwarzania. Parametry wykorzystywane do wstępnej ekstrakcji informacji grupowane są w *filtry*, które realizują funkcję *splotu* maski K z kolejnymi fragmentami funkcji I. K jest najczęściej macierzą kwadratową o wymiarze N, gdzie najczęściej $N = (1, 3, 5, 7, 9, 11)$ z uwagi na rosnącą wraz z N złożonością obliczeniową. Równanie splotu można zapisać następująco:

$$I'(x, y) = \sum_{j=-n}^{n} \sum_{i=-k}^{k} I(x - j, y - i) K(j, i), \tag{4.1}$$

gdzie I' jest nową funkcją obrazową powstałą po filtracji, a i i j to kolejne współrzędne maski filtru w odniesienia do jego punktu centralnego. Dla wartości brzegowych podczas realizacji splotu, współrzędne poza sygnałem ustawiane są z reguły na wartość 0. Rzadziej stosuje się inne metody takie jak: odbicie wartości I obrazu poza jego granicami; powtórzenie wartości I obrazu bez odbicia; modyfikację wymiaru maski filtru na brzegu obrazu, tak by nie wychodziła poza jego granicę.

W kontekście sieci konwolucyjnych, pojedynczy splot filtru z obrazem wejściowym nazywany jest *cechą* (zob. [53]). W ostatnich latach powstało wiele algorytmów do wizualizacji cech umożliwiających wydajną analizę wyników działania sieci konwolucyjnych np. Saliency Maps [58], GradCam [56] albo metody bazujące na skierowanych acyklicznych grafach [41]. Zasada działania tych algorytmów bazuje na selekcji cech, które mają zasadniczy wpływ na wynik końcowy. Grupy cech obliczane na tym samym obszarze obrazu wejściowego nazywane są *mapami cech* (ang. *feature maps*). Zazwy-

czaj mapy cech zawierają hierarchicznie uporządkowaną strukturę zawierającą cechy od najniższego do najwyższego poziomu komplikacji (np. od charakterystycznych skupisk pikseli, przez obiekty składowe do finalnego kształtu struktury). Do cech najniższego rzędu zaliczyć można:

- *krawędzie* – ciągi punktów o gwałtownych zmianach $I(x,y)$;
- *rogi* – przecięcia dwóch krawędzi;
- *grzbiety* lub *doliny* – lokalne maksimum lub odpowiednio minimum funkcji I;
- *skupiska* (ang. *blobs*) – obszary jednorodnych wartości funkcji I, różniących się znacząco od najbliższego otoczenia;
- *tekstury* – charakterystyczne, przestrzenne ułożenie wartości funkcji I w powtarzające się wzory.

Wszystkie cechy wyższego rzędu są kombinacją powyższych składowych podstawowych tworząc bardziej skomplikowane struktury odpowiadające charakterystycznym obiektom znajdującym się w obrazach wejściowych.

Na podstawie cech, na końcu toru przetwarzania danych z wykorzystaniem sieci neuronowych, realizowane jest wnioskowanie końcowe, gdzie w zależności od problemu można wyszczególnić następujące zadania:

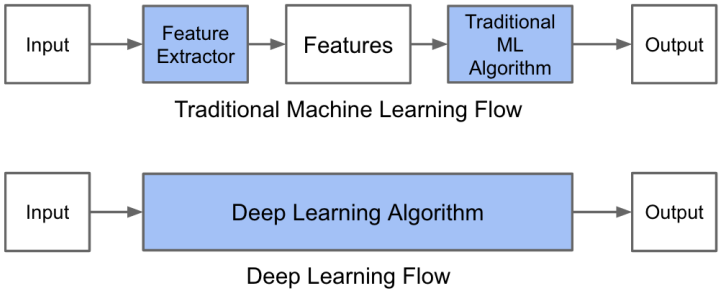
- *segmentację* – podział obrazu na spójne fragmenty, najczęściej wiążące się z wyodrębnieniem *obiektu* z *tła* na podstawie ustalonego progu lub miary np.:

$$I(x,y) \geqslant T_p \Rightarrow (x,y) \in \text{"}obiekt\text{"}$$
$$I(x,y) < T_p \Rightarrow (x,y) \in \text{"}tlo\text{"},$$

  przy czym metody ustalania wartości progowej T_p lub wielu wartości T_p^1,...,T_p^n są obszarem szerokich badań (zob. [6]).

- *klasyfikację* – przyporządkowanie obiektu do odpowiedniej klasy (np. tkanka zdrowa lub patologiczna).
- *detekcję* – binarne rozróżnienie traktujące o tym czy obiekt znajduje się w obrazie czy nie.
- *śledzenie* – detekcja lub też klasyfikacja obiektów w kolejnych krokach czasowych.

Jeszcze do niedawna tor przetwarzania danych w większości badań wykorzystujących uczenie się maszyn operacje wyliczenia cech i wnioskowanie końcowego realizowano w dwóch osobnych krokach[1]. Porównanie tego schematu z obecnie funkcjonującym podejściem realizowanym w algorytmach głębokiego uczenia się przedstawiono na Rys. 4.1. W nowym podejściu zarówno ekstrakcja cech jak i ostateczne wnioskowanie na



Rysunek 4.1: Porównanie schematów przetwarzania danych z wykorzystaniem metod głębokiego uczenia się i innych algorytmów.

ich podstawie realizowane jest w jednym kroku, co nazywane jest paradygmatem *end-to-end learning*. W kolejnej sekcji omówiono dokładniej jak wyglądała ewolucja tego podejścia do obecnej postaci.

4.1 Zarys historyczny

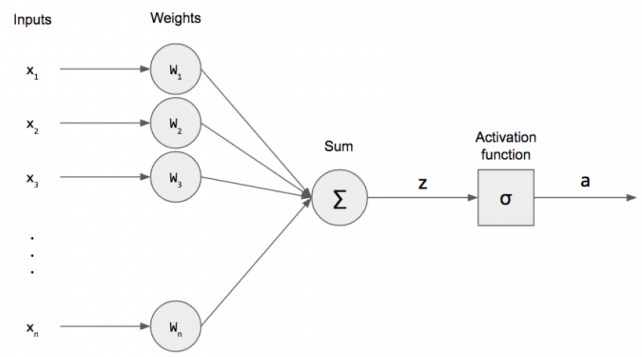
Pierwszy formalny model neuronu został zaproponowany przez Warrena McCulloch i Waltera Pitts w roku 1943 (zob. [45]). Była to bramka logiczna, której wyjście stawało się aktywne w momencie, gdy liczba aktywnych wejść przekroczyła pewien zdefiniowany próg. Taka zależność sygnału wyjściowego y od sygnałów wejściowych $x_1 ... x_n$ została potem nazwana *funkcją aktywacji neuronu*, którą zapisujemy jako:

$$y = f(x_1, x_2, ..., x_n) \tag{4.2}$$

W modelu neuronu McCulloch-Pitts można było modyfikować parametr progu, nie istniała natomiast możliwość uczenia się takiej architektury. Problem ten rozwiązano w 1957 proponując sztuczną sieć neuronową zawierającą wiele neuronów z ważonymi połączeniami między sobą (zob. [50]). Sieć nazwano perceptronem, co było implika-

[1]Są wyjątki od tej reguły jak np. [57], inkrementalny algorytm SVM

cją zamiłowania jego twórcy Franka Rosenblatta do aplikacji związanych z percepcją, zwłaszcza mowy czy pisma. Schemat sieci pokazano na Rys. 4.2.



Rysunek 4.2: Topologia perceptronu.

Zastosowanie dodatkowej jednowymiarowej tablicy współczynników, czyli *wektora wag* $w = (w_1,...,w_n)$ dało możliwość uczenia się poprzez adaptacyjną zmianę wartości poszczególnych jego elementów. W modelu Rosenblatta zastosowano ponadto progową funkcję aktywacji z progiem T_a:

$$y(z) = \begin{cases} 0, & n < T_a, \\ 1, & n \geqslant T_a, \end{cases} \tag{4.3}$$

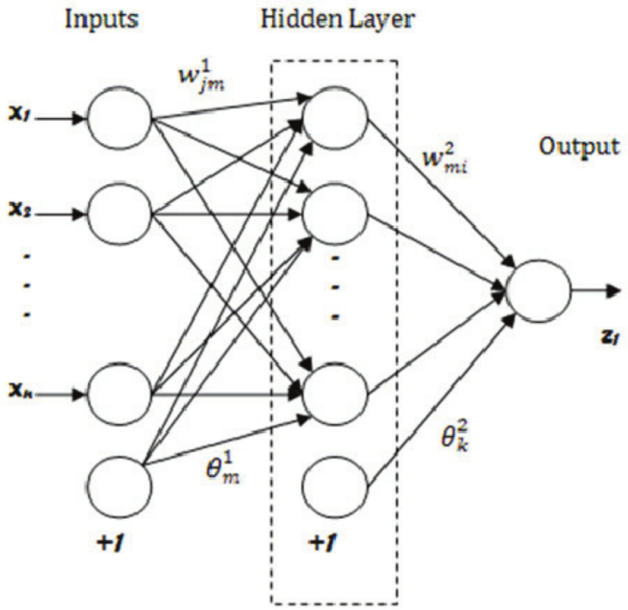
gdzie z to suma ważona wyjść poszczególnych neuronów:

$$z = \sum_{i=1}^{n} w_i x_i \tag{4.4}$$

Dla przykładu, w dwuwymiarowej przestrzeni sygnałów wejściowych x_1, x_2 działanie perceptronu można opisać jako wyliczenie funkcji liniowej rozdzielającej obserwacje o_1 od o_2. W trzech wymiarach będzie to płaszczyzna, a w n-wymiarach, hiperpłaszczyzna. Z uwagi na swój liniowy charakter perceptron proponowany przez Rosenblatta miał wiele ograniczeń, które trafnie sformułował w 1969 roku Marvin Minsky i Seymour Papert w książce *Perceptrons* (zob. [44]). Autorzy opublikowali listę problemów, których nie można było rozwiązać z użyciem perceptronu m.in. do najszerzej dyskutowanych należał przykład związany z brakiem możliwości modelowania *funkcji XOR*, aktywującej wyjście przy aktywnym jednym z dwóch wejść.

Po latach intensywnych prac, część z opisanych przez Minsky-Papert problemów udało się rozwiązać za sprawą pracy Davida Rumelharta, Geoffa Hintona i Ronalda

Williams, którzy opublikowali w 1986 roku pracę [51], traktującą o perceptronach wielowarstwowych. Schemat takiej sieci zaprezentowano na Rys. 4.3



Rysunek 4.3: Topologia perceptronu wielowarstwowego.

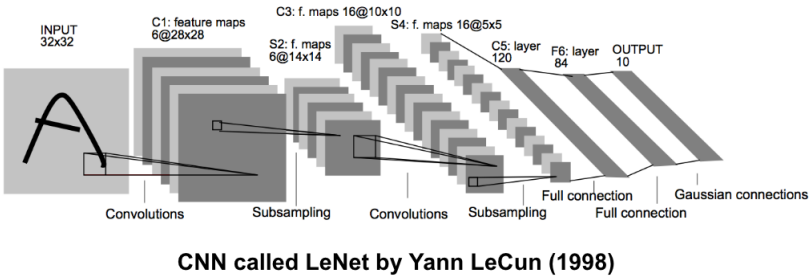
Spośród najważniejszych innowacji wprowadzonych w perceptronie wielowarstwowym wyszczególnić można zastosowanie w praktyce nowego algorytmu uczenia się sieci, który został opisany w kolejnej sekcji jak również nowej, sigmoidalnej funkcji aktywacji:

$$y(x) = \frac{1}{1 + e^{-x}} \tag{4.5}$$

Z wykorzystaniem sieci wielowarstwowych możliwe stało się modelowanie funkcji XOR jak i innych problemów nieliniowych o praktycznym wymiarze.

Kolejny przełom nastąpił w 1989 roku kiedy to Yann LeCunn, były uczeń Geaoffa Hintona, zaprezentował swoje wyniki dotyczące klasyfikacji odręcznego pisma z użyciem sieci wielowarstwowych (zob. [38]). Finalnie, badania te doprowadziły do przedstawienia w 1998 roku pierwszej sieci konwolucyjnej nazwanej LeNet (zob. [37]). Architekturę tej sieci przedstawiono na Rys. 4.4.

Sieć składała się z 7 warstw i zawierała około 60,000 parametrów. Oryginalnie, sygnał wejściowy sieci stanowił obrazek o wymiarach 32×32. W architekturze LeNet zaobserwować można było dwie podstawowe składowe współczesnych sieci konwolucyjnych wykorzystywanych do klasyfikacji obrazów tj.:



Rysunek 4.4: Topologia sieci LeNet.

- *ekstraktor cech* – część zawierająca m.in. filtry, służące do automatycznej ekstrakcji wektora cech w.
- *klasyfikator* – część wykorzystywana do zadania wnioskowania końcowego na podstawie w.

Przy użyciu takiej architektury możliwe stało się zastosowanie paradygmatu end-to-end learning rozumianego w tym kontekście jako znalezienie możliwie dobrej transformacji, która surowe obrazy przekształca bezpośrednio w ostateczną klasyfikację. Dokładny opis szkolenia się sieci konwolucyjnych i problemów z tym związanych został przedstawiony w kolejnej sekcji.

4.2 Szkolenie głębokich sieci neuronowych

Większość algorytmów szkolenia głębokich sieci neuronowych obejmuje zadanie *optymalizacji*, rozumiane jako minimalizację, bądź maksymalizację *funkcji celu* $f(x)$ przez zmianę x. W literaturze można też znaleźć inne nazwy funkcji celu takie jak *kryterium*, *funkcja kosztów*, *funkcja strat* lub *funkcja błędów* (por. [23]).

Podczas zadania optymalizacji bardzo często wykorzystuje się *pochodną funkcji* oznaczaną jako $f'(x)$ lub $\frac{\delta y}{\delta x}$, gdyż niesie ona informacje o nachyleniu funkcji w punkcie x. W praktyce funkcje celu są wielowymiarowe dlatego wykorzystywane są *pochodne cząstkowe* informujące o nachyleniu w poszczególnych wymiarach. Wektor zawierający pochodne cząstkowe funkcji nazywany jest *gradientem* i oznaczany jest jako $\bigtriangledown f(x)$. W praktyce wykorzystywane są również macierz pochodnych cząstkowych tzw. *macierz Jacobiego* oznaczana jako J oraz macierz *drugich pochodnych* cząstkowych (tj. pochodnych pochodnych) tzw. *macierz Hessego* oznaczana jako H.

Szereg metod optymalizacyjnych niegradientowych zostało opracowanych (zob. [66]), jednak z uwagi na szybkie znajdowanie lokalnych minimów, to właśnie metody optymalizacji bazujące na wartości gradientu są najczęściej używane w szkoleniu głębokich sieci neuronowych. Lokalne minimum przeważnie nie jest najlepszym możliwym rozwiązaniem, ale wiele badań wykazało następujące fakty (por. [13]):

1. Dla sieci neuronowych o dużych rozmiarach większość lokalnych minimów charakteryzuje się podobnymi wartościami, przekładającymi się na porównywalny efekt wnioskowania końcowego.

2. Prawdopodobieństwo znalezienia lokalnego minimum, którego implikacją będą niezadowalające rezultaty wnioskowania przy użyciu sieci maleje wraz ze wzrostem rozmiaru sieci.

3. Próba znalezienia globalnego minimum bardzo często prowadzi do problemu nadmiernego dopasowania, omówionego dokładniej w sekcji 4.2.1.

Podsumowując, metody gradientowe są wydajne obliczeniowo i prowadzą do znalezienia wielu satysfakcjonujących rozwiązań, które mogą być wykorzystane do rozwiązania praktycznych problemów.

Zasada działania metod gradientowych bazuje na obieraniu w kolejnych krokach iteracji następujących wartości funkcji f, przesuwając się w kierunku spadku gradientu:

$$x' = x - \epsilon \bigtriangledown f(x), \tag{4.6}$$

gdzie ϵ to *szybkość uczenia się*, parametr określający wielkość kroku.

Funkcja celu w przypadku praktycznych zadań optymalizacyjnych, wykorzystujących głębokie uczenie się jest *funkcją złożoną*, a zatem efekt jej działania jest równoważny operacjom wykonywanym przez kilka lub więcej funkcji po kolei. W takich sytuacjach do obliczeń spadku gradientu wykorzystywane są funkcje, których pochodne są znane.

Aplikuje się je do tzw. *reguły łańcuchowej*. Dla przykładu, przypuśćmy, że $v = g(p)$ i $u = f(g(p)) = f(q)$, gdzie p i q to wektory. Wówczas regułę łańcuchową można zapisać jako:

$$\frac{\delta u}{\delta p_i} = \sum_j \frac{\delta u}{\delta q_j} \frac{\delta q_j}{\delta p_i}, \tag{4.7}$$

co w zapisie wektorowym równoważne jest z równaniem:

$$\bigtriangledown_x z = (\frac{\delta q}{\delta p})^T \bigtriangledown_q u, \tag{4.8}$$

gdzie $\frac{\delta q}{\delta p}$ to macierz Jacobiego.

Regułę łańcuchową zapisaną w 4.8 prosto uogólnia się do zmiennych tensorowych (zob. [23], str. 205) i stosuje się w różnych meta-algorytmach służących do szkolenia sieci. Przykładem jest *algorytm propagacji wstecznej*, który oblicza regułę łańcuchową w wydajnej kolejności stosując działania w grafie takim jak topologia perceptronu wielowarstwowego (zob. [23]).

Proces szkolenia się sieci ma na celu najlepsze możliwe przybliżenie docelowej klasyfikacji bazując na danych przykładach, czyli *zbiorze uczącym* U. Algorytmy optymalizacyjne używane do szkolenia głębokich sieci neuronowych zazwyczaj działają pośrednio, optymalizując pewną miarę wydajności P, która jest zdefiniowana na *zbiorze testowym*, zawierającym przykłady inne niż w U. Często bierze się również pod uwagę jeszcze dodatkowy podzbiór, rozłączny z U i T – *zbiór walidacyjny*, który ma pomóc w wyborze najlepszych algorytmów i wartości parametrów wpływających na jakość szkolenia sieci.

W procesie szkolenia zmniejszana jest funkcja kosztów f w oparciu o U, a celem jest poprawa P. Algorytmy optymalizacyjne wykorzystujące cały zbiór U do liczenia wartości gradientu nazywane są *pakietowymi* lub *deterministycznymi*, gdyż przetwarzają jednocześnie wszystkie przykłady szkoleniowe. Te, które używają jednego przykładu na raz są nazywane *stochastycznymi*. Natomiast w praktyce przy szkoleniu głębokich sieci stosowane są algorytmy *minipakietowe*, wykorzystujące więcej niż jeden przykład, ale mniej niż cały zbiór. Z reguły są to liczby z przedziału 8-256. Takie podejście zapewnia kompromis między szybkością obliczeń i dokładnością estymacji wartości gradientu.

Przykładem algorytmu minipakietowego jest *stochastyczny spadek gradientu* (ang. *stochastic gradient descent*, SGD). Bazuje on na założeniu, że estymację gradientu można otrzymać wyciągając średnią gradientu na minipakiecie m przykładów. Kolejne kroki algorytmu można zapisać następująco:

1. Wybierz wartość parametru szybkości uczenia się ϵ_k.
2. Próbkuj minipakiet złożony z m przykładów ze zbioru szkoleniowego.
3. Oblicz estymację gradientu $g = \frac{1}{m} \bigtriangledown \sum_{i=1}^{m} L(x_i, y_i, f)$, gdzie L to funkcja strat na jeden przykład o wejściowej wartości próbki x_i i oczekiwanym wyjściu y_i.

4. Zastosuj aktualizację wartości funkcji celu równą $\epsilon_k g$.

5. Jeżeli kryterium stopu nie zostało spełnione wróć do kroku 2.

Kryterium stopu jest najczęściej określone liczbą iteracji lub satysfakcjonującą wartością funkcji f. Kwestia optymalnego wyboru parametru ϵ_k zależy od problemu i najczęściej stosowane są metody empiryczne, przy czym zazwyczaj ϵ_k maleje wraz ze zbliżaniem się do satysfakcjonującego rozwiązania.

Parametry algorytmów szkoleniowych nazywane są *hiperparametrami*, gdyż nie są wyznaczane bezpośrednio w procedurze uczenia się. Strategia nadawania hiperparametrom wartości początkowych jest silnie dyskutowana w literaturze (zob. [31]) i jej kompleksowy opis wykracza poza zakres tej pracy. Warto jednak wspomnieć o algorytmach z adaptacyjną szybkością uczenia się, gdyż jest to jeden z najtrudniejszych do ustawienia hiperparametrów, a jednocześnie bardzo istotny. Są to m.in.:

- Adaptive Gradient Algorithm (AdaGrad) [18] – wykorzystywany do indywidualnej adaptacji szybkości uczenia się wszystkich parametrów modelu, skalując je odwrotnie proporcjonalnie do pierwiastka kwadratowego sumy wszystkich historycznych kwadratów gradientów.

- Root Mean Square Propagation (RMSProp) [55] – modyfikacja AdaGrad, w której zamiast akumulacji gradientu wykorzystuje się wykładniczo ważoną ruchomą średnią z gradientu.

- Adaptive moments algorithm (Adam) [30] – W porównaniu z RMSProp, Adam poza momentem pierwszego rzędu (tj. średnią) wykorzystuje również moment drugiego rzędu (tj. *wariancję*). Dokładniej rzecz biorąc, w algorytmie liczona jest wykładnicza ruchoma średnia gradientu i kwadrat z gradientu oraz parametry β_1 i β_2, które kontrolują zakres liczenia średnich.

W momencie pisania tej pracy algorytm Adam jest najczęściej rekomendowanym jako domyślna metoda szkolenia głębokich sieci neuronowych dlatego poniżej zamieszczono jej dokładny opis:

1. Wybierz wartość początkową ϵ, β_1, β_2 oraz ustaw początkowe wartości zmiennych momentu 1 i 2 stopnia $s = 0$ i $r = 0$, wartość kroku czasowego $t = 0$ i stałą σ używaną do stabilizacji numerycznej.

2. Próbkuj minipakiet złożony z m przykładów ze zbioru szkoleniowego.
3. Oblicz estymację gradientu $g = \frac{1}{m} \bigtriangledown \sum_{i=1}^{m} L(x_i, y_i, f)$ i zwiększ t o 1
4. aktualizuj estymację pierwszego momentu. $s = \beta_1 s$ + (1-β_1)g
5. aktualizuj estymację drugiego momentu. $r = \beta_2 r$ + (1-β_2)$g \bigodot g$
6. skoryguj obciążenie momentu pierwszego rzędu $s = \frac{s}{1-\beta_1^t}$
7. skoryguj obciążenie momentu drugiego rzędu $r = \frac{r}{1-\beta_2^t}$
8. Zastosuj aktualizację wartości funkcji celu równą -$\epsilon \frac{s}{\sqrt{r}+\sigma}$.
9. Jeżeli kryterium stopu nie zostało spełnione wróć do kroku 2.

Ocena procesu szkolenia się sieci polega na obliczeniu odpowiednich miar i współczynników odzwierciedlających przybliżenie zbioru T lub/i W przez znalezione rozwiązanie. W powszechnym problemie klasyfikacji binarnej, występującym również w tej pracy, w kontekście oceny efektywności algorytmów można wyszczególnić następujące parametry:

- *Fałszywie pozytywna klasyfikacja* (*FP* od ang. *False Positive*) – liczba obserwacji zaklasyfikowanych jako pozytywne, a należących do klasy obserwacji negatywnych.
- *Fałszywie negatywna klasyfikacja* (*FN* od ang. *False Negative*) – liczba obserwacji zaklasyfikowanych jako fałszywie negatywne, a należących do klasy obserwacji pozytywnych.
- *Prawdziwie pozytywna klasyfikacja* (*TP* od ang. *True Positive*) – liczba wyników poprawnie zaklasyfikowanych jako pozytywne.
- *Prawdziwie negatywna klasyfikacja* (*TN* od ang. *True Negative*) – liczba wyników poprawnie zaklasyfikowanych jako negatywne,

oraz miary:

- *Dokładność klasyfikacji* (ang. *Accuracy*) – $ACC = \frac{TP+TN}{TP+TN+FP+FN}$.
- *Czułość klasyfikacji* (ang. *Sensitivity*) – $TPR = \frac{TP}{TP+FN}$.

- *Swoistość klasyfikacji* (ang. *Specificity*) – $TNR = \frac{TN}{TN+FP}$.
- *Precyzja klasyfikacji* (ang. *Precision*) – $PPV = \frac{TP}{TP+FP}$.

W problemach empirycznych dąży się do maksymalizowania TP, TN, ACC, TPR, TNR i PPV oraz minimalizowania FP i FN. Prawidłowe podejście polega również na wybraniu możliwie efektywnej architektury sieci. Problemy z tym związane zostaną omówione w kolejnych podsekcjach.

4.2.1 Problem nadmiernego dopasowania

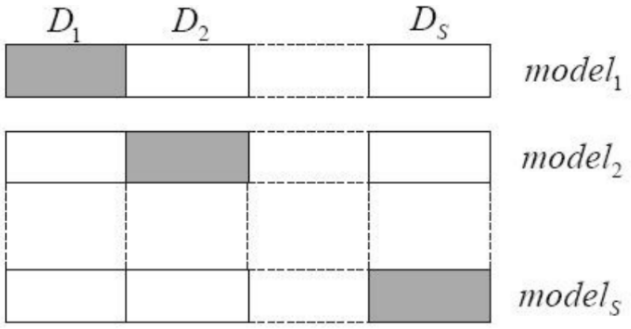
Dążenie do najlepszego możliwego przybliżenia zbioru U wprowadza niepożądane zjawisko zwane *nadmiernym dopasowaniem*. Wtedy to dokładność klasyfikacji zbioru U jest wysoka lub nawet bezbłędna, natomiast znacznie niższa jest dokładność klasyfikacji zbioru testowego T i walidacyjnego W. W praktyce oznacza to, że model staje się mało użyteczny, gdyż wnioskowanie na nowych danych charakteryzuje się niską dokładnością.

Zatem ogólnym dążeniem w procesie uczenia się sieci jest osiągnięcie *maksymalnej generalizacji* klasyfikacji. Sieć o wysokim współczynniku generalizacji lepiej klasyfikuje ogół zadanych wektorów wejściowych niż sieć, która ma niski współczynnik generalizacji i jest nadmiernie dopasowana do zbioru U. Właściwym działaniem jest więc ustalenie maksymalnie ogólnych, dostatecznych warunków poprawnej klasyfikacji, dzięki którym wzrasta prawdopodobieństwo, że przykład z poza zbioru U będzie poprawnie klasyfikowany.

W tym celu można zastosować *metodę oceny krzyżowej* (ang. *cross-validation*). Metoda ta polega na podziale zbioru uczącego na s segmentów D, z których każdy w innej iteracji służy jako zbiór testujący i walidacyjny, a pozostałe segmenty pełnią rolę zbioru uczącego. Podział zobrazowany jest na Rysunku 4.5. Stosując metodę oceny krzyżowej dla różnych modeli sieci można stwierdzić, który z nich spełnia najlepiej kompromis między dobrą klasyfikacją zbioru U i wysoką generalizacją.

Kombinacja predykcji wielu różnych modeli jest bardzo wydajną metodą do polepszenia generalizacji i zmniejszenia błędu klasyfikacji na zbiorach z po za U (zob. [8,10]). Jednak współczesne sieci neuronowe, których przykłady zostały opisane w dalszych sekcjach mogą zawierać miliony parametrów i ich optymalizacja jest wymagająca obliczeniowo, dlatego w praktyce ogranicza się liczbę segmentów obierając $s \in \langle 5, 10 \rangle$.

Innym podejściem zaproponowanym w 2012 w [28] jest technika *dropout*, której



Rysunek 4.5: Reprezentacja graficzna oceny krzyżowej.

główna idea bazuje na zerowaniu wyjścia neuronów sieci z prawdopodobieństwem 0,5 przy każdej iteracji treningu sieci. Neurony, które są w ten sposób tymczasowo dezaktywowane nie mają wpływu w danej iteracji na predykcję sieci i nie są uwzględniane przy wstecznej propagacji gradientu. Podejście to można porównać do treningu w każdej iteracji różnych modeli sieci. Dla przykładu w [34] wykazano, że metoda dropout wymaga jedynie 2 razy więcej iteracji do przybliżenia zbioru U, przy tym uzyskując znacznie lepszą generalizację.

Kluczowym składnikiem potrzebnym do treningu sieci i maksymalizacji generalizacji jest odpowiedni rozmiar zbioru danych. Jest to problem szeroko dyskutowany, gdyż zwłaszcza w danych medycznych istnieje szereg ograniczeń związanych z dostępem i akwizycją odpowiedniego materiału badawczego (np. ograniczenia prawne, związane z prywatnością lub z etyką). W przypadku, gdy zgromadzenie odpowiedniego zbioru danych jest niemożliwe pewnym rozwiązaniem problemu jest zastosowanie metod jego sztucznego powiększania (ang. *data augmentation*).

Dla obrazów stosuje się metody *afinicznych przekształceń* zgodne z definicją algebraiczną:

$$\mathbf{o} \mapsto \mathbf{A}\mathbf{o} + \mathbf{b}, \tag{4.9}$$

gdzie A jest macierzą przekształcenia liniowego, a b wektorem przesunięcia. Jako przykłady takich przekształceń dla dwuwymiarowych obrazów można wymienić:

- *rotację* – obrót obrazu o kąt θ, gdzie:

$$A = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{4.10}$$

- *odbicie lustrzane* – odwrócenie kolejności pikseli w każdym wierszu, gdzie:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.11}$$

- *skalowanie* – zmiana rozmiaru obrazu o S, gdzie:

$$A = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \tag{4.12}$$

- *translację* – przesunięcie punktów obrazu o wektor b.

Analogiczne równania istnieją dla obrazów trójwymiarowych (zob. [27]). W określonych przypadkach używane są również nieafiniczne przekształcenia. Np. w [49] wykorzystano nieliniowe deformacje do powiększenia zbioru 30 obrazów mikroskopowych przedstawiających macierz komórkową i uzyskano znacząco lepsze wyniki detekcji komórek niż istniejące w 2015 algorytmy typu *state-of-the-art*.

W przypadku danych medycznych należy szczególnie zwrócić uwagę, aby powiększony zbiór zawierał dane przypominające w rzeczywistości występujące przypadki np. nieduże obroty występujące u pacjentów skanowanych rezonansem magnetycznym lub niewielkie skalowania rozmiaru kości widocznych w TK. Szeroką dyskusję prowadzi się również na temat wykorzystania sztucznie generowanych zbiorów danych o czym więcej można przeczytać w [40, 60].

4.2.2 Problem redukcji wymiarowości

Duży rozmiar wektora cech prowadzi do problemu nazwanego *przekleństwem wymiarowości* (ang. *curse of dimensionality*). Określenie zostało po raz pierwszy sformułowane przez Richarda Bellmana w latach 50-tych XX wieku. Naukowiec ten podczas swojej pracy obserwował algorytmy doskonale działające w 3 wymiarach, a prezentujące znacząco gorsze wyniki w hiperprzestrzeni (zob. [9]).

Problem przekleństwa wymiarowości ma dwie główne przyczyny: (1) nie wszystkie cechy są jednakowo znaczące w kontekście rozróżnienia danych; (2) w miarę wzrostu rozmiaru przestrzeni cech, liczba obserwacji w zbiorze uczącym potrzebnych do wiarygodnego oszacowania funkcji wyjściowej rośnie wykładniczo.

Problem (1) jest szczególnie istotny w dość prostych algorytmach takich jak np. algorytm K-najbliższych sąsiadów, gdzie do poprawnego działania należy policzyć dystans pomiędzy sąsiednimi obserwacjami. Uwzględniając dużą liczbę nieistotnych cech jako argumenty funkcji dystansu uzyskuje się wyniki utrudniające lub nawet uniemożliwiające poprawną klasyfikację zbioru. Konieczna jest wówczas adaptacja wpływu poszczególnych cech np. poprzez wprowadzenie wektora wag.

Problem (2) ma duże implikacje w praktyce stosowania sieci neuronowych, gdyż posiadanie odpowiednio dużego, ustrukturyzowanego zbioru danych stanowi przeważnie wyzwanie. W poprzedniej podsekcji omówiono możliwość sztucznego powiększania zbioru danych. Inną opcją jest zmniejszenie rozmiaru wektora cech, co może być wykonane na dwa sposoby:

- wybór podzbioru istotnych cech o liczności $n' << n$,
- przekształcenie oryginalnych n zmiennych na nowy zbiór n' cech, gdzie ponownie $n' << n$.

W pierwszym przypadku, wybór podzbioru istotnych cech polega na określeniu minimalnego podzbioru, dla którego rozkład prawdopodobieństwa różnych klas obiektów jest jak najbliższy oryginalnemu rozkładowi uzyskanemu z wykorzystaniem wszystkich cech. Do tych zagadnień wykorzystywane są metody takie jak:

- *miary siły związku* – określające podobieństwo między rozkładami zmiennych losowych. Najczęściej stosowana jest *korelacji Pearsona*, której współczynnik r dla dwóch zmiennych losowych X i Y zapisywany jest następującym wzorem:

$$r_{XY} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}, \tag{4.13}$$

  gdzie X_i, Y_i to wartości kolejnych obserwacji, a $\overline{X}$ i $\overline{X}$ to ich średnie. Innymi słowy tak zapisany współczynnik r_{XY} jest ilorazem *kowariancji* i iloczynu *odchyleń standardowych* zmiennych X i Y.

- *miary entropii względnej* – określające rozbieżności między rozkładami zmiennych losowych. Najczęściej stosowana jest *dywergencja Kullbacka-Leiblera*, której współczynnik d_{KL} dla dwóch rozkładów prawdopodobieństwa p i q zapisany jest wzorem:

$$d_{KL}(p, q) = \sum_i p(i) \log_2 \frac{p(i)}{q(i)} \tag{4.14}$$

- *teoria zbiorów przybliżonych* – wykorzystująca informacje o elementach zbioru i klasyczną teorię zbiorów do porównywania rozkładów (zob. [32]).

Podobieństwo rozkładów ocenione na podstawie powyższych miar daje możliwość skutecznej redukcji wymiarowości z zachowaniem efektywności algorytmu.

W drugim przypadku, gdy zaistnieje potrzeba przekształcenia przestrzeni cech i wyliczenia nowego zbioru, metodą *state-of-the-art* jest algorytm *analizy składowych głównych*) (ang. *Principal Component Analysis*, w skr. PCA) opracowany przez Karla Pearsona w 1901 r (zob. [47]).

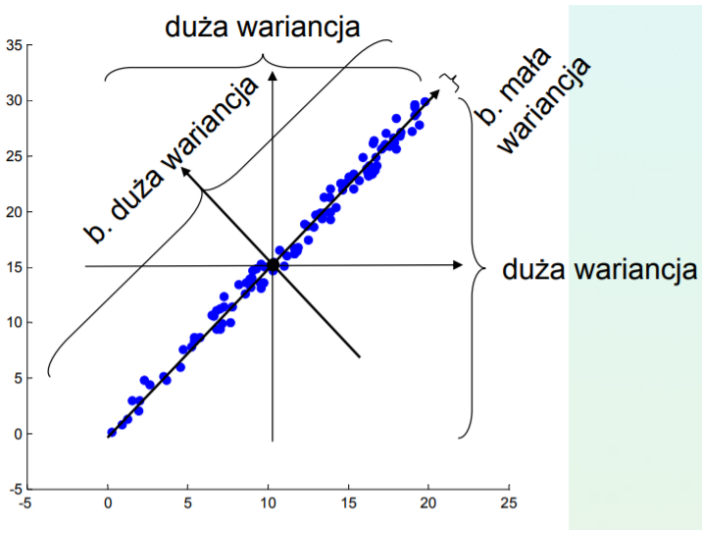
Istotą PCA jest przekształcenie początkowych, skorelowanych cech w nowy zbiór nieskorelowanych zmiennych. Nowe zmienne, tzw. *składowe główne*, powstają z przekształcenia oryginalnych zmiennych skorelowanych, w taki sposób aby w maksymalnym stopniu wyjaśniać całkowitą wariancję w próbie cech oryginalnych. Wariancje składowych głównych są *wartościami własnymi* macierzy kowariancji oryginalnych zmiennych. Innymi słowy są to *wartości skalarne* (pojedyncze liczby) opisujące poziom wariancji w danych w stosunku do zadanych *ortogonalnych wektorów*, tj. gdzie kąt w przestrzeni Euklidesowej pomiędzy wektorami równy jest 90°.

Dla przykładu pierwsza składowa główna redukuje największą część zróżnicowania, druga kolejną, której nie redukowała poprzednia itd. Składowych głównych jest zawsze tyle ile wymiarów danych. Przykład dwuwymiarowy został przedstawiony na Rys. 4.6. Można zaobserwować, że nowa przestrzeń charakteryzuje się znacznie większym zróżnicowaniem poziomu wariancji niż przestrzeń oryginalna. Wybierając zatem tylko pierwszą składową uzyskana zostanie reprezentacja *skomprymowana stratnie* (tzn. taka która nie daje gwarancji odtworzenia oryginalnych wartości), ale za to zachowująca znaczną część wariancji.

Opis szczegółowej procedury PCA wygląda następująco:

1. Oblicz macierz kowariancji: $S_x = X^T X$, gdzie X to macierz danych zawierająca obserwacje w wierszach. Macierz S_x jest symetryczna i pozwala ocenić: (1) wariancje zmiennych poprzez analizę elementów na głównej przekątnej; (2) zależności pomiędzy zmiennymi poprzez analizę elementów poza główną przekątną.

2. Dokonaj diagonalizacji: $S_x = KLK^T$, gdzie L to macierz diagonalna, a K to macierz odwracalna składająca się z wektorów własnych odpowiadających kolejnym wartościom własnym.



Rysunek 4.6: Dwuwymiarowa przestrzeń cech wraz z zaznaczonymi poziomami warianci i składowymi głównymi.

3. Utwórz macierz nowych zmiennych wykonując operację: $Y = XK$

Oprócz PCA, w kontekście redukcji wymiarowości można również wyróżnić następujące algorytmy:

1. *Multidimensional scaling*, MDS – bazuje na wyliczeniu odpowiednio zdefiniowanego dystansu między wartościami cech w oryginalnej przestrzeni i próbie zachowania tego dystansu w zredukowanej wymiarowości (zob. [11]).

2. *T-distributed stochastic neighbor embedding*, t-SNE – stosowany do wizualizacji struktur cech w różnych skalach (zob. [68]).

3. *Isomap* – algorytm nieliniowej redukcji wymiarowości pozwalający na gwarantowane znalezienie globalnego minimum (zob. [67]).

4. *Independent Component Analysis*, ICA – stosowany do znalezienia reprezentacji danych składających się z niezależnych elementów (zob. [36]).

5. *Latent Semantic Analysis*, LSA – stosowany głównie w problemach dotyczących przetwarzania języka naturalnego, zwłaszcza do znalezienia kontekstu użycia danego słowa poprzez analizę statystyczną dużych bloków tekstów (zob. [71]).

6. *Self Organizing Maps*, SOM – bazuje na zachowaniu własności topologicznych przestrzeni cech oryginalnych w zredukowanej wymiarowości (zob. [69]).

Zarówno przekleństwo wymiarowości, jak i wcześniej opisany problem nadmiernego dopasowania są zjawiskami bardzo często występującymi w praktycznych zastosowaniach algorytmów głębokiego uczenia się. Nie wyczerpują one jednak tematu. Szereg kolejnych wyzwań i rozwiązań został opisany w kolejnej sekcji przy okazji przedstawienia współczesnych topologii.

4.3 Przykłady współczesnych topologii

Współczesne sieci konwolucyjne służące do klasyfikacji obrazów, podobnie jak u protoplasty tj. architektury LeNet, składają się z dwóch głównych części: ekstraktora cech i klasyfikatora. W porównaniu jednak do pierwszych topologii, różnorodność warstw uległa zwiększeniu. Do najczęściej stosowanych komponentów współczesnych sieci konwolucyjnych można zaliczyć:

- *warstwy konwolucyjne* – grupują filtry używane do ekstrakcji cech obrazowych różnego poziomu (np. krawędzie, skupiska, obiekty);

- *warstwy max-pool* – stosowane do nieliniowej redukcji wymiarowości. Z obszaru $n \times n$ w obrazie wyliczana jest największa wartość [2] $I(x, y)$. W praktyce zazwyczaj n=2, podobnie jak wartość *kroku* tj. odstępu pomiędzy kolejnymi próbkowaniami funkcji I.

- *warstwy aktywacji* – zawierające funkcje aktywacji neuronów. We współczesnych architekturach zamiast funkcji sigmoidalnych, czy progowych stosuje się najczęściej funkcję ReLU:

$$f(x) = \max(0, x). \tag{4.15}$$

  Jest ona mniej wymagająca obliczeniowo niż funkcja sigmoid, a dla współczesnych topologii o dużej liczbie parametrów testy empiryczne wykazały, że równie dobrze wpływa na uzyskiwane wyniki dokładności (zob. [34]).

- *warstwy fully connected*, FC – składające się z neuronów, z których każdy jest połączony z każdym neuronem kolejnej warstwy. Występują zazwyczaj w ostatniej części sieci konwolucyjnych tj. w klasyfikatorze.

[2] W uzasadnionych przypadkach stosuje się również obliczanie średniej, normy $L2 = \|x\| = \sqrt{\sum_{k=1}^{n} |x_k|^2}$ i innych współczynników.

- *warstwy normalizacji*[3] – normalizujące wyjście poprzedzającej warstwy do pewnego określonego przedziału np. (-1,1). Wykorzystywane są w celu zrównoważenia poziomu wpływu poszczególnych neuronów na wynik końcowy. Bardzo często stosuje się w nich operację *Local Response Normalization* daną wzorem:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^i_{x,y})^2 \right)^{\beta}, \tag{4.16}$$

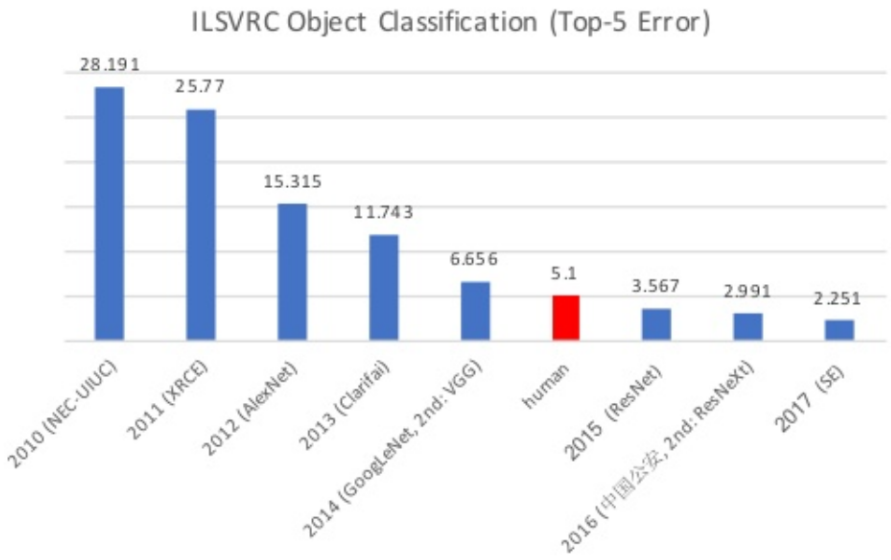
  gdzie $a^i_{x,y}$ jest aktywacją danego neuronu, a k, α, n i β to stałe dobierane empirycznie na podstawie zbioru walidacyjnego (za [34]).

Wprowadzone w ostatnich latach dedykowanych dla głębokiego uczenia się rozwiązań zarówno w warstwie sprzętowej jak i oprogramowania umożliwiło tworzenie architektur o wysokim stopniu komplikacji. W głównej mierze rozwiązania te umożliwiły optymalizację fazy szkolenia omówionej w poprzedniej sekcji jak również *fazy wnioskowania*, gdzie wytrenowana sieć użyta jest do przetwarzania kolejnych obserwacji. W warstwie sprzętowej wyróżnić można dedykowane akceleratory do operacji macierzowych takie jak [42], *NVIDIA Tensor Processing Unit* (w skr. *TPU*) czy [35], firmy Intel oraz rozwiązania takie jak [4] minimalizujące czasy przetwarzania sygnału przez wytrenowaną sieć np. stosując optymalizację reprezentacji liczb zmiennoprzecinkowych lub przetwarzanych macierzy.

W warstwie oprogramowania należy wspomnieć o dynamicznym rozwoju framework'ów do zoptymalizowanych obliczeń z udziałem głębokich sieci neuronowych takich jak: Caffe, Caffe2, TensorFlow, Theano, PyTorch czy MXNet. Porównanie powyższych framework'ów osadzone w kontekście aplikacji medycznych można znaleźć w [19].

Postęp w rozwoju współczesnych sieci konwolucyjnych doskonale odzwierciedla progres w rezultatach konkursu ImageNet Large Scale Visual Recognition Competition (w skr. ILSVRC) przedstawiony na Rys. 4.7. Prezentowane wyniki dotyczą wartości błędu klasyfikacji na zbiorze danych [16] o nazwie ImageNet, uzyskiwanego w kolejnych latach przez zwycięskie algorytmy biorące udział w konkursie. *Błąd top-n* należy rozumieć jako zdarzenie, w którym dla danego obrazka w n wskazanych przez algorytm najbardziej prawdopodobnych etykietach nie było poprawnej. W zakresie zmniejszenia wartości błędu top-n, znaczący progres dokonał się w 2012 roku, gdzie błąd top-5 zmalał o 10.4 punktów procentowych, co było rezultatem działania nowej architektury

[3]Coraz rzadziej stosowane w praktyce z uwagi na znikomy wpływ na rezultaty.



Rysunek 4.7: Błąd top-5 klasyfikacji obiektów w kolejnych latach uzyskiwany przez zwycięzców konkursu ILSVRC

sieci konwolucyjnej. W kolejnych latach sieci konwolucyjne deklasowały inne podejścia doprowadzając w 2015 roku do spadku błędu top-5 do poziomu 3,5%, co jest uznawane za poziom lepszy niż możliwości ludzkiej klasyfikacji zbioru ImageNet. Lata 2016 i 2017 to intensywne prace nad synergią i złożeniami różnego rodzaju modeli, które w konsekwencji doprowadziły do obniżenia wartości błędu top-5 do poziomu 2,2%. W kolejnych podsekcjach zostanie dokładniej omówiona ewolucja zwycięskich architektur z konkursu ILSVRC.

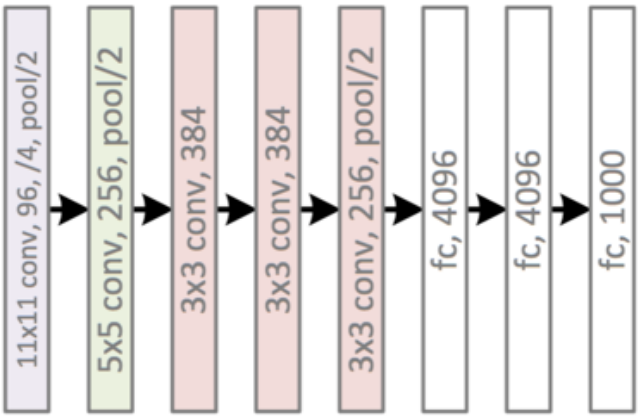
4.3.1 AlexNet

Sieć AlexNet, której nazwa pochodzi od imienia głównego twórcy tej architektury Alexa Krizhevsky, zawiera blisko 60 milionów parametrów i 650 tysięcy neuronów. Architekturę zaprezentowano na Rys. 4.8

W skład topologii wchodzi pięć warstw konwolucyjnych i trzy typu fully-connected. Po pierwszej, drugiej i piątej warstwie konwolucyjnej występują operacje typu max-pool z maską o wymiarach 2×2 [4].

Pierwsza warstwa konwolucyjna przyjmuje na wejściu dane o wymiarze 227×227×3,

[4]Autorzy pracy podają też przykłady użycia masek o wymiarze 2×3, które nakładają się w przestrzeni funkcji obrazowej. Nie znalazły one jednak miejsca w finalnej implementacji.



Rysunek 4.8: Topologia architektury AlexNet.

na których wykonywana jest operacja splotu z 96 filtrami z maską o wymiarach 11×11×3 i krokiem 4. W rezultacie (uwzględniając również operację max-pool) objętość wynikowa przekazywana do kolejnej warstwy ma wymiar 27×27×96. W drugiej warstwie konwolucyjnej wykonywana jest operacja splotu z 256 filtrami z maską o wymiarach 5×5×96. Wymiar objętości wynikowej zostaje ponownie zredukowany poprzez operacje max-pool do 13×13×256. Kolejne 3 warstwy konwolucyjne są połączone bezpośrednio ze sobą. Trzecia warstwa zawiera 384 filtry z maską o wymiarze 3×3×256, w skład czwartej wchodzą 384 filtry z maską o wymiarze 3×3×384, a w piątej znajduje się 256 filtrów również z maską o wymiarze 3×3×384. Końcowe dwie warstwy typu FC zawierają po 4096 neuronów, a ostatnia zawiera tyle neuronów ile klas występuje w ostatecznym podziale – w oryginalnej pracy było to 1000 (por. [34]).

W celu lepszego zrozumienia przetwarzania sygnału wejściowego przez sieć poniżej przedstawiono przykład algorytmu wykorzystywanego dla pierwszej warstwy konwolucyjnej opisywanej topologii:

1. Z danych wejściowych o wymiarze [227×227×3] wybierany jest co czwarty blok o wymiarach [11×11×3] (zarówno wzdłuż wysokości jak i szerokości). W rezultacie otrzymywanych jest 217 punktów w każdym rzędzie i w kolumnie, w których mieści się [55×55] tj. 3025 bloków.

2. Zarówno 11×11×3 = 363 wagi znajdujące się w 96 filtrach jak i wartości 363 punktów obrazowych znajdujących się 3025 blokach są przedstawiane w postaci macierzy A o wymiarach [96×363] i B o wymiarach [363×3025].

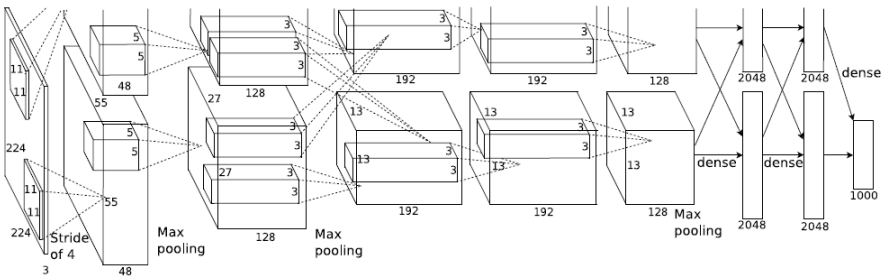
3. Następnie liczony jest iloczyn skalarny w postaci $\mathrm{A}^{\intercal}\mathrm{B} = C$, gdzie nowa, wyjściowa

macierz C ma wymiar [96×3025].

4. Rezultat w postaci macierzy C ponownie przewymiarowany jest na postać [55×55×96].

W architekturze jako funkcję aktywacji neuronów wykorzystano ReLU, co znacząco przyspieszyło trening sieci. Dla przykładu uzyskano 6-krotne przyspieszenie treningu dla danych [33], CIFAR-10 w stosunku do tej samej topologii wykorzystującej sigmoidalną funkcję aktywacji. Ponieważ funkcja ReLU nie posiada górnego ograniczenia neurony teoretycznie mogą posiadać nieograniczone wartości funkcji aktywacji. W celu polepszenia kontrastu pomiędzy neuronami i wydobycia tych, które na tle innych się wyróżniają, zastosowano normalizację zgodną ze wzorem 4.16. W wyniku czego uzyskano redukcję błędu klasyfikacji top-5 o wartość 1,2 punktu procentowego.

W kontekście zwiększenia efektywności treningu zastosowano również powiększenie rozmiaru danych poprzez rotacje i modyfikacje funkcji obrazowej z wykorzystaniem czynników głównych (zob. [34]) zmniejszając błąd top-1 o 1%. Zastosowano też technikę dropout opisaną w 4.2.1. Ostatecznie wprowadzono także trening z wykorzystaniem wielu GPU (zob. Rys. 4.9).



Rysunek 4.9: Topologia architektury AlexNet z podziałem na dwa akceleratory GPU.

Topologia z podziałem na 2 karty zwiększyła dwukrotnie sumaryczną pamięć i pozwoliła na kolokację parametrów sieci.

Praca Alexa Krizhevsky, Ilya Sutskever i Geoffrey'a Hinton zapoczątkowała wzrost zainteresowania technikami głębokiego uczenia się, co doprowadziło do publikacji kolejnych podobnych architektur. Do najbardziej znanych należą ZFNet z 2013 roku [75], gdzie m.in. zastosowano zmniejszenie wymiaru maski stosowanego w filtrach pierwszej warstwy konwolucyjnej do 7×7 oraz VGGNet [59] z 2014 roku, gdzie zastosowano większą liczbę warstw konwolucyjnych z mniejszym wymiarem maski. Również w 2014 roku zaprezentowano innowacyjną koncepcję modułów sieci konwolucyjnych, co do-

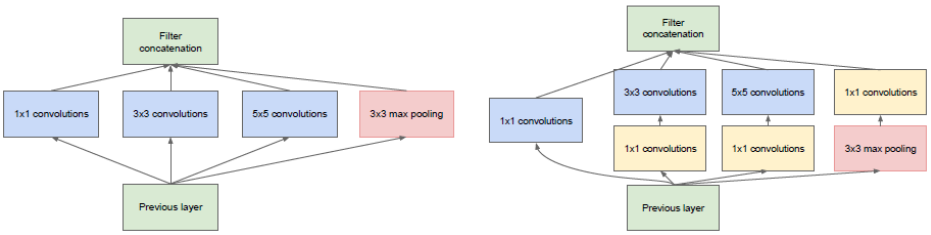
prowadziło do zwycięstwa w ILSVRC. Idea ta została dokładniej opisana w kolejnej podsekcji.

4.3.2 GoogLeNet

Architekturę o nazwie GoogLeNet zaprezentowano w 2014 r. w pracy [63]. Nazwa architektury pochodzi od nazwy zwycięskiego zespołu startującego w ILSVRC 2014, składającego się z pracowników firmy Google. Oryginalnie topologia składała się z 22 warstw i zawierała około 5 mln parametrów (12 razy mniej niż w przypadku sieci AlexNet).

Redukcję liczby parametrów przy jednoczesnym podwyższeniu dokładności klasyfikacji obiektów udało się uzyskać poprzez poszukiwania konstrukcji optymalnych lokalnych topologii i ich połączeń. Mianowicie, wiadomo że duża część funkcji aktywacji neuronów przyjmuje wartość 0 lub jest redundantna z powodu wysokiej korelacji między sobą (zob. [7]). Matematyka dotycząca przetwarzania *macierzy rzadkich*, tj. gdzie przeważająca liczba elementów przyjmuje wartość 0, jest dobrze znana (zob. np. [46]). Jednak implementacje bibliotek do obliczeń związanych z algebrą liniową są zoptymalizowane pod kątem *macierzy gęstych*, gdzie przeważająca liczba elementów przyjmuje wartości różne od 0 (zob. [34, 61]).

Ideą modułu incepcji zaproponowanego przez twórców GoogLeNet jest aproksymacja rzadkich macierzy z użyciem komponentów o gęstej strukturze. Takie komponenty nazwano *modułami incepcji* (ang. *inception modules*), a ich przykłady pokazano na Rys. 4.10.

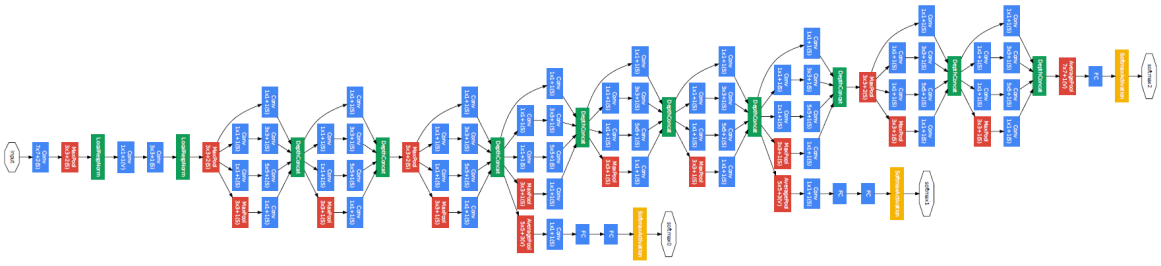


Rysunek 4.10: Topologia architektury AlexNet z podziałem na dwa akceleratory GPU.

(a) przedstawia naiwną formę modułu incepcji, gdzie grupowane są operacje filtrów z maską o wymiarach 5×5, 3×3, 1×1 oraz operacja max-pool. (b) prezentuje koncepcję zoptymalizowaną obliczeniowo gdzie filtry z maską 1×1 służą do redukcji wymiarowo-

ści i używane są bezpośrednio przed splotami z bardziej wymagającymi obliczeniowo splotami 5×5 i 3×3.

Przy pomocy złożenia różnego rodzaju modułów incepcji otrzymano topologię zaprezentowaną na Rys. 4.11



Rysunek 4.11: Topologia architektury GoogleNet

Dokładne zestawienie parametrów znajduje się w Tabeli 4.1

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Tabela 4.1: Parametry architektury GoogleNet

Ważną cechą sieci GoogleNet jest brak warstw typu FC na zakończeniu, gdzie w przypadku sieci AlexNet znajduje się około 90% parametrów. Końcowe wnioskowanie jest realizowane na podstawie wartości średniej z dwuwymiarowych map cech.

Dla lepszego zrozumienia idei redukcji wymiarowości realizowanej przez moduły incepcji, podobnie jak w przypadku sieci AlexNet, przeanalizowane zostanie działanie

pierwszego modułu w topologii z Rys. 4.11. Moduł zawiera 128 filtrów z maskami o wymiarach 3×3 i 32 filtry z maskami o wymiarach 5×5. Dane na wejściu modułu mają 192 kanały (zob. Tabela 4.1). Dla przykładu, rząd wielkości obliczeń operacji splotów 32 filtrów 5×5 wynosi 25×32×192=153 600 i dalej wzrastałby z głębokością sieci. W celu zapobiegnięcia nadmiarowi obliczeń stosowana jest redukcja z użyciem 16 filtrów z maską o wymiarach 1×1. W efekcie rząd wielkości obliczeń spada do 16×192 + 25×32×16=15 876, co pozwala na dalsze budowanie wielowarstwowych struktur.

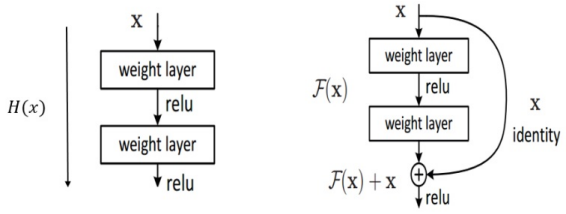
Topologia GoogLeNet jest wciąż rozwijana. Po pierwszej prezentacji pojawiły się kolejne modernizacje wprowadzające dodatkowe faktoryzacje modułów jak w Inception-v2 lub normalizacje wartości wynikowych poszczególnych warstw jak w Inception-v3. Obie sieci zostały przedstawione w [64]. Kolejny innowacyjny pomysł, bazujący na dodatkowych połączeniach między blokami, został wprowadzony w 2015 roku w sieci ResNet opisanej w kolejnej podsekcji.

4.3.3 ResNet

Jednym z najbardziej oczywistych pomysłów na polepszenie dokładności działania sieci neuronowych jest zwiększenie liczby warstw. Jednak wraz ze wzrostem liczby warstw, trening takich architektur z użyciem tradycyjnych metod gradientowych (takich jak algorytm wstecznej propagacji błędu) staje się mniej wydajny. Problem wynika z faktu, że zmiana wartości sygnału na wyjściu sieci w odpowiedzi na sygnał wejściowy jest mniejsza wraz ze wzrostem liczby warstw. W takiej sytuacji gradient wyliczany na podstawie sygnału będącego różnicą pomiędzy sygnałem wejściowym a wejściowym może przyjmować wartości bliskie 0 uniemożliwiając progres uczenia się. Problem zanikającego gradientu (ang. *vanishing gradient problem*) rozwiązywany jest poprzez zastosowanie normalizacji oraz nieliniowych funkcji aktywacji. Dzięki tym mechanizmom algorytm treningu głębokich sieci neuronowych w większej liczbie przypadków zbiega do użytecznego minimum lokalnego.

W momencie znalezienia takiego minimum dodanie kolejnych warstw i parametrów sieci jest redundantne, a nawet prowadzi do pogorszenia wyników treningu sieci. Zjawisko to nosi nazwę *degradacji treningu* (ang. *degradation problem*). Twórcy architektury ResNet, przedstawionej w [25], zaproponowali rozwiązanie tego problemu poprzez implementację bloków rezydualnych (ang. *Residuum Units*) zawierających dodatkowe, skrótowe połączenia (ang. *skip conections*) pomiędzy wejściem a wyjściem

bloków. Porównanie schematów funkcjonalnych nowych bloków i wcześniej istniejącego rozwiązania stosowanego np. w AlexNet zostało przedstawione na Rys. 4.12.



Rysunek 4.12: Schemat funkcjonalny pojedynczego bloku w architekturze ResNet.

Ogólną postać równania bloku rezydualnego można zapisać następująco:

$$\begin{aligned} y_l &= h(x_l) + F(x_l, W_l), \\ x_{l+1} &= f(y_l), \end{aligned} \tag{4.17}$$

gdzie x_l i x_{l+1} stanowią sygnał wejściowy i wyjściowy l-tego bloku. F stanowi funkcję rezydualną optymalizowaną podczas treningu sieci, $h(x_l)$ stanowi funkcję przekształcenia sygnału x_l przekazywanego skrótowym połączeniem, f jest funkcją ReLU, a W stanowi macierz wag.

Funkcja $h(x_l)$ jest funkcją tożsamościową, a zatem $h(x_l) = x_l$. Żeby uzasadnić ten wybór należy rozważyć propagację gradientu wewnątrz sieci składającej się z bloków rezydualnych. Dla każdego L-tego bloku zachodzi równanie:

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, W_i) \tag{4.18}$$

Korzystając z reguły łańcuchowej można zapisać równanie na gradient funkcji kosztu ε:

$$\frac{\partial \varepsilon}{\partial x} = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F(x_i, W_i)\right) \tag{4.19}$$

z czego wynika, że gradient może być podzielony na dwie addytywne składowe: (1) $w = \frac{\partial \varepsilon}{\partial x_L}$ propagowaną bez wpływu na warstwy zawierające wagi i (2) $\lambda = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F(x_i, W_i)$ propagowaną przez nie.

Przykład propagacji gradientu w sieci składającej się z trzech bloków wygląda zatem następująco:

$$\frac{\partial \varepsilon}{\partial x_0} = \frac{\partial \varepsilon}{\partial x_3} * (w_2 + \lambda_2) * (w_1 + \lambda_1) * (w_0 + \lambda_0) \tag{4.20}$$

Wartości w są zazwyczaj znormalizowane do przedziału (-1;1) można więc rozważyć 4 istotne przypadki równania 4.20:

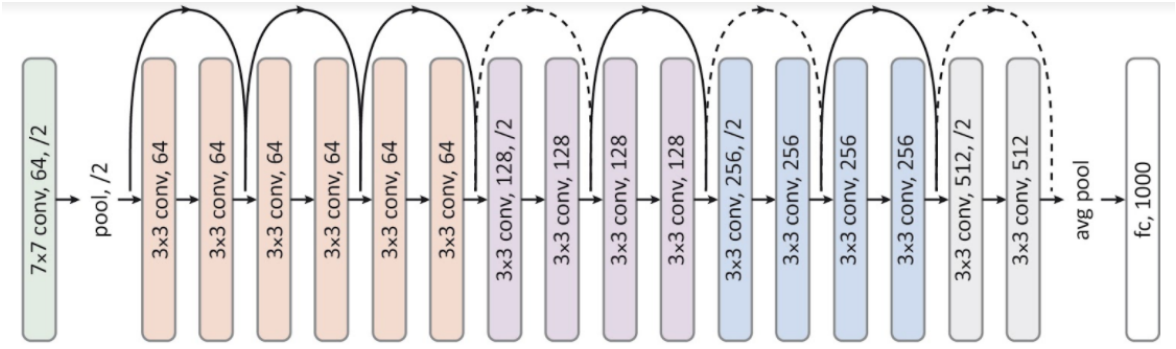
1. $\lambda = 0$ – nie ma skrótowych połączeń, co odpowiada płaskiej strukturze sieci. Ponieważ wartości w są z przedziału (-1;1) dodawanie kolejnych warstw wzmacnia wcześniej omówiony efekt zanikającego gradientu

2. $\lambda > 1$ – z każdą warstwą, sumaryczna wartość gradientu zwiększa się inkrementalni, co nazywane jest *problemem eksplozji gradientu* (ang. *exploding gradient problem*).

3. $\lambda < 1$ – przy założeniu, że $w + \lambda < 1$, dla sieci składających się z wielu warstw występuje problem zaniku gradientu, jak w przypadku 1. Natomiast, gdy $w + \lambda > 1$ podobnie jak w przypadku 2 może występować problem eksplozji gradientu

4. $\lambda = 1$ – wartości w są inkrementowane dokładnie o 1, co eliminuje problemy podane w przypadkach 1, 2 i 3 i stanowi uzasadnienie dla wyboru funkcji tożsamościowej $h(x_l)$ w architekturze ResNet.

Dokładny opis matematyczny funkcjonowania bloków rezydualnych wraz z dowodami znajduje się w [26]. Przykład topologii sieci składającej się z 8 bloków i łącznie 18 warstw tzw. ResNet-18, przedstawiono na Rys. 4.13.



Rysunek 4.13: Topologia architektury ResNet-18.

Pierwsza warstwa konwolucyjna zawiera filtry z maską o wymiarach 7×7. W kolejnych zastosowano wymiar 3×3. Zastosowanie mniejszych wymiarów masek niż w AlexNet oraz podobnie jak w przypadku sieci GoogLeNet wyliczenie na końcu wartości średniej z dwuwymiarowych map cech zredukowało liczbę parametrów.

Architektura ResNet-18 jest najmniejszą z pojawiających się w literaturze przykładów tego typu. W praktyce, z powodzeniem wykorzystywano topologie składające się nawet z 1202 warstw (zob. [25]). W 2016 roku zaprezentowano w [62] hybrydę sieci GoogleNet i ResNet. Pracowano również nad bardziej złożonymi blokami, co w konsekwencji doprowadziło w 2017 roku do zaprezentowania architektury ResNetX w [74], która w wielu testach klasyfikacji różnych zbiorów okazała się być lepsza niż poprzednicy. Przegląd dotyczący historii tych prac można znaleźć w [5].

Sieć ResNet i jej warianty dla wielu testowych zbiorów danych takich jak ImageNet, CIFAR czy COCO [39] osiągnęły dokładność klasyfikacji porównywalną z możliwościami ludzkiego obserwatora. Dalszy progres był możliwy m.in. dzięki zastosowaniu synergii wielu modeli, co zostało opisane w kolejnej podsekcji.

4.3.4 Złożenia

Uczenie złożeń sieci (ang. *esemble learning*) polega na wykorzystywaniu kilku modeli bazowych i wybranej metody ich synergii. W kontekście głębokiego uczenia się stosowane są różne metody kombinacji modeli bazowych (zob. [29]). Jako często stosowane przykłady można podać: uśrednianie, głosowanie, klasyfikacja Bayesa, generalizację stosów. Zostaną one kolejno omówione:

Uśrednianie

Uśrednianie jest prostą metodą kombinacji wyników predykcji. Najczęściej stosowane jest uśrednienie bez wag, gdzie suma wyników predykcji modeli bazowych podzielona jest przez ich liczbę. Uśredniać można bezpośrednio wyniki ostatecznej klasyfikacji jak również prawdopodobieństwa przynależności do odpowiednich klas, które są np. wynikiem *funkcji softmax*:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \tag{4.21}$$

używanej często bezpośrednio przed ostatnią warstwą sieci neuronowych dla z sygnałów wejściowych i j wyjściowych.

Główną zaletą uśredniania jest redukcja wariancji. Jest ona tym większa im bardziej nieskorelowane są wyniki predykcji modeli bazowych. Pomimo prostoty, tego rodzaju koncepcja odnosiła już sukcesy m.in. w lasach losowych (zob. [10]).

Zastosowanie uśredniania przy silnie odstających od średniej najgorszych predyk-

cjach znacząco obniża dokładność całego złożenia. Dlatego przy tak nieheterogenicznych modelach bazowych dających bardzo różne wyniki poszukiwane są inne metody.

Głosowanie

W głosowaniu stosuje się mechanizm zliczania przewidzianych przez modele bazowe etykiet. Etykieta, która została wybrana przez największą liczbę modeli bazowych jest obierana jako wynik ostatecznej predykcji. Jest to tzw. *głosowanie większościowe*.

W porównaniu do uśredniania, głosowanie jest mniej czułe na predykcje pojedynczych modeli. Wykorzystuje jednak jedynie informacje o przewidzianych etykietach, co utrudnia konstrukcje bardziej wyszukanych rozwiązań.

Klasyfikacja Bayesa

W przypadku tej metody, każdy model bazowy j postrzegany jest jako hipoteza h_j. Każda z hipotez posiada wagę proporcjonalną do prawdopodobieństwa zdarzenia, w którym dany zbiór trenujący zostałby wybrany z ogółu danych gdyby dana hipoteza była prawdziwa. Jest to tzw. optymalna klasyfikacja Bayesa, którą można zapisać następującym równaniem:

$$y = argmax_{c_j \in C} \sum_{h_i \in H} P(c_j \mid h_i) P(T \mid h_i) P(h_i) \tag{4.22}$$

gdzie y to przewidziana etykieta, C jest zbiorem wszystkich możliwych klas, H to przestrzeń hipotez, a T to zbiór danych trenujących.

W praktyce z uwagi na dużą złożoność obliczeniową nie stosuje się optymalnej klasyfikację Bayesa, a jedynie aproksymacje tej metody np.: BPA (od ang. *Bayesian parameter averaging*) [15], BMA (od ang. *Bayesian model averaging*) [52], czy też BMC (od ang. *Bayesian model combination*) [43].

Generalizacja stosów

Idea generalizacji stosów oryginalnie została zaproponowana w [72]. Wykorzystana została koncepcja *meta-uczenia*, a zatem konstrukcja nadrzędnego klasyfikatora, którego zadaniem jest wybór optymalnego wektora wag a dla stosu s predykcji dla danych

x:

$$s(x) = \sum_{i=1}^{m} a_i s_i(x) \tag{4.23}$$

W praktyce predykcje z modeli bazowych składowane są na stosie, a następnie klasyfikator nadrzędny wykorzystuje je jako dane do treningu poprawnych wartości a wykorzystując jako odniesienie znane, poprawne etykiety.

4.4 Zastosowania w medycynie

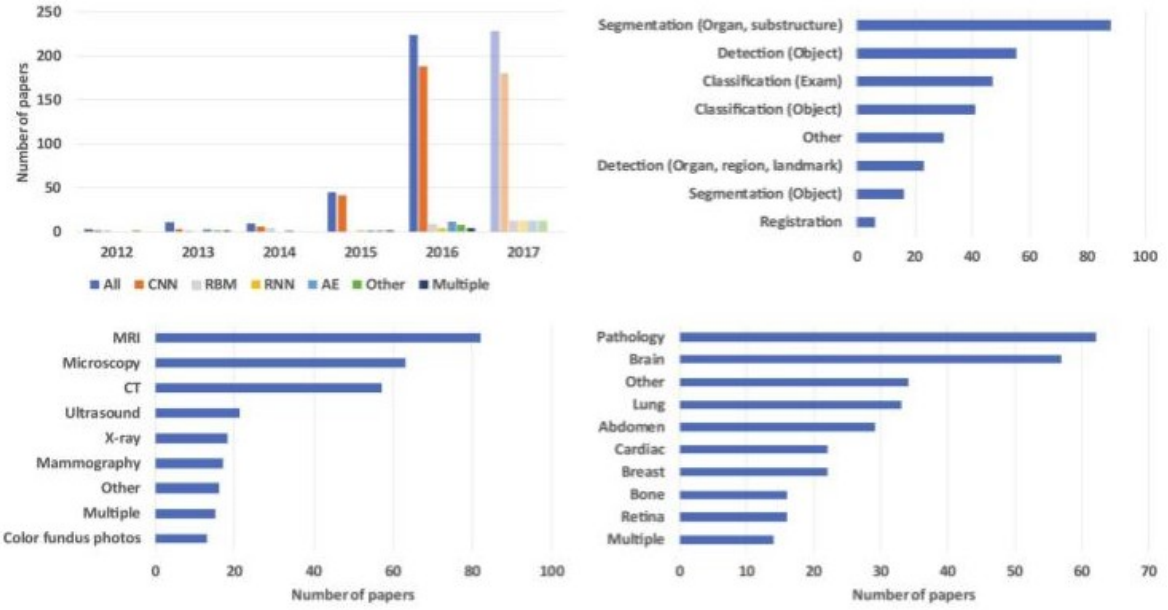
W 1994 roku ukazała się pierwsza praca, która w praktyce wykorzystywała mechanizmy związane z głębokim uczeniem się do przetwarzania obrazów medycznych (zob. [76]). Użyte wówczas sieci nazywano sieciami typu *shift-invariant*. Zastosowanie ich pozwoliło na eliminacje 55% FP otrzymywanych przy wcześniejszych metodach stosowanych do detekcji skupisk mikro-zwapnień w mammografach *Shift-invariant* oznaczało, że przesunięcie obrazu wejściowego nie powodowało zmian w klasyfikacji, co jest istotną wartością dodaną, z uwagi na specyfikę implementacji toru akwizycji danych w praktyce radiologicznej.

Po roku 2012 nastąpił znaczący wzrost zainteresowania metodami głębokiego uczenia się w medycynie. Obrazuje to praca [40] z 2017 roku, w której przytoczono statystyki medycznych publikacji zawierających słowa kluczowe związane z deep learning. Wybrane dane przedstawiono a Rys. 4.14.

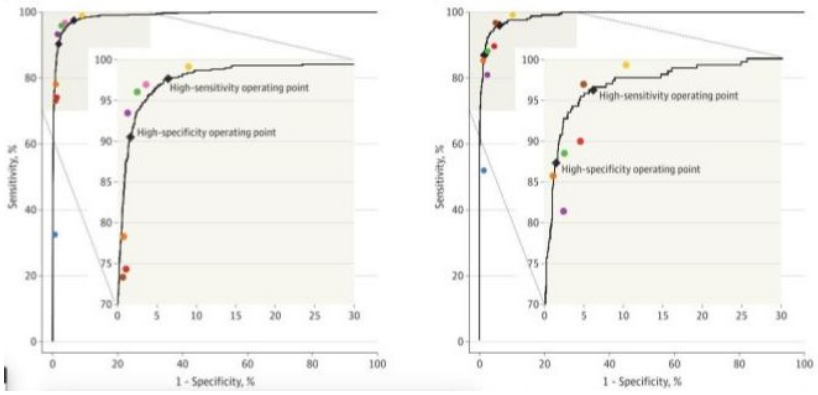
Widoczny wzrost liczby publikacji nastąpił począwszy od 2015, co związane było z kilkuletnią adaptacją nowych metod w dziedzinie przetwarzania obrazów medycznych i gromadzeniem odpowiednich zbiorów danych. Lata 2016 i 2017 były pod pewnym względem przełomowe gdyż pojawiało się coraz więcej prac naukowych, w których przedstawiano rezultaty dokładności klasyfikacji na poziomie dorównującym ekspertom dziedzinowym.

Dla przykładu, w Listopadzie 2016 ukazała się praca [24] grupy Google Research z Mountain View w Kalifornii, gdzie zastosowano sieć GoogLeNet w wersji inception-v3 do zautomatyzowanej detekcji retinopatii cukrzycowej i cukrzycowego obrzęku plamki w obrazach dna oka. Wyniki porównano z panelem składającym się z 7 ekspertów, okulistów. Porównanie przedstawiono na Rys. 4.15.

Na wykresach dla dwóch zadań klasyfikacyjnych umieszczono krzywe reprezentu-



Rysunek 4.14: Statystyki dotyczące publikacji medycznych zawierających słowa kluczowe związane z głębokim uczeniem się.

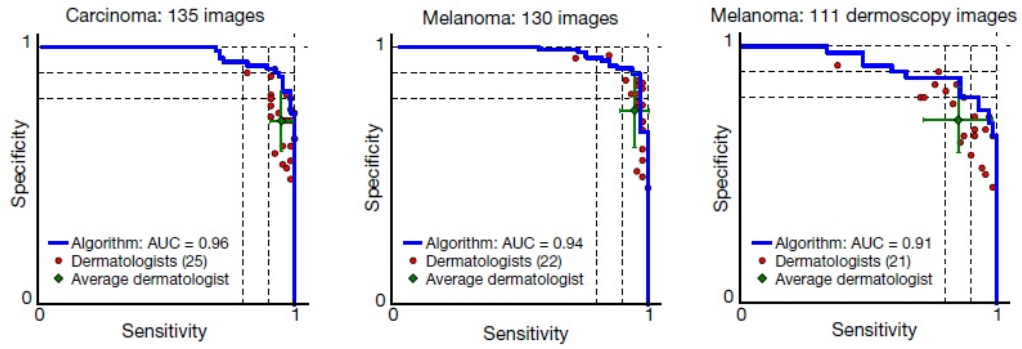


Rysunek 4.15: Porównanie automatycznej klasyfikacji retinopatii cukrzycowej i cukrzycowego obrzęku plamki z oceną panelu ekspertów.

jące zależność swoistości od czułości dla algorytmu automatycznego oraz 7 punktów oznaczających wynik oceny każdego z okulistów. Ogółem mniej niż połowa ekspertów uzyskała lepszy wynik niż algorytm sztucznej inteligencji.

Kolejna ciekawa praca tj. [21], pojawiła się w czasopiśmie Nature w styczniu 2017 roku i traktowała o automatycznej detekcji raka skóry na zdjęciach. Autorzy wykorzystali dane składające się z 129.450 obrazów klinicznych, na których zobrazowano 2.032 różne schorzenia skóry. Ponownie do klasyfikacji wykorzystano sieć GoogleNet w wersji

inception-v3. Wyniki klasyfikacji automatycznej porównano z oceną przeprowadzoną przez 21 certyfikowanych dermatologów. Przykład porównania zaprezentowano na Rys. 4.16.



Rysunek 4.16: Porównanie automatycznej klasyfikacji 3 chorób skóry z oceną ekspertów dermatologów.

Wykres przedstawia zależność czułości od swoistości. Czerwonymi punktami oznaczono wynik oceny poszczególnych ekspertów, a zielonym krzyżykiem wynik uśredniony. W każdym przypadku średnia ocena była gorsza od automatycznej klasyfikacji.

Obrazy medyczne nie są jedynymi danymi, które z powodzeniem są przetwarzane za pomocą metod głębokiego uczenia się. W lipcu 2017, przez grupę ze Stanford University została opublikowana praca [48] dotycząca klasyfikacji arytmii na podstawie szeregów czasowych zapisanych na elektrokardiogramach. Autorzy wykorzystali dane z 64.121 elektrokardiogramów, próbkowanych z częstotliwością 200 Hz, pochodzących od 29.163 pacjentów. Zaprojektowano dedykowaną, 34-warstwową sieć konwolucyjną do detekcji 12 różnych dysfunkcji pracy serca, pracy prawidłowej i szumów (łącznie 14 klas). Wyniki klasyfikacji porównano z oceną prowadzoną przez 3 kardiologów. Średnia dokładność klasyfikacji automatycznej wyniosła 80%, natomiast manualnej 72%.

Podobnych przykładów zostało opublikowanych dużo więcej. Architektura AlexNet z sukcesem była użyta do detekcji polipów w kolonoskopii w [65]. Sieć ResNet sprawdziła się w badaniach zrealizowanych w [20] w Mayo Clinic Rotschester. Dotyczyły one radio-genomiki i rozróżnienia zmian w mózgu bez konieczności biopsji. Złożenia natomiast z sukcesem zostały zaaplikowany w pracach dotyczących detekcji raka płuc, gdzie modele bazowe analizowały różne skale problemu (zob. [17]). W wielu pracach raportuje się dokładność klasyfikacji automatycznej znacząco przewyższające możliwości dziedzinowych ekspertów np. [14, 22, 54, 70].

Powyższe przykłady pokazują, że dla szczególnych przypadków pewien element pracy eksperta zajmującego się danymi medycznymi (np. radiologa) może być z sukcesem wspomagany (lub nawet zastąpiony) przez algorytmy głębokiego uczenia się. Należy jednak podkreślić, że jest również szereg problemów wiążących się z wykorzystaniem tego rodzaju sztucznej inteligencji w medycynie. Do najważniejszych należą:

1. Gromadzenie dużych zbiorów danych z odpowiednimi etykietami.
2. Wykorzystanie heterogenicznych danych pochodzących np. z wielu urządzeń lub modalności.
3. Kalibracja i szacowanie niepewności wyników modeli.
4. Unifikacja modeli wykonujących podobne zadania.
5. Minimalizacja liczby parametrów modelu przy zachowaniu satysfakcjonującego poziomu dokładności.

Dyskusja na temat tych problemów wciąż jest tematem wielu paneli dyskusyjnych i debat konferencyjnych (zob np. [2]). Najbardziej zaawansowane prace dotyczą problemu gromadzenia dużych zbiorów danych medycznych, co wymaga bliskiej współpracy ekspertów medycznych z ekspertami od uczenia maszynowego. Często konieczna jest również modyfikacja bądź tworzenie dedykowanych programów do akwizycji danych medycznych. Jako przykłady takich inicjatyw można wymienić programy Stanford Medicine [3], Harvard School of Medicine [12] czy Massachusetts General Hospital, które w swoich repozytoriach zgromadziły już dziesiątki milionów zdjęć radiologicznych (za [1]). Ponadto w roku 2018, na konferencji NVIDIA GTC (GPU Technology Conference) w San Jose (Kalifornia), Amerykańskie Stowarzyszenie Radiologii i stowarzyszenie MICCAI (od ang. *Medical Image Computing and Computer Assisted Intervention*) ogłosiły porozumienie, co do wspólnej współpracy mającej na celu eliminacje barier legislacyjnych związanych ze współpracą przy pozyskiwaniu danych i wykorzystania algorytmów uczenia maszynowego.

Autor tej rozprawy jest świadom ograniczeń jakie są związane z wykorzystaniem algorytmów głębokiego uczenia się. Jednocześnie jednak nowe algorytmy i ostatnie sukcesy zastosowań współczesnych sztucznych sieci neuronowych w aplikacjach medycznych stanowią silną motywację do przeprowadzenia własnych badań.

Rozdział 5

Nowa metoda oceny procesu gojenia ścięgna Achillesa

5.1 Metodyka

5.2 Rozróżnienie ścięgna zdrowego i po zerwaniu

5.3 Obliczanie krzywych gojenia

5.3.1 Topologia sieci

5.3.2 Redukcja wymiarowości

5.3.3 Miara wygojenia

Rozdział 6

Wyniki i walidacja

6.1 Ocena procesu gojenia z użyciem nowej metody

6.2 Porównanie z wynikami z rezonansu magnetycznego

6.3 Porównanie z wynikami ultrasonografii

6.4 Porównanie z wynikami badań biomechanicznych

Rozdział 7

Podsumowanie

Bibliografia

[1] Deep learning: The next step in applied healthcare data. https://insights.samsung.com/2016/07/12/deep-learning-the-next-step-in-applied-healthcare-data/. Accessed: 2018-22-05.

[2] From challenges to impact of machine learning in clinical practice. http://on-demand.gputechconf.com/gtc/2018/video/S8897/. Accessed: 2018-22-05.

[3] Medical image net. http://langlotzlab.stanford.edu/projects/medical-image-net/. Accessed: 2018-22-05.

[4] Nvidia tensorrt. https://developer.nvidia.com/tensorrt. Accessed: 2018-22-05.

[5] An overview of resnet and its variants. https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035. Accessed: 2018-22-05.

[6] Salem Saleh Al-amri, Namdeo V. Kalyankar, Khamitkar S. D. Image segmentation by using threshold techniques. *CoRR*, abs/1005.4020, 2010.

[7] Sanjeev Arora, Aditya Bhaskara, Rong Ge, Tengyu Ma. Provable bounds for learning some deep representations. *CoRR*, abs/1310.6343, 2013.

[8] Robert M. Bell, Yehuda Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, Grudzie/n 2007.

[9] Richard Ernest Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[10] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[11] Chun-houh Chen, Wolfgang Hrdle, Antony Unwin, Chun-houh Chen, Wolfgang Hrdle, Antony Unwin. *Handbook of Data Visualization (Springer Handbooks of*

Computational Statistics). Springer-Verlag TELOS, Santa Clara, CA, USA, wydanie 1, 2008.

[12] Junghwan Cho, Eunmi Lee, Hyunkwang Lee, Bob Liu, Xinhua Li, Shahein Tajmir, Dushyant Sahani, Synho Do. Machine learning powered automatic organ classification for patient specific organ dose estimation. *Society for Imaging Informatics in Medicine*, wolumen 2017, 2017.

[13] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, Yann LeCun. The loss surface of multilayer networks. *CoRR*, abs/1412.0233, 2014.

[14] Eric M. Christiansen, Samuel J. Yang, D. Michael Ando, Ashkan Javaherian, Gaia Skibinski, Scott Lipnick, Elliot Mount, Alison O'Neil, Kevan Shah, Alicia K. Lee, Piyush Goyal, William Fedus, Ryan Poplin, Andre Esteva, Marc Berndl, Lee L. Rubin, Philip Nelson, Steven Finkbeiner. In silico labeling: Predicting fluorescent labels in unlabeled images. *Cell*, 173(3):792–803.e19, apr 2018.

[15] Valentino Dardanoni, Giuseppe De Luca, Salvatore Modica, Franco Peracchi. Bayesian Model Averaging for Generalized Linear Models with Missing Covariates. Raport instytutowy, 2013.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR09*, 2009.

[17] Qi Dou, Hao Chen, Lequan Yu, Jing Qin, Pheng-Ann Heng. Multilevel contextual 3-d CNNs for false positive reduction in pulmonary nodule detection. *IEEE Transactions on Biomedical Engineering*, 64(7):1558–1567, jul 2017.

[18] John Duchi, Elad Hazan, Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, Lipiec 2011.

[19] Bradley J. Erickson, Panagiotis Korfiatis, Zeynettin Akkus, Timothy Kline, Kenneth Philbrick. Toolkits and libraries for deep learning. *Journal of Digital Imaging*, 30(4):400–405, Aug 2017.

[20] Bradley J. Erickson, Panagiotis Korfiatis, Timothy L. Kline, Zeynettin Akkus, Kenneth Philbrick, Alexander D. Weston. Deep learning in radiology: Does one size fit all? *Journal of the American College of Radiology*, 15(3):521–526, mar 2018.

[21] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, jan 2017.

[22] Matthew F. Glasser, Timothy S. Coalson, Emma C. Robinson, Carl D. Hacker, John Harwell, Essa Yacoub, Kamil Ugurbil, Jesper Andersson, Christian F. Beckmann, Mark Jenkinson, Stephen M. Smith, David C. Van Essen. A multi-modal parcellation of human cerebral cortex. *Nature*, 536(7615):171–178, jul 2016.

[23] Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep Learning*. MIT Press, 2016.

[24] Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C. Nelson, Jessica L. Mega, Dale R. Webster. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22):2402, dec 2016.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[27] F.S. Hill, Stephen M. Kelly. *Computer Graphics using OpenGL*. Prentice Hall, 2006.

[28] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[29] C. Ju, A. Bibaut, M. J. van der Laan. The Relative Performance of Ensemble Methods with Deep Convolutional Neural Networks for Image Classification. *ArXiv e-prints*, Kwiecie/n 2017.

[30] Diederik P. Kingma, Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[31] Patrick Koch, Brett Wujek, Oleg Golovidov, Steven Gardner. Automated hyperparameter tuning for effective machine learning. 2017.

[32] Stanisław Kowalik. Zastosowanie teorii zbiorów przybliżonych do podejmowania decyzji. *Zarządzanie przedsiębiorstwem w erze postindustrialnej – ekonomia, prawo, kultura, etyka*, strony 39–44, Kazimierz Dolny, Polska, 2003.

[33] Alex Krizhevsky. Learning multiple layers of features from tiny images. Raport instytutowy, 2009. Accessed: 2018-05-21.

[34] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, strony 1097–1105, USA, 2012. Curran Associates Inc.

[35] Y. Kwon, M. Rhu. A case for memory-centric hpc system architecture for training deep neural networks. *IEEE Computer Architecture Letters*, 17(2):134–138, July 2018.

[36] K. N. Leach. A survey paper on independent component analysis. *Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory (Cat. No.02EX540)*, strony 239–242, 2002.

[37] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[38] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. D. S. Touretzky, redaktor, *Advances in Neural Information Processing Systems 2*, strony 396–404. Morgan-Kaufmann, 1990.

[39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick. Microsoft coco: Common objects in context. David Fleet, Tomas Pajdla, Bernt Schiele, Tinne Tuytelaars, redaktorzy, *Computer Vision – ECCV 2014*, strony 740–755, Cham, 2014. Springer International Publishing.

[40] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram

van Ginneken, Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, dec 2017.

[41] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, Shixia Liu. Towards better analysis of deep convolutional neural networks. *CoRR*, abs/1604.07043, 2016.

[42] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, Jeffrey S. Vetter. NVIDIA tensor core programmability, performance & precision. *CoRR*, abs/1803.04014, 2018.

[43] Miriam Martínez, Luis Enrique Sucar, Hector Gabriel Acosta, Nicandro Cruz. Bayesian model combination and its application to cervical cancer detection. Jaime Simão Sichman, Helder Coelho, Solange Oliveira Rezende, redaktorzy, *Advances in Artificial Intelligence - IBERAMIA-SBIA 2006*, strony 622–631, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[44] Seymour Papert Marvin Lee Minsky. *Perceptrons: An Introduction to Computational Geometry*. Mit Press, 1969.

[45] Warren S. McCulloch, Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[46] Ümit V. Çatalyürek, Cevdet Aykanat, Bora Uçar. On two-dimensional sparse matrix partitioning: Models, methods, and a recipe. *SIAM Journal on Scientific Computing*, 32(2):656–683, 2010.

[47] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[48] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, A. Y. Ng. Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks. *ArXiv e-prints*, Lipiec 2017.

[49] Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[50] Frank Rosenblatt. The perceptron: A perceiving and recognizing automaton. Raport instytutowy, 1957.

[51] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986.

[52] N. Russell, T. Brendan Murphy, A. E Raftery. Bayesian model averaging in model-based clustering and density estimation. *ArXiv e-prints*, Czerwiec 2015.

[53] R. Kumar S. Hijazi, C. Rowen. Using convolutional neural networks for image recognition. Raport instytutowy, Cadence, 2015. Accessed: 2018-05-21.

[54] Saman Sarraf, Ghassem Tofighi. Deep learning-based pipeline to recognize alzheimers disease using fMRI data. *2016 Future Technologies Conference (FTC)*. IEEE, dec 2016.

[55] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[56] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[57] Dipanjan Sengupta, Yida Wang, Narayanan Sundaram, Theodore L. Willke. High-performance incremental svm learning on intel $${\tiny{\textregistered}}$$xeonphi$^{\mathrm{TM}}$processors. $Julian\ M. Kunkel, Rio Yokota, Pavan Bala$ $-138, Cham, 2017. Springer International Publishing.$

[58] Karen Simonyan, Andrea Vedaldi, Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[59] Karen Simonyan, Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[60] Leon Sixt, Benjamin Wild, Tim Landgraf. Rendergan: Generating realistic labeled data. *CoRR*, abs/1611.01331, 2016.

[61] Fengguang Song, Jack Dongarra. Scaling up matrix computations on shared-memory manycore systems with 1000 cpu cores. *Proceedings of the 28th ACM International Conference on Supercomputing*, ICS '14, strony 333–342, New York, NY, USA, 2014. ACM.

[62] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[63] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[64] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[65] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, may 2016.

[66] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, Tom Goldstein. Training neural networks without gradients: A scalable ADMM approach. *CoRR*, abs/1605.02026, 2016.

[67] Joshua B. Tenenbaum, Vin de Silva, John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

[68] Laurens van der Maaten, Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[69] Juha Vesanto, Johan Himberg, Esa Alhoniemi, Juha Parhankangas. Self-organizing map in matlab: the som toolbox. *In Proceedings of the Matlab DSP Conference*, strony 35–40, 2000.

[70] D. Wang, A. Khosla, R. Gargeya, H. Irshad, A. H. Beck. Deep Learning for Identifying Metastatic Breast Cancer. *ArXiv e-prints*, Czerwiec 2016.

[71] Michael B. W. Wolfe, Susan R. Goldman. Use of latent semantic analysis for predicting psychological phenomena: Two issues and proposed solutions. *Behavior Research Methods, Instruments, & Computers*, 35(1):22–31, Feb 2003.

[72] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[73] Robert H. Wurtz. Recounting the impact of hubel and wiesel. *The Journal of Physiology*, 587(12):2817–2823, jun 2009.

[74] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.

[75] Matthew D. Zeiler, Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[76] Wei Zhang, Kunio Doi, Maryellen L. Giger, Yuzheng Wu, Robert M. Nishikawa, Robert A. Schmidt. Computerized detection of clustered microcalcifications in digital mammograms using a shift-invariant artificial neural network. *Medical Physics*, 21(4):517–524, apr 1994.

Dodatek A

AchillesDL: System komputerowego wspomagania oceny gojenia ścięgien i więzadeł