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

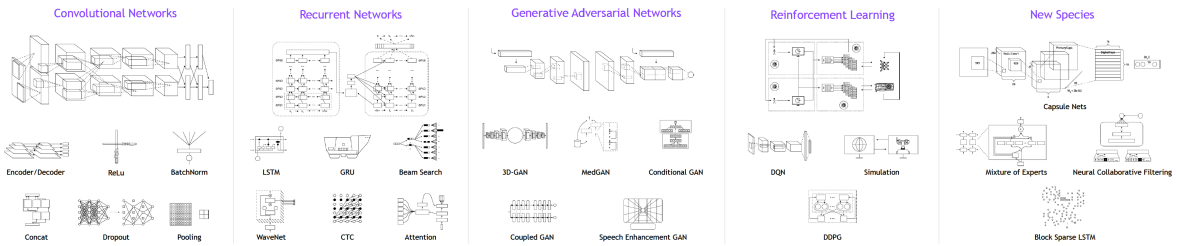
Artificial Neural Networks — 1975

Expectation Maximization — 1977

Decision tree — 1986

Q-learning — 1989

Random forest — 1995



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

Rozdział 2

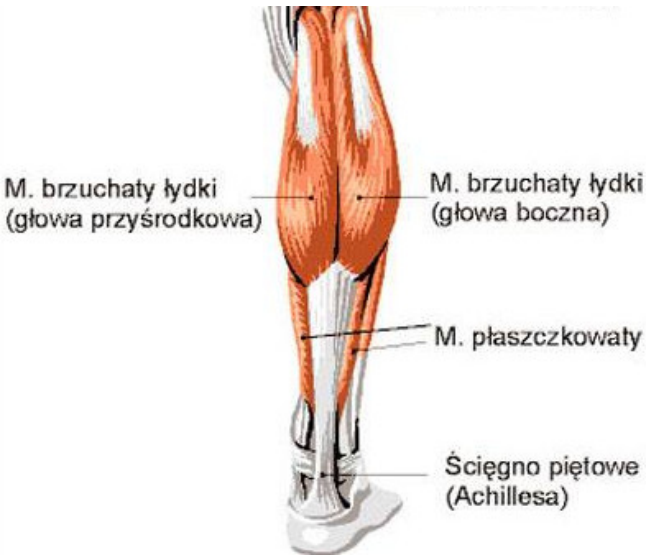
Cel i przebieg pracy

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)

funkcja obrazowa; splot; filtr; jądro; warstwy konwolucyjne; wartwy aktywacji neuronów; warstwy FC; Dropout; Normalizacja;

4.1 Zarys historyczny

4.2 Problem nadmiernego dopasowania

-dropout Kolejnym zabiegiem wykorzystywanym tym razem w celu polepszenia generalizacji klasyfikacji zbioru danych było użycie techniki dropout [Dropout].

Combining the predictions of many different models is a very successful way to reduce test errors [1, 3], but it appears to be too expensive for big neural networks that already take several days to train. There is, however, a very efficient version of model combination that only costs about a factor of two during training. The recently-introduced technique, called "dropout" [10], consists of setting to zero the output of each hidden neuron with probability 0.5. The neurons which are "dropped out" in this way do not contribute to the forward pass and do not participate in backpropagation. So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in

conjunction with many different random subsets of the other neurons. At test time, we use all the neurons but multiply their outputs by 0.5, which is a reasonable approximation to taking the geometric mean of the predictive distributions produced by the exponentially-many dropout networks. We use dropout in the first two fully-connected layers of Figure 2. Without dropout, our network exhibits substantial overfitting. Dropout roughly doubles the number of iterations required to converge.

–data augmentation

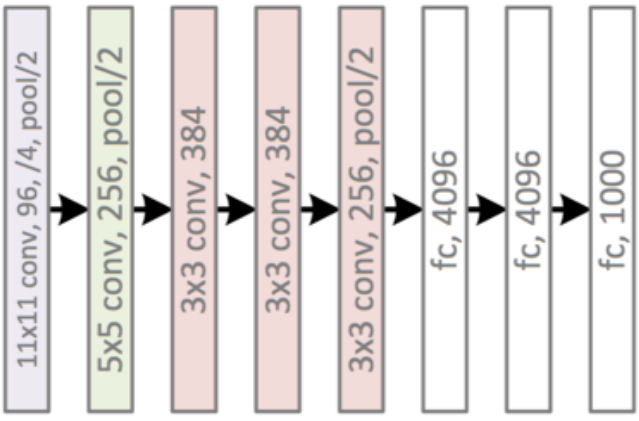
-5 fold

4.3 Problem redukcji wymiarowości

4.4 Przykłady współczesnych topologii

4.4.1 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. 4.1



Rysunek 4.1: Topologia architektury AlexNet.

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 jądrem o wymiarach 2×2 [1].

[1]autorzy pracy podają też przykłady użycia jąder o wymiarze 2×3, które nakładają się w prze-

Pierwsza warstwa konwolucyjna przyjmuje na wejściu dane o wymiarze 227×227×3, na których wykonywana jest operacja splotu z 96 filtrami z jądrem splotu 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 jądrem 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 zawieraja 384 filtry o wymiarze 3×3×256, w skład czwartej wchodzą 384 filtry o wymiarze 3×3×384, a w piątej znajdują się 256 filtry ponownie 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 [AlexNet].

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 (zarówno wzdłuż wysokości jak i szerokości) o wymiarach [11×11×3]. Punkty krawędziowe, które stanowią margines potrzebny do wyliczenia splotu są zazwyczaj pomijane. 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 sie 3025 blokach są przedstawiane w postaci macierzy A o wymiarach [96×363] i B o wymiarach [363×3025].

3. liczony jest iloczyn skalarny w postaci $\mathrm{A}^{\intercal}\mathrm{B} = C$, gdzie nowa, wyjściowa macierz C ma wymiar [96×3025].

4. Resultat w postaci macierzy C ponownie przewymiarowywany 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 CIFAR-10 [CIFAR] w stosunku do tej samej topologii wykorzystującej funkcję aktywacji tanh.

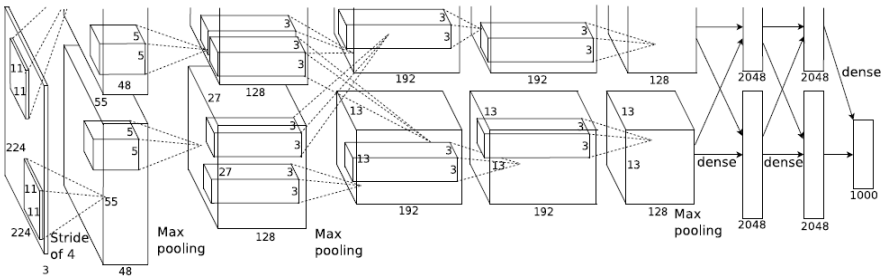
strzeni funkcji obrazowej

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:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^i_{x,y})^2 \right)^{\beta} \tag{4.1}$$

gdzie n to liczba filtrów znajdujących się w tej samej przestrzennej lokalizacji, N to suma wszystkich filtrów w warstwie, $a^i_{x,y}$ to wartość funkcji aktywacji neuronu po splocie funkcji obrazowej na pozycji (x, y) z filtrem i, a b to wynik normalizacji. Po zastosowaniu tej techniki autorom pracy udało się zredukować błąd klasyfikacji top-5 o wartość 1,2 punkta procentowego.

W kontekście zwiększenia efektywności treningu zastosowano powiększenie rozmiaru danych poprzez rotacje i modyfikacje funkcji obrazowej z wykorzystaniem czynników głównych (zob [AlexNet]), co zmniejszyło błąd top-1 o 1%. Zastosowano również technikę dropout opisaną w 4.2. Ostatecznie wprowadzono także trening z wykorzystaniem wielu GPU (zob. Rys. 4.2).



Rysunek 4.2: Topologia architektury AlexNet z podziałem na dwa akceleratory GPU.

Topologia z podziałem na 2 karty zwiększyła dwukrotnie sumaryczną pamięć i pozwoliła na koalokację 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 [ZFNet], gdzie m.in. zastosowano zmniejszenie wymiaru jądra stosowanego w filtrach pierwszej warstwy konwolucyjnej do 7×7 oraz VGGNet z 2014 roku, gdzie zastosowano większą liczbę warstw konwolucyjnych z mniejszym wymiarem jądra splotu. Innowacyjnym pomysłym, który został zaprezentowany również w 2014 było pojawienie się

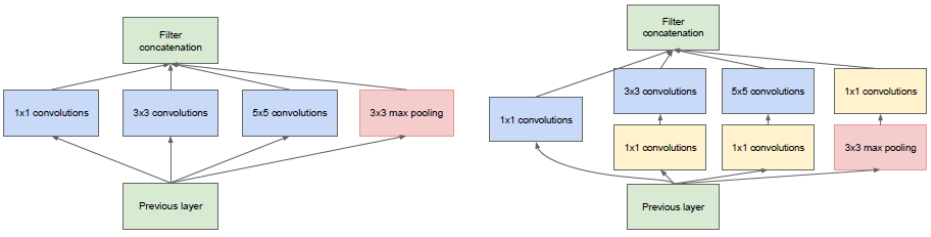
nowych modułów w sieci GoogleNet, która dokładniej została opisana w kolejnej podsekcji.

4.4.2 GoogLeNet

Architekturę o nazwie GoogLeNet zaprezentowano w 2014 r. w pracy [GoogleNet]. Nazwa architektury pochodzi od nazwy zwycięzkiego zespołu startującego w ILSVRC14, 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 i lokalizacji obiektów (zob. [ILSCV]) 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 redundatna z powodu wysokiej korelacji między sobą (zob. [Arora z GoogleNet]). Matematyka dotycząca przetwarzania *macierzy rzadkich*, tj. gdzie przeważająca liczba elementów przyjmuje wartość 0, jest dobrze znana np. [3-GoogleNet]. Jednak implementacje bibliotek do obliczeń związanych z algebrą liniową są zooptymalizowane pod kątem *macierzy gęstych*, gdzie przeważająca liczba elementów przyjmuje wartości różne od 0 (zob. [16, 9 googleNet]).

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*). Przykłady modułów incepcji pokazano na Rys. 4.3

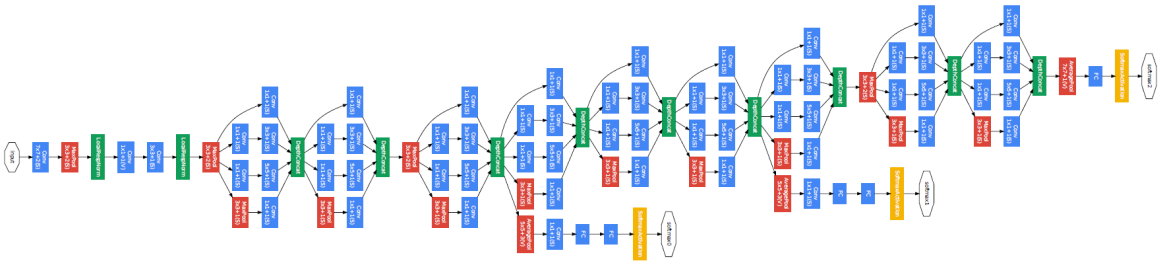


Rysunek 4.3: Topologia architektury AlexNet z podziałem na dwa akceleratory GPU.

Rys. 4.3 (a) przedstawia najwną formę modułu incepcji, gdzie grupowane są operacje filtrów z jądrem o wymiarach 5×5, 3×3, 1×1 oraz operacja max-pool. Rys. 4.3 (b)

prezentuje koncepcję zooptymalizowaną obliczeniowo gdzie filtry 1×1 służą do redukcji wymiarowości i używane są bezpośrednio przed splotami z bardziej wymagającymi obliczeniowo splotami z jądrami o wymiarach 5×5 i 3×3.

Złożenie różnego rodzaju modułów dało topologię zaprezentowaną na Rys.



Rysunek 4.4: 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 znajdowało 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. 4.4. Moduł zawiera 128 filtrów z jądrami o wymiarach 3×3 i 32 filtry z jądrami 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 jądrem 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.

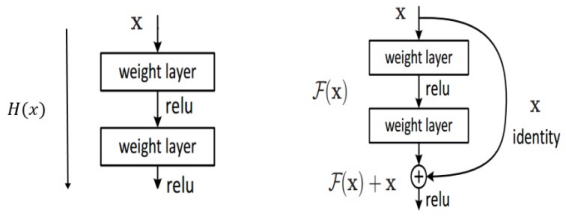
Topologia GoogLeNet jest wciąż rozwijana. Po pierwszej prezentacji pojawiły się kolejne modernizacje wprowadzające dodatkowe faktoryzacje modułów jak w Inception-v2 [Inception-v2], lub normalizacje wartości wynikowych poszczególnych warstw jak w Inception-v3 [Inception-v3]. Kolejny innowacyjny pomysł,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

4.4.3 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 [ResNet]. 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, których przykłady zostały opisane w sekcji 4.4.1 oraz szeroko w literaturze np. w [ResNet 2,3,4]. 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 redundatne, a nawet prowadzi do pogorszenia wyników treningu sieci, co związane jest z trudnościami optymalizacji przestrzeni wieloparametrycznych [Opt]. Zjawiskio to nosi nazwę degradacji treningu (ang. *degradation problem*). Twórcy architektury ResNet 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 został przedstawiony na Rys. 4.5.



Rysunek 4.5: Schemat funkcjonalny pojedynczego bloku w architekturze ResNet.

Ogólną postać równania bloku rezydualnego można zapisać następująco:

$$\begin{aligned} y_l &= h(x_l) + F(x_l, W_l), \\ x_{l+1} &= f(y_l), \end{aligned} \tag{4.2}$$

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:

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, W_i) \tag{4.3}$$

Korzystając z reguły łańcuchowej [ResNet-wyj-9] można zapisać równanie na gradient funkcji kosztu ε:

$$\text{??}\frac{\partial \varepsilon}{\partial x} = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F(x_i, W_i)\right) \tag{4.4}$$

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.

Rozważając przykład sieci składający się z trzech bloków można zapisać:

$$\frac{\partial \varepsilon}{\partial x_0} = \frac{\partial \varepsilon}{\partial x_3} * (w_2 + \lambda_2) * (w_1 + \lambda_1) * (w_0 + \lambda_0) \tag{4.5}$$

Można następnie wyszczególnić 4 przypadki:

(https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cc5d0adf648e) Przykład: lambda = gradient Case-1, Lambda = 0: This will be a plain network. Since w2, w1, w0 are all between -1, 1, the gradient vanishes as the network depth increases. This clearly shows vanishing gradient problem

Case-2, Lambda ¿1: In this case, The backprop value increases incrementally and lead to exploding of gradients.

Case-3, Lambda ¡1: For shallow networks this might not be problem. But for extra large networks, weight+lambda is still less than ¡1 in most cases and it achieves the same problem as case-1

case-4, Lambda =1: In this case, Every weight is incremented by 1, This eliminates the problem of multiplying with very large numbers as in case-2 and small numbers as in case-1 and acts as a good barrier.

Kompletna sieć zaproponowana w 2015 roku została pokazana na Rys.: (wstaw rysunek topo)

Warianty ResNet: (https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035)

4.4.4 Złożenia

4.5 Zastosowania w medycynie

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] Witold Pokorski and Graham G. Ross. Flat directions, string compactification and three generation models. 1998.

Dodatek A

AchillesDL: System komputerowego wspomagania oceny gojenia ścięgien i więzadeł