VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HLUBOKÉ NEURONOVÉ SÍTĚ PRO ANALÝZU MEDICÍNSKÝCH OBRAZOVÝCH DAT

DEEP NEURAL NETWORKS FOR MEDICAL VISUAL DATA ANALYSIS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ BÍL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL ŠPANĚL, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Bíl Tomáš**
Program: Informační technologie
Název: **Hluboké neuronové sítě pro analýzu medicínských obrazových dat**
Deep Learning for Medical Image Analysis
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s problematikou hlubokých neuronových sítí a jejich učení.
2. Zorientujte se v metodách segmentace medicínských obrazových dat s využitím hlubokých neuronových sítí (medicínská CT data, RTG snímky, apod.).
3. Vyberte vhodnou metodu použitelnou pro řešení zvoleného problému analýzy medicínského obrazu.
4. Implementujte navrženou metodu s využitím existujících nástrojů pro trénování hlubokých neuronových sítí.
5. Proveďte experimenty nad připravenou datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Krizhevsky, A., Sutskever, I. and Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Španěl Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 2. dubna 2019

Abstrakt

Cílem této práce je vyvinout konvoluční neuronovou síť, která bude správně klasifikovat vhodnost rentgenových snímků pro účely kefalometrické analýzy. V rámci tohoto zadání byly vytvořeny 4 sítě, které byly natrénovány na datasetu. Jedná se 2 sítě typu VGG, jedna upravená síť UNet a jedna síť Resnet. Dataset byl vygenerován z ct snímků. Nejlepších výsledků dosáhla síť VGG se 4 bloky. Naměřená přesnost na testovacím datasetu je 97%.

Abstract

The goal of this thesis is developing convolutional neural network which is able to classify if x-ray images are suitable for cephalometry analysis. Four networks were created and trained on a dataset for this purpose. Two of them are VGG type, one is based on UNet and one is Resnet. The dataset was generated from ct scan images. VGG network with four blocks has got the best results. Measured accuracy performed on test dataset is 97%.

Klíčová slova

strojové učení, neuronové sítě, kefalometrie, rozpoznání obrazu

Keywords

machine learning, Artificial Neural Network, cephalometry, image recognition

Citace

BÍL, Tomáš. *Hluboké neuronové sítě pro analýzu medicínských obrazových dat.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Španěl, Ph.D.

Hluboké neuronové sítě pro analýzu medicínských obrazových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Bíl
9. května 2019

Poděkování

Děkuji za vedení při tvorbě této práce svému vedoucímu.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá validací a případným návrhem na korekci kefalometrických obrázků. Kefalometrickými obrázky se v tomto případě myslí boční rentgenové snímky hlavy pacienta. Při snímání dochází k chybě lidského faktoru a pacient může pohnout hlavou a znehodnotit snímek pro účely kefalometrie. Cílem této práce je zjistit, zda lze provést na snímku kefalometrická analýza, která je náročná na přesnost určení určitých bodů na snímku. Z tohoto důvodu je důležité, aby byl snímek kvalitní a pacient byl ve správné poloze při vyfocení rentgenového snímku. Pokud je snímek vyhodnocen jako znehodnocený, program doporučí otočení hlavy pacienta o určitý počet stupňů pro nápravu. Toto umožní okamžitou nápravu snímku bez nutnosti konzultace s lékařem, která zvyšuje celkovou dobu potřebnou na provedení vyšetření.

Pro dosažení žádaného výsledku byly vytvořeny a natrénovány 4 konvoluční neuronové sítě. Každá z nich se inspiruje již navrženými a osvědčenými architekturami. Tyto architektury byly implementovány a pozměněny tak, aby vyhovovaly záměrům této práce. S těmito sítěmi se poté experimentovalo, aby se dosáhlo nejlepších výsledků. Při každém experimentu se měřily určité metriky, které charakterizují schopnosti trénované sítě. V závislosti na naměřených hodnotách se upraví parametry dalších experimentů.

Pro účely trénování a testování sítí byl vytvořen dataset. Tento dataset je rozdělen na testovací a trénovací data. Toto rozdělení je důležité, protože přesnost na trénovacích datech může být zavádějící. Proto se měří skutečná přesnost sítě na datech, se kterými se síť při trénování nesetkala. Dataset bylo potřeba uměle vygenerovat z CT skenů kvůli nedostatku volně dostupných zdrojů s potřebnými snímky. Kefalometrické snímky jsou sice dostupné, bohužel však jen ty, které jsou vhodné pro kefalometrii. To znamená, že dostupné datasety reprezentují jen část snímků, které chceme, aby se naučila síť rozpoznávat. Všechny snímky byly rozděleny do tříd, které odpovídají míře natočení hlavy na snímku.

Natrénované sítě většinou dosahují přesnosti přes 90%. To znamená, že sítě dělají při klasifikaci jen minimum chyb. Z naměřených metrik také vyplývá, že jsou zaměňovány hlavně snímky podobných kategorií. Není tedy tak zásadní, jestli program doporučí korekci natočení hlavy o 5 stupňů, když je reálné natočení oproti ideální pozici 4 stupně.

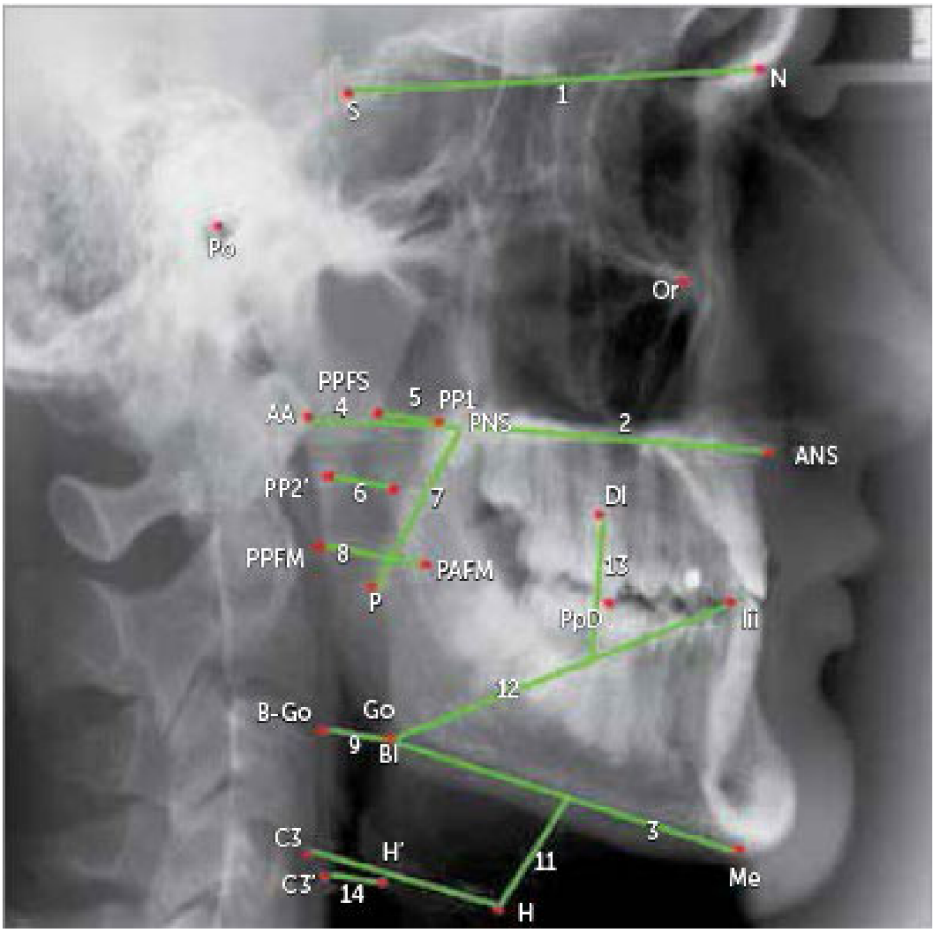
V druhé kapitole je stručný popis kefalometrie, který vysvětluje problém, který se tato práce snaží řešit. Třetí kapitola se zabývá teoretickým základem neuronových sítí, jež se v této práci využívají. Tvorba a rozdělení datasetu je obsaženo v kapitole číslo 4. Popis použitých architektur je v kapitole pět a informace o implementaci v kapitole šest. Způsob experimentování a naměřené výsledky na implementovaných sítích se nacházejí v kapitole sedm.

Kapitola 2

Kefalometrie

Pro kefalometrickou analýzu je potřeba určit na bočním rentgenovém snímku hlavy takzvané kefalometrické landmarky. Kefalometrické landmarky jsou body na rentgenovém snímku, které vypovídají o budoucím vývoji a aktuálním stavu lebky a čelistí. Detekce těchto landmarků je prvním krokem kefalometrické analýzy. Kefalometrická analýza slouží k rozpoznání budoucích problémů při vývoji lebky, zhodnocení jejich léčby a naplánování případných chiurgických zákroků. Kefalometrická analýza získává tyto informace ze zhodnocení vztahů mezi landmarky. Mezi tyto vztahy patří například vzdálenosti nebo úhly. Proto je důležité, aby byl zkoumaný snímek co možná nejlepší. Natočení hlavy v momentě focení snímku může ovlivnit vztahy mezi určenými landmarky a tím znehodnotit nebo dokonce pozměnit diagnózu. Riziko špatné diagnózy je o to větší v případě, že je kefalometrická analýza prováděna automaticky, protože většina programů, které tuto analýzu provádějí, buď kvalitu analyzovaného snímku nekontroluje, nebo pokud ano, tak jen zběžně. Zkreslené výsledky takového programu by mohly v konečném důsledku zamezit odhalení problému u pacienta a tak ho ohrozit.

Poměrně velkou neznámou je nalezení hranice, která odděluje snímky validní od nevalidních. Podle informací od pracovníka radiologického oddělení neexistuje žádné obecné pravidlo, které by určovalo, jaké natočení hlavy je stále akceptovatelné a jaké již ne. Naopak se ideální natočení liší podle toho, co se snažíme ze snímku zjistit. Doktoři si často dokonce určité natočení vyžádají.



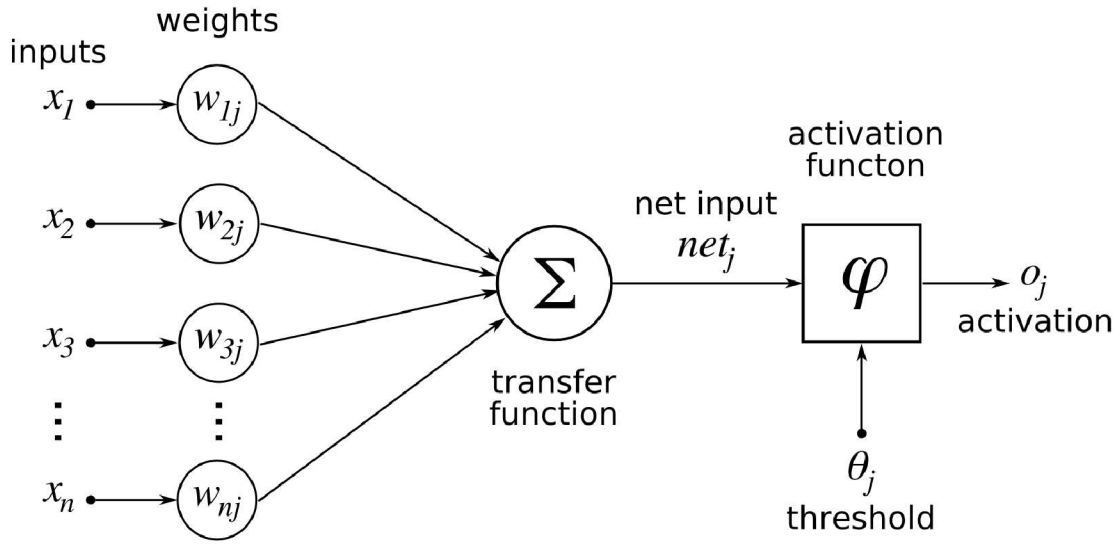
Obrázek 2.0: Příklad kefalometrické analýzy [14].

Kapitola 3

Umělé neuronové sítě

Umělá neuronová síť je model velmi často využívaný v oboru informatiky, která se zabývá umělou inteligencí. Tento model má svou inspiraci v biologickém neuronu. Neurony jsou navzájem propojeny a každý neuron signál upraví a předá dál. Tímto způsobem se vstupní informace transformuje do stavu, který vyžadujeme. Zbývajícím problémem je, jak nastavit transformace, které vykonávají jednotlivé neurony, aby byla výsledná informace taková, jakou si ji představujeme. Způsob transformování signálu určují váhy. Tyto váhy jsou upravovány během fáze učení. Učení se rozlišuje na učení s učitelem a učení bez učitele. V případě učení s učitelem jsou síti předkládány vstupy, neuronová síť vstup zpracuje a poté se výsledky ze sítě porovnají se správnými výsledky a podle toho se v síti upraví váhy, aby byly získané výsledky podobnější skutečným. Toto se opakuje vícekrát. Tento druh učení využívám i v této práci. Úprava vah se řídí algoritmem Backpropagation. Jednotlivé neurony bývají uspořádány do vrstev [18].

3.1 Umělý neuron



Obrázek 3.0: Schéma umělého neuronu [4].

Jednotlivé vstupy umělého neuronu jsou na obrázku označeny jako x. Jako w je označen vektor vah. θ je práh, který aktivuje neuron. Toto jsou všechno vstupy do aktivační funkce.

Neuron se dá použít pouze pro řešení lineárně separovatelných problémů. Pro složitější problémy je třeba použít síť vytvořenou z těchto neuronů. Předchůdcem neuronu je perceptron, který je neuronu velmi podobný [18].

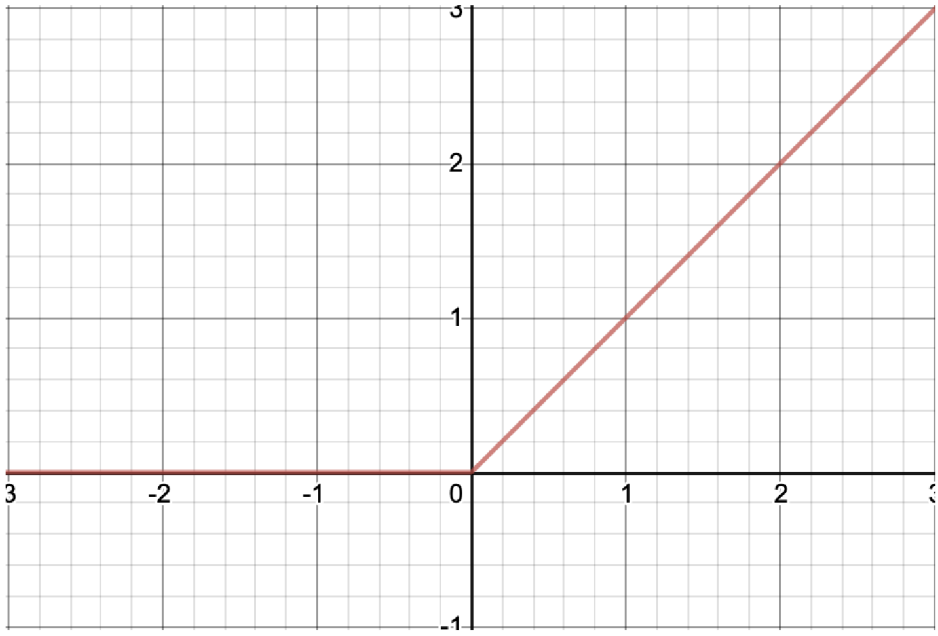
3.2 Aktivační funkce

Aktivační funkce je součástí neuronu, která vytváří nelinearitu ve výstupu neuronu. Toto je důležité z toho důvodu, že pokud by aktivační funkce v neuronu nebyla, byla by síť schopná řešit pouze trivialní problémy. Použití aktivačních funkcí umožňuje menším sítím počítat složitější problémy.

ReLU

ReLU [3] (Rectified Linear Unit) je nejpoužívanější aktivační funkce u dopředných neuronových sítí. Průběh funkce je velmi podobný lineární funkci, a proto zachovává některé výhody lineárních funkcí. Výhody použití této aktivační funkce jsou snadná propagace gradientu a efektivní výpočet. Tuto funkci lze implementovat pomocí pouze operací porovnání, sčítání a násobení. Předpis této funkce je:

$$f(x) = max(0, x)$$

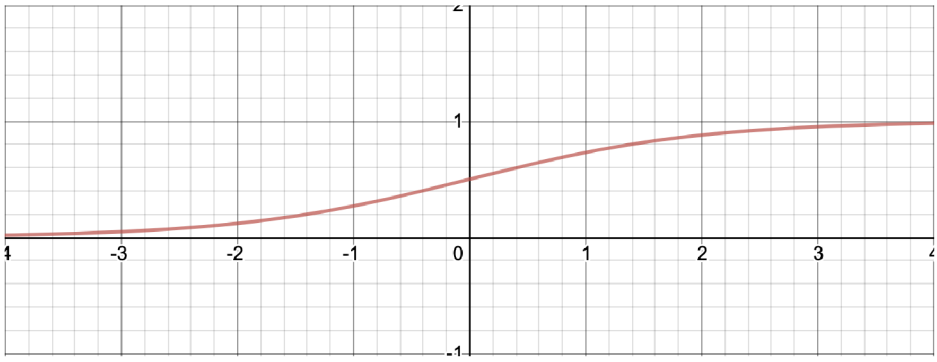


Obrázek 3.1: ReLU.

Sigmoida a hyperbolický tangens

Tyto funkce jsou si podobné, rozdíl je v rozsahu oboru hodnot. Používají se hlavně na výstupních vrstvách, použití na skrytých vrstvách se příliš nedoporučuje, protože rozsah jejich definičního oboru ztěžuje učení sítě. Předpis sigmoidy [3] je:

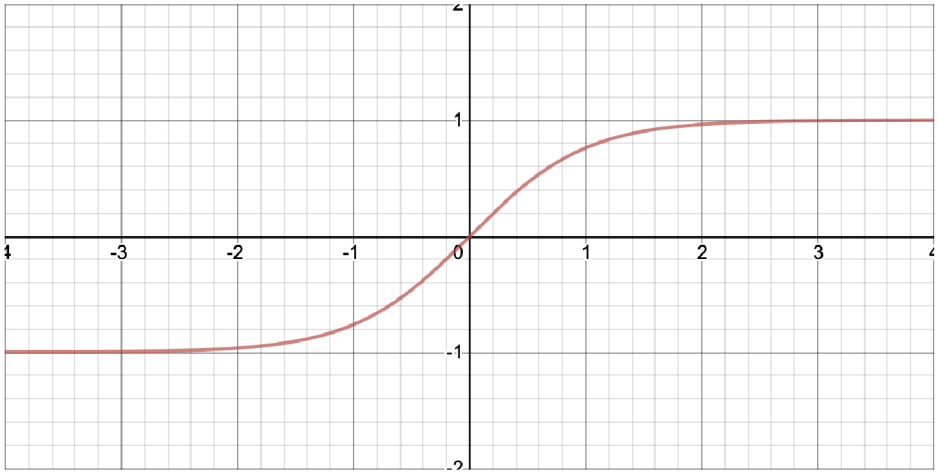
$$f(x) = \sigma(x)$$



Obrázek 3.2: Sigmoida.

Předpisem hyperbolického tangensu [3] je:

$$f(x) = tanh(x)$$



Obrázek 3.3: Hyperbolický tangens.

Softmax

Tato funkce se používá také u výstupních neuronů. Tato funkce je podobná sigmoidě, také je výstup mezi 0 a 1. Rozdíl je v tom, že softmax normalizuje výstup tak, že součet všech hodnot výstupních neuronů dává 1. Tímto způsobem se určí pravděpodobnost, že vstupní data sítě patří do dané kategorie [13].

3.3 Topologie

Topologií se, pokud mluvíme o neuronových sítích, myslí jejich uspořádání. Neurony jsou uspořádány do vrstev. V případě dopředné neuronové sítě přivádí každý neuron svůj výstup na vstup každého neuronu další vrstvy, Takto tvoří provázanost jednotlivých neuronů něco jako síť, podle toho vznikl název Neuronová síť [18].

3.4 Typy vrstev

Vrstvy neuronových sítí můžou mít více typů v závislosti na druhu neuronů, které je tvoří. Vrstvy, ze kterých se neuronová síť skládá, udávají z velké části vlastnosti a schopnosti sítě.

Plně propojená dopředná vrstva

Plně propojená dopředná vrstva je uspořádání neuronů, každý neuron má na vstupu výstup každého neuronu předchozí vrstvy. Neurony ve stejné vrstvě na sebe nejsou navzájem napojené. Takto se signál šíří pouze jedním směrem. Tato vrstva je tedy nezávislá na kontextu, minulé vstupy nemají žádný vliv na aktuální stav [7].

Rekurentní vrstva

Rekurentní vrstvy neuronové sítě berou v potaz i časový kontext vstupů. Tyto vrstvy tedy šíří signál i v opačném směru. Tímto způsobem je pak neuronová síť schopna vnímat i minulé vstupy. Výstup z této vrstvy je přiveden na vstup té samé vrstvy při zpracování dalšího vstupu.

Konvoluční vrstva

Tato vrstva se využívá hlavně při rozpoznávání obrazu. Proto je tato vrstva jednou z hlavních součástí všech architektur, které jsou v této práci použity. Vstupem do této vrstvy je n-dimensionální pole, výstupem také. Prvky výstupu se nazývají feature mapy. Každá feature mapa je výsledkem aplikace konvolučního filtru na vstup. Počet filtrů definuje velikost výstupního pole. Velikost filtru udává, kolik vah se musí v rámci tohoto filtru natrénovat. Filtry mívají typicky souměrné velikosti dimenzí a na rozpoznání obrazu se zpravidla využívají dvourozměrné filtry. V praxi se osvědčily filtry s lichými velikostmi rozměrů, proto pokud se podíváme na aplikaci konvolučních vrstev, uvidíme hlavně filtry s velikostmi 3x3 a 5x5 [3].

Pooling vrstva

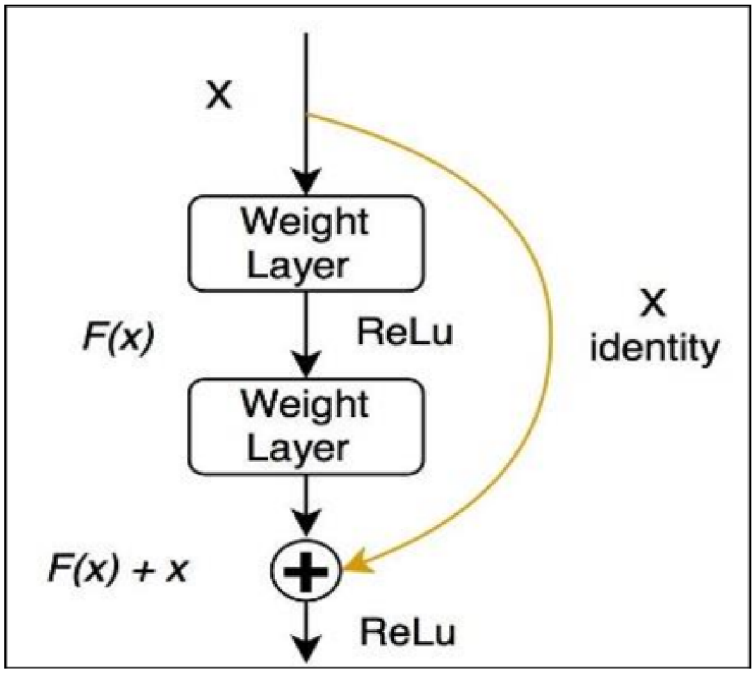
Funkcí pooling vrstvy je redukce prostoru, zmenšení vstupu do dalších vrstev sítě a tím zmenšení počtu parametrů, které se potřeba natrénovat. Tato vrstva má 2 parametry, prvním je velikost jádra a druhým krok posunu. Velikost jádra určuje velikost prostoru, ze kterého se budou vybírat nové hodnoty. Krok posunu určuje, o kolik se bude jádro posouvat. Čím menší je jádro, tím větší detaily je schopno zachovávat. Podle výběru hodnot z prostoru jádra se pooling vrstvy dělí na max-pooling a average-pooling. Max-pooling vrstvy vybírají z prostoru nejvyšší hodnoty a ty předá na svůj výstup. Average-pooling vrstvy mají jinou strategii výběru výstupní hodnoty, udělají průměr z celého prostoru jádra a ten předají na výstup [3].

Batch normalizace

Batch normalizace se používá, aby se síť nepřestávala učit. Při trénování sítě dochází k vytrácení a explodování signálu. Toto se stává hlavně v hlubokých sítí, kdy dochází k opakovanému násobení signálu, signál se může dostat do stavu, kdy vymizí nebo se zvětší do příliš vysokých hodnot. Síť se potom přestává učit. Batch normalizace pomáhá předcházet těmto problémům tak, že od každé dimenze odečte průměr batche a vydělí standardní odchylkou batche. V další vrstvě může optimalizátor změnit tyto parametry za účelem snížení loss funkce [12].

Reziduální spoj

Reziduální spoje se využívají v hlubokých neuronových sítích. Jeho cílem je předcházení mizejícího explodujícího signálu a optimalizace hloubky sítě, aby se zbytečně neučily vrstvy, které nejsou potřeba. Tento spoj funguje tak, že se sečte na vstupu do bloku výstup předcházejícího bloku s blokem ještě před blokem předcházejícím [10].



Obrázek 3.4: Residuální spoj [8].

Dropout vrstva

Dropout vrstva slouží k redukci přetrénování sítě. Při šíření signálu touto vrstvou dochází k deaktivaci určitého množství prvků. Díky tomu síť zapomene určité vlastnosti, které se během trénování naučila. Takto se předchází přetrénování neuronové sítě [7].

3.5 Trénování

Motivací při trénování neuronové sítě je snaha najít co možná nejmenší součet odchylky a variance. Variance je na začátku minimální a v průběhu trénování se snažíme, aby rostla co nejméně. Odchylka je naopak na začátku vysoká a v průběhu trénování neuronové sítě se ji snažíme minimalizovat. Trénování se snažíme ukončit v momentě, kdy je součet odchylky a variance minimální.

Trénování si jde představit jako šíření signálu skrz neuronovou síť a následnou úpravu vektorů vah každého neuronu pomocí algoritmu zpětné propagace chyby. Toto je prováděno několikrát, dokud nezískáme pocit, že je součet odchylky a variance minimální.

Při trénování se obvykle data, která máme k dispozici, rozdělují na trénovací a validační. Toto rozdělení slouží k tomu, aby bylo možné kontrolovat, zda při trénování sítě nedochází k přetrénovaní. Na validační sadě se také měří chyba a v momentě, kdy se tato chyba přestane snižovat, je vhodné trénování ukončit [18].

Kromě rozdělení dat na trénovací a validační se v rámci těchto kategorií dělí na takzvané batche. Toto rozdělení slouží k tomu, aby se váhy neupravovaly po každém jednotlivém vstupu, ale až po celém batch. Tím se předchází přetrénovaní a zároveň se podstatně zrychlí proces učení neuronové sítě, protože se může zpracovávat každý vstup v rámci batche paralelně. Rozdělení datasetu na batche se dělá po zamíchání celého trénovacího datasetu. Velikost batche je třeba volit v závislosti na velikosti celého trénovacího datasetu a možnostech hardwaru, na kterém se síť trénuje.

3.6 Algoritmus zpětné propagace chyby

Algoritmus zpětné propagace chyby (Backpropagation) spočívá v porovnání výsledku, který vznikl zpracováním vstupních dat neuronovou sítí, s výsledkem změřeným (správným) a v následném výpočtu gradientu chyby. Gradient chyby se vypočítá jako parciální derivace funkce, která je tvořena jednotlivými váhami na vstupu každého neuronu. Tento výpočet se provede pro každou vrstvu naší neuronové sítě. Gradient chyby si můžeme představit jako směr, kterým se musí váhy změnit, aby byla chyba co nejmenší. Poté upravíme váhy podle gradientu chyby. Velikost úpravy se řídí nastavením sítě (proměnná learning rate) [18].

3.7 Vyhodnocení úspěšnosti

Pro vyhodnocení úspěšnosti vytvořené sítě se používá více metrik. Pro různé úlohy je vhodné použít různé metriky, aby co nejlépe reprezentovaly schopnost sítě řešit zadaný problém. Zmíním zde nejdůležitější metriky pro klasifikační úlohy.

Důležité pojmy

V případě, že v rámci řešené úlohy je výsledkem zařazení do kategorií typu „ano/ne“, můžeme kategorii „ano“ prezentovat jako positive a kategorii „ne“ jako negative.

- True positive (TP) je počet snímků, které patří do kategorie positive a zároveň je síť zařadila do positive.
- True negative (TN) je počet snímků, které patří do kategorie negative a zároveň je síť zařadila do negative.
- False positive (FP) je počet snímků, které patří do kategorie negative a zároveň je síť zařadila do positive.
- False negative (FN) je počet snímků, které patří do kategorie positive a zároveň je síť zařadila do negative.

Přesná shoda

Přesná shoda [17] se vypočítá jako poměr správně klasifikovaných obrázků a celkového počtu obrázků, které byly klasifikovány. Tato metrika se využívá hlavně pro úlohy, kde každý obrázek patří do právě jedné kategorie. V případě, že obrázek patří do více kategorií, počítá se jako úspěšná klasifikace správné určení všech jeho kategorií. Pokud ovšem síť rozpozná pouze jednu ze dvou kategorií, je obrázek špatně zařazený navzdory tomu, že je jedna kategorie správná. Proto není vhodné používat tuto metriku pro úlohy, kde může mít obrázek více kategorií.

Accuracy

Accuracy [17] lze vypočítat jako:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

a určuje poměr správně zařazených obrázků vůči celkovému počtu obrázků.

Precision

Precision [17] určuje, s jakou přesností zařazuje síť obrázky do positive kategorie. Precision se vypočítá takto:

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall [17] udává, kolik positive obrázků síť zaznamenala. Recall se vypočítá jako:

$$Recall = \frac{TP}{TP + FN}$$

F-measure

F-measure [17] je kombinací Precision a Recall. Výpočet je následující:

$$F = 2 * \frac{Precision * Recall}{Precision + Recall}$$

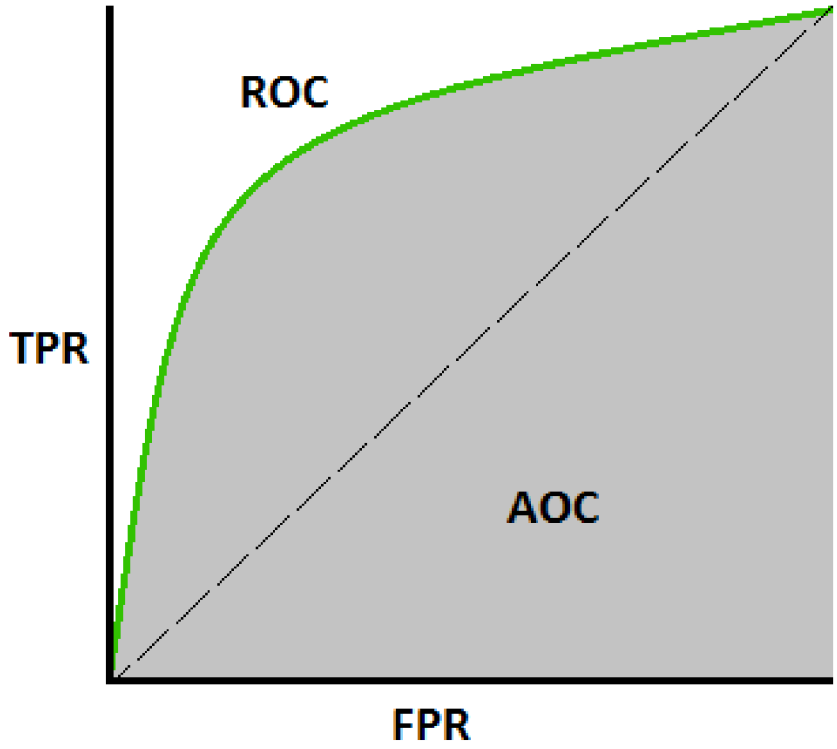
AUC-ROC

AUC-ROC [6] je možné vypočítat jako plochu pod ROC křivkou. ROC (Receiver Operating Characteristic) křivka slouží k hodnocení binárního klasifikátoru. Binární klasifikátor vypočítává pravděpodobnost, že zpracovávaný obrázek patří do dané kategorie. Hranice pravděpodobnosti, podle které se rozhoduje, zda bude obrázek do třídy zařazen, bývá obvykle 0,5. Pokud je pravděpodobnost vyšší než 0,5, je obrázek zařazen do dané kategorie, v případě, že je vypočtená pravděpodobnost nižší, obrázek do kategorie nebude zařazen. Hodnoty na ose x na ROC křivce jsou hodnoty specificity. Na ose y jsou hodnoty sensitivity. Specificity je false positive rate a sensitivity je true positive rate. False positive rate se vypočítá jako:

$$FPR = \frac{TN}{TN + FP}$$

True positive rate se vypočítá jako:

$$TPR = \frac{TP}{TP + FN}$$

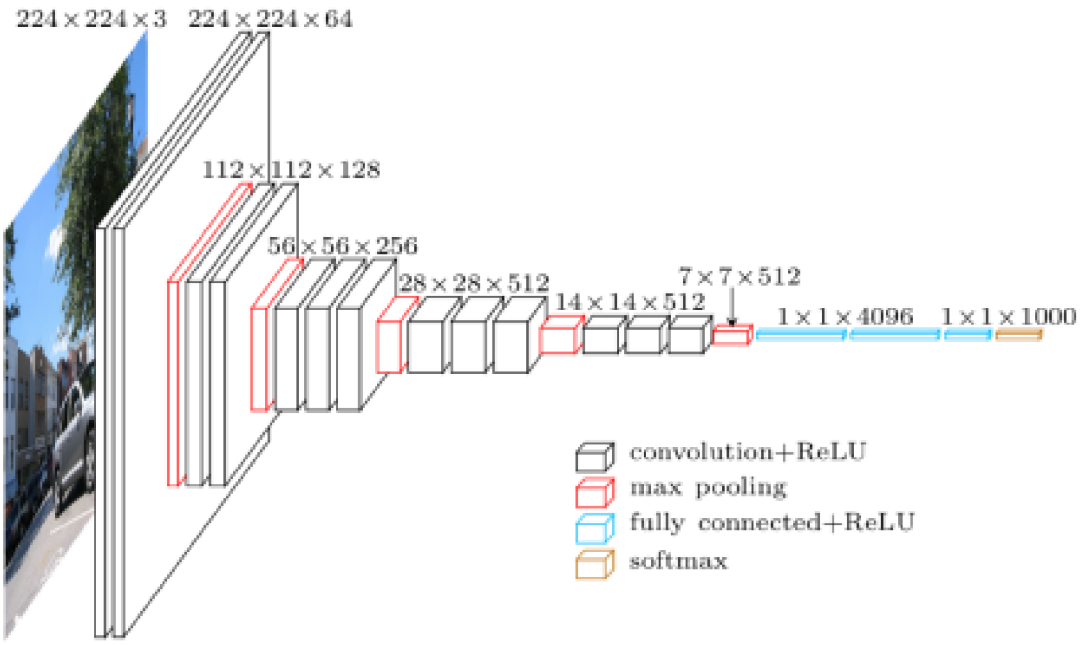


Obrázek 3.5: AUC-ROC křivka.

3.8 Architektury konvolučních sítí

VGGNet

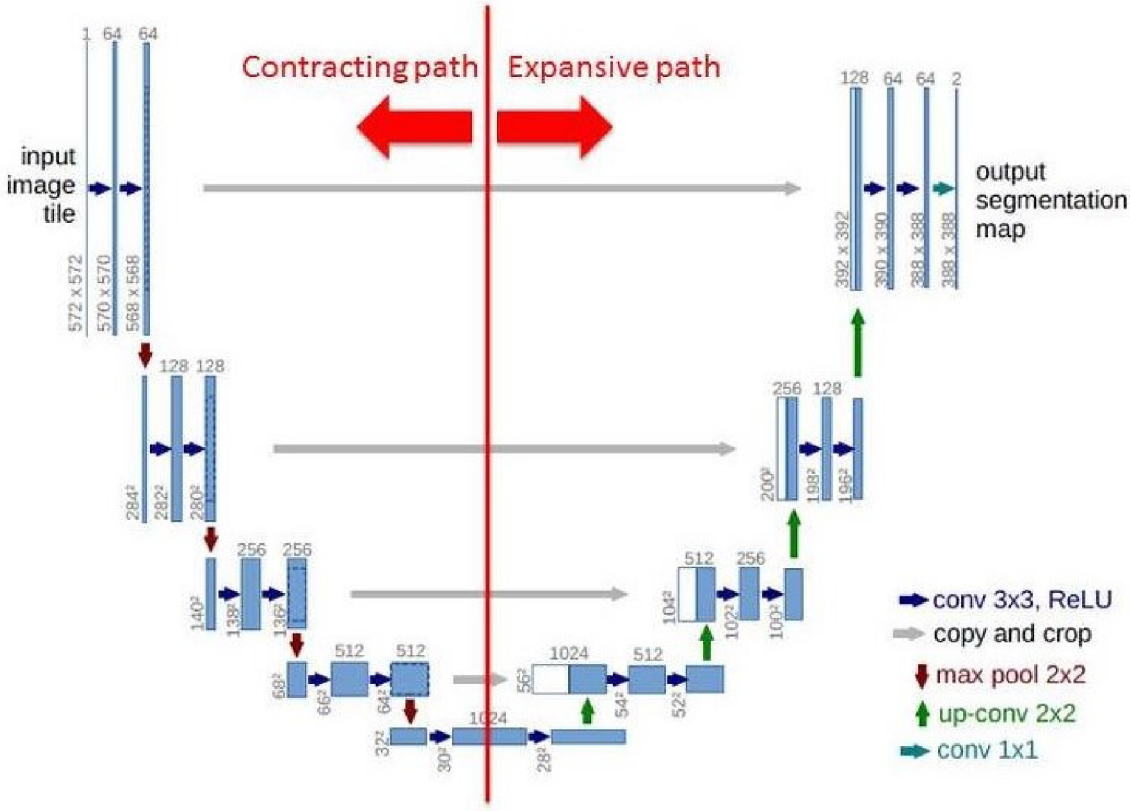
VGG [16] je jedna z nejznámějších architektur, které se využívají v klasifikaci obrazových dat. Střídají se v ní vrstvy, které se rozdělují do bloků. Jeden blok se skládá z několika konvolučních vrstev, nelineární aktivace a pooling vrstvy. Za sebou je několik bloků následovaných jednou nebo více lineárními vrstvami, které tvoří na výstupu vektor, jenž udává ohodnocení pravděpodobnosti, že obrázek patří do dané kategorie.



Obrázek 3.6: VGG architektura [2].

UNet

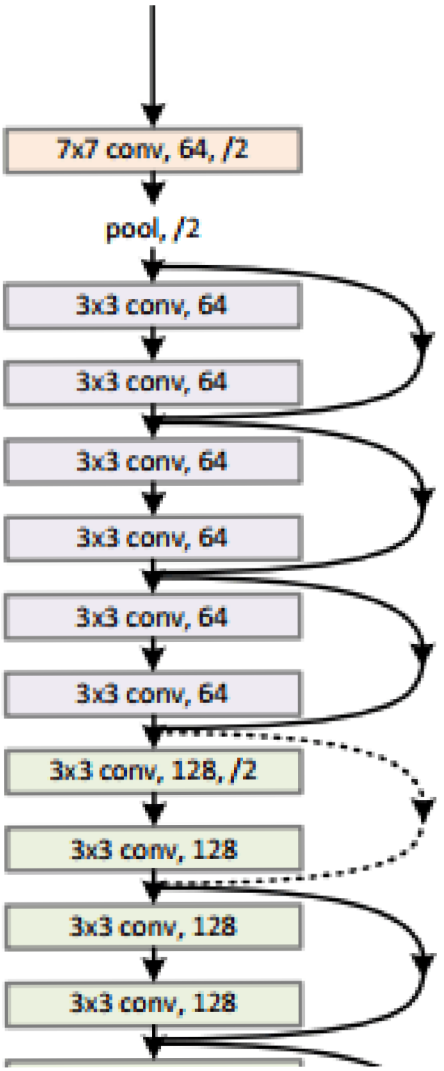
UNet [15] je architektura konvolučních neuronových sítí navržená pro práci s biomedicínskými obrazy. Tato síť se skládá ze dvou částí. První částí je jednoduchá VGG síť, která podvzorkovává vstup. Druhá část naopak signál nadvzorkovává, dokud nemá stejný tvar jako na vstupu do sítě. UNet se liší od jednoduchého hourglass modelu tím, že vrstva, která podvzorkovává, dává svůj výstup kromě nižší vrstvy také na protilehlou vrstvu, která nadvzorkovává.



Obrázek 3.7: UNet architektura [15].

ResNet

Architektu ResNet [11] neboli residual neural network řeší problém vytrácejíciho se signálu, který se často objevuje právě u architektur s velkou hloubkou (velké množství za sebou jdoucích vrstev). Problém vytrácejícího se signálu je způsoben opakovaným násobením signálu malými hodnatami vah, což má za následek, že jsou hodnoty v nízkých vrstvách sitě blízké nule. ResNet využívá možnosti přeskočení určité části sítě, kde se ke vstupu přičte výstup z vyšší vrstvy sítě. Kromě vytrácejícího se signálu tato architektura také optimalizuje rychlost učení sítě, protože síť má možnost některé vrstvy obejít, a tím obnovuje sílu signálu v případě, že dojde k jeho vytrácení.



Obrázek 3.8: ResNet architektura [1].

Kapitola 4

Dataset

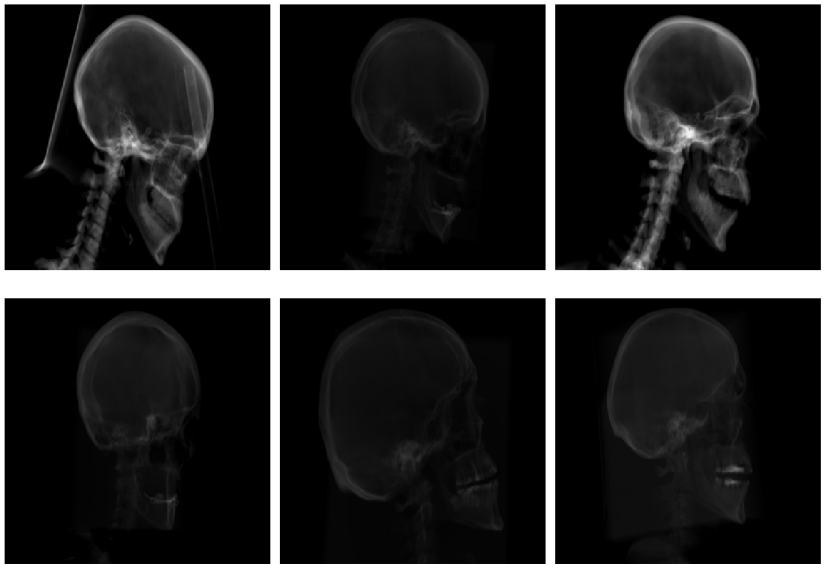
Dataset pro tuto úlohu byl vytvořen z CT snímků. Data jsou rozřazena do 6 kategorií podle úrovně rotace hlavy v okamžiku vyfocení snímku. Skutečné rentgenové snímky jsou součástí pouze kategorie, která značí nulovou nebo jen malou rotaci hlavy, protože pro medicínské účely obvykle nemá žádnou praktickou hodnotu uchovávat znehodnocené rentgenové snímky. Z tohoto důvodu bylo nutné zbytek kategorií vygenerovat z CT skenů celé hlavy. Celý dataset je rozdělen do šesti kategorií. Kategorie od -5 do 5 stupňů vychýlení značí validní snímek. Kategorie -10 až -5 a 5 až 10 znamenají, že se pacient drobně pohnul a snímek není ideální. Tyto kategorie jsou dvě, aby byl program schopen doporučit směr korekce. Zbylé kategorie jsou -10 až -20, 10 až 20 a zbytek. Poslední kategorie je tak rozsáhlá z toho důvodu, že podstata problému je zde naprosto zjevná, a proto je důležitá jen její detekce.

4.1 Generování rentgenových snímků

Pro generování rentgenových snímků z CT snímků byl použit skript, který byl vytvořen v rámci bakalářské práce slečny Dronzekové [9]. Tento skript byl upraven pro potřeby této práce. Program nejdříve načte obrázky, které jsou řezy vytvořené při CT skenu, v dicom formátu. Z těchto řezů se vytvoří 3D model celé hlavy a vykreslí se. K vytvoření podobnosti k rentgenovému snímku byl použit v původním skriptu volume ray casting a additive blending. Vykresleným modelem jde pohybovat, měnit jeho přiblížení a hlavně je možné jím rotovat. Toho využívám, když otočím model do výchozí rotace (ideální boční snímek) a na zmáčknutí tlačítka se vytvoří série snímků simulací rentgenu, kde je každý pootočen o daný počet stupňů. Snímky jsou uloženy do složky, která odpovídá jeho kategorii. Pro každou kategorii je vytvořen rotací modelu různý počet snímků. Počet snímků na kategorii je odhadnutý tak, aby zastoupení kategorie v celém datasetu odpovídalo procentuálnímu zastoupení v reálném nasazení programu. Tento systém není úplně přesný, například zastoupení kategorie 20 až 340 stupňů v datasetu je pravděpodobně větší, než bude zastoupení v reálném provozu, to je způsobeno tím, že pokud bych dodržel předpokládané zastoupení, při trénování neuronové sítě by se síť pravděpodobně s touto kategorií příliš nesetkávala, a proto by síť nejspíš nedokázala rozpoznat a správně zařadit v momentě, kdy by se se snímkem patřícím do této kategorie setkala.

Problémem je také nalezení ideální polohy s nulovým natočením. Takováto pozice není nijak snadno definovatelná, protože každý člověk má určitá specifika, co se týče tvaru lebky a poměru velikostí různých kostí, ze kterých se lebka skládá. Při generování jsem se řídil

pravidlem, že ideální rotace je taková, kde párové struktury lebky nejsou vidět dvakrát, ale překrývají se. To se samozřejmě nedá udělat dokonale, protože se některé překrývat mohou a jiné ne a neexistuje úhel, kde by se překrývaly všechny. Proto je stanovený ideální snímek pouze orientační a je nastavená určitá tolerance odchylky. Tuto toleranci definuje rozdělení datasetu na třídy podle intervalů.



Obrázek 4.0: Příklad snímků tříd -5 až 5, 5 až 10, 10 až 20, 20 až -20, -20 až -10, -10 až -5.

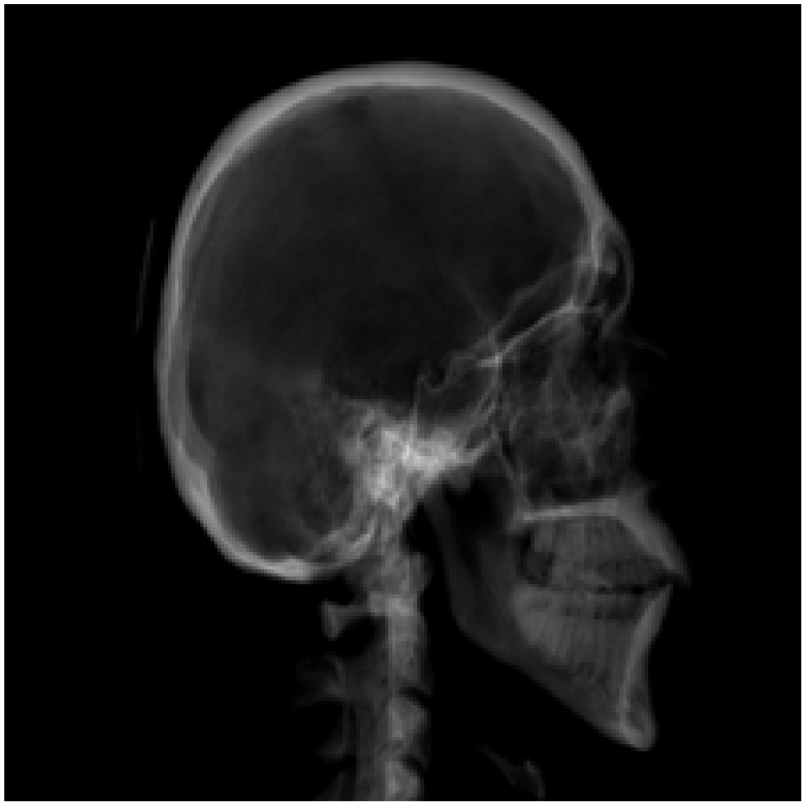
Obrázek 4.1: Porovnání skutečného snímku se snímkem vygenerovaným.

Volume ray casting

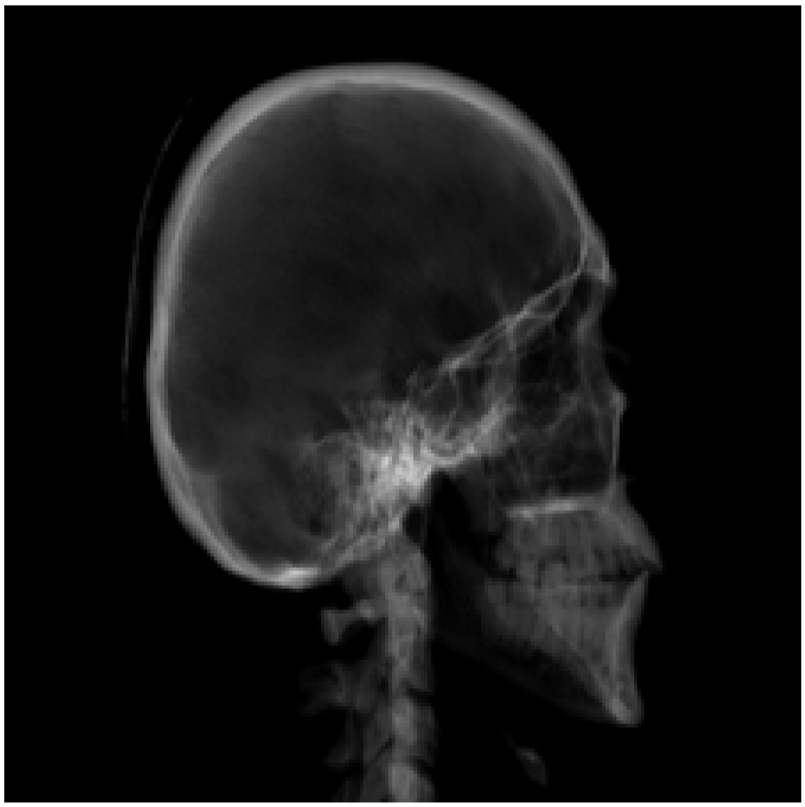
Volume ray casting je technika, která umožňuje projekci 3D modelu do 2D. Toto je využito na zobrazení naší simulace rentgenového obrázku z CT skenu, CT sken je zde ve formě následujících řezů, které dohromady tvoří 3D model. Volume ray casting je algoritmus, kde se z místa kamery, v našem případě je to pozice, ze které chceme zobrazit náš model, vysílají paprsky směrem k modelu, který chceme zobrazit. Tyto paprsky procházejí modelem a tím vznikají vzorky, které po součtu všech vzorků pro daný paprsek dají dohromady barvu. Tato barva je potom barvou, kterou bude mít pixel v zobrazovací rovině. Pro každý pixel se vysílá právě jeden paprsek. Tímto způsobem získáme barvu každého pixelu a tím i obraz, který chceme vykreslit.

Additive blending

Additive blending slouží k vytváření barvy pixelu z jednotlivých vzorků, které k pixelu náležejí. Je založeno na tom, že výsledná barva pixelu bude součtem barev všech vzorků. Právě v kombinaci s ray castingem byla vypočítaná výsledná barva každého jednotlivého pixelu.



Obrázek 4.1: Příklad generovaného snímku kategorie -5 až 5 stupňů.



Obrázek 4.2: Příklad generovaného snímku kategorie -10 až -20 stupňů.

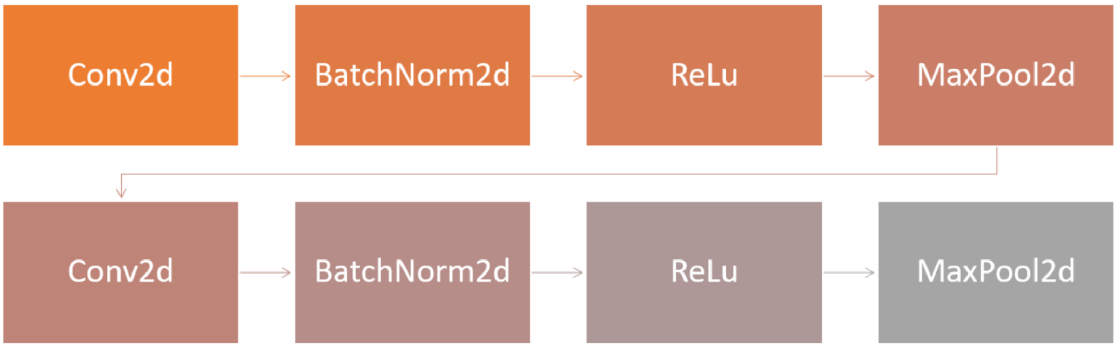
Kapitola 5

Návrh konvolučních neuronových sítí

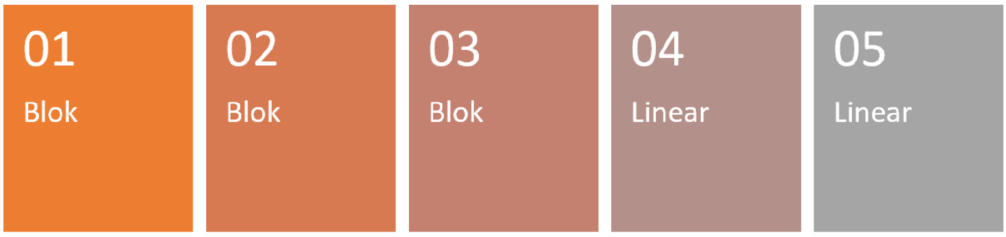
Jak jsem se již zmiňoval výše, použil jsem v této práci několik architektur založených na architekturách popsaných v podkapitole o architekturách konvolučních sítí. Budu zde popisovat, jak přesně byly upraveny a proč. Byly použity celkem 4 různé architektury.

5.1 VGG3

Tato síť je, jak název napovídá, implementací VGG sítě. Skládá se ze tří sériově zapojených bloků. Tyto bloky se skládají dvourozměrné konvoluční vrstvy, dvourozměrné batch normalizace, aktivační funkce ReLU a maximum pooling vrstvy, tato kombinace se v každém bloku opakuje dvakrát. Tyto tři bloky následují dvě plně propojené vrstvy. Zajímavé je také sledovat vývoj počtu kanálů v průchodu sítí. Do sítě vstupují 3 kanály, každý zde reprezentuje jednu z barev, které definují výslednou barvu pixelu. Vstupem do prvního bloku jsou tedy tři kanály, výstupem je kanálů osm. Osm kanálů je logicky vstupem do druhého bloku. Druhý blok rozšíří počet kanálů na 16 a třetí blok na 32. Následná lineární vrstva se zúží a poslední vrstva má již výstup velikosti, která odpovídá počtu tříd, do kterých klasifikuje. Nakonec se pro účely trénování použije pro výpočet chybové funkce CrossEntropy.



Obrázek 5.0: VGG blok.



Obrázek 5.1: VGG3 architektura.

Jak je vidět na obrázcích, tato síť není příliš hluboká, v praxi to znamená, že se učí rychleji a není tak náchylná k přetrénování jako rozměrnější sítě. Je to způsobeno tím, že je k dispozici menší počet vah, které lze nastavit. Na druhou stranu to znamená, že nebude rozpoznávat takové detaily jako hlubší sítě. Je tu riziko, že problém, který chceme, aby síť řešila, spočívá v tak drobných detailech, že se je nebude schopna naučit rozpoznávat.

5.2 VGG4

Tato síť je velmi podobná té předešlé. Obě jsou založené na návrhu VGGNet a její blok se skládá ze stejných částí jako VGG3. Liší se prakticky jen ve své hloubce, VGG4 se totiž skládá ze 4 bloků oproti VGG3, kde jsou jen 3. Tento fakt také ovlivňuje výsledný počet kanálů z posledního bloku. Počet kanálů u bloků je stejný jako u VGG3 s tím rozdílem, že poslední blok má na výstupu 64 kanálů. I zde je použita CrossEntropy funkce na výpočet chyby při trénování. Tuto architekturu používám jako mezistupeň mezi jednoduchou VGG3 sítí a hlubšími a komplikovanějšími sítěmi. Snažím se tím skloubit výhody obou.

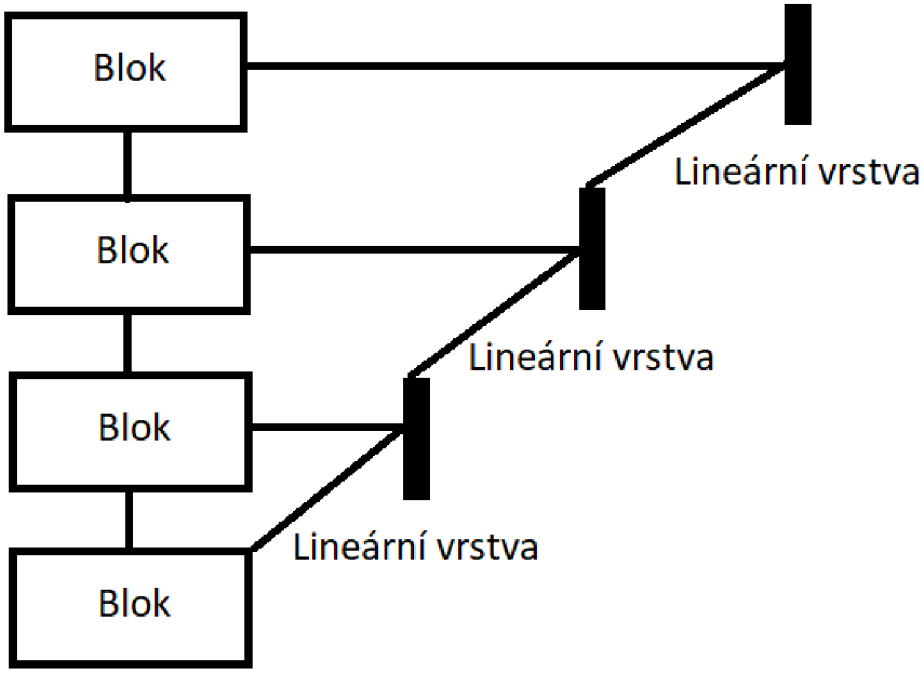
Obrázek 5.2: VGG4 architektura.

5.3 Half UNet

Half UNet architektura se inspiruje v UNet architektuře, kde se snaží využít zachování výstupů vyšších vrstev sítě, což se osvědčilo při práci s biomedicínskými daty. UNet byl navržen speciálně pro rozpoznávání lékařských snímků a Half UNet se snaží vlastnosti, které se pro tuto práci hodí, zachovat a zároveň architekturu modifikovat tak, aby co nejlépe vyhovovala našim účelům. Problémem UNet architektury a důvodem, proč ji zde nejde použít, je to, že velikost jejího vstupu odpovídá i výstupu. To je vhodné, pokud chceme, aby nám síť zvýraznila část obrázku nebo v něm našla nějaké objekty, ale to není náš případ. V této práci chceme, aby byl výstup ve tvaru zařazení do kategorie a tomu musí odpovídat

i výstupní vrstva sítě. Proto byla použita pouze část UNet architektury. Konkrétně jde o první část, která podvzorkovává vstup. Naopak část, která nadvzorkovává byla vynechána. Rozdíl oproti VGG modelu spočívá v tom, že výstupy z vyšších vrstev sítě jsou zachovány a přivedeny na vstup plně propojených dopředných vrstev, které se nacházejí za konvolučními vrstvami, společně se vstupem z bezprostředně předchozí vrstvy. Pro podvzorkování jsou použity stejné bloky jako ve sítích typu VGG.

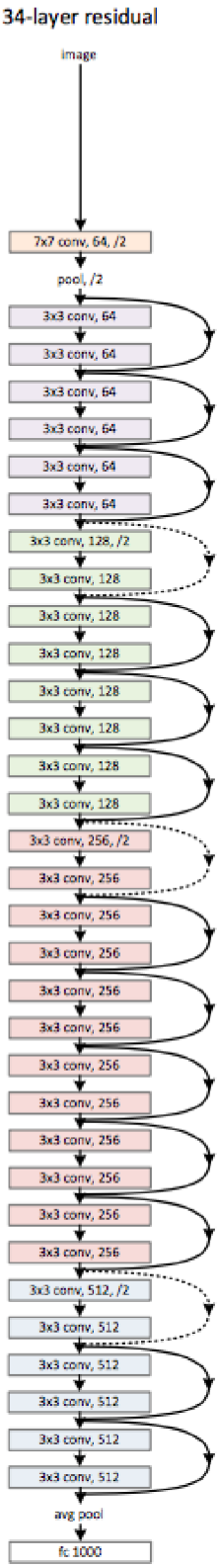
Jsou použity 4 bloky a 3 lineární vrstvy. Bloky jsou zařazeny za sebou a za nimi následují lineární vrstvy. Výstup prvního bloku je přiveden na vstup druhého a zároveň na vstup poslední lineární vrstvy. Stejným zpusobem je výstup druhého bloku přiveden vstup třetího bloku a druhé lineární vrstvy. Stejně je napojen i výstup třetího bloku. Výstup posledního bloku je napojen na vstup první lineární vrstvy.



Obrázek 5.3: Half UNet architektura.

5.4 Resnet

Resnet síť byl použita hlavně kvůli vlastnosti, kterou jí propůjčuje zapojení reziduálního bloku. Použitý dataset je relativně malý, a proto je u hlubokých sítí velké riziko přetrénování. Další obrovskou výhodou a zjednodušením implementace je, že součástí knihovny pytorch je již implementovaný model resnet architektury. Knihovna dokonce nabízí 6 resnet sítí, které se navzájem liší velikostmi. Dalším důvodem, proč jsem použil implementaci z PyTorch knihovny, je, že pokud chceme, můžeme použít již předtrénované modely. Předtrénování podstatně urychluje proces učení sítě. Předtrénování je technika, kde síť nejdříve trénujeme na jiném, často velmi rozsáhlém datasetu. Toto trénování sice nijak nezlepší přesnost klasifikace našeho problému, ale pokud má tento rozsáhlý dataset alespoň nějakou podobnost s naším datasetem, nebude na počátku trénování na našem datasetu nastavení vah naprosto náhodné, ale určitým způsobem již přednastavené. Právě předtrénováním na podobném datasetu můžeme podstatně urychlit proces trénování, protože váhy v síti bude potřeba upravit jen trochu.



Obrázek 5.4: Resnet architektura [2].

Kapitola 6

Implementace

V rámci implementace řešení této úlohy jsem navrhnul a natrénoval 4 architektury, které vycházejí z architektur popsaných ve třetí kapitole. Některé tyto architektury byly modifikovány, aby lépe vyhovovaly našim motivům. Například mnoho architektur konvolučních neuronových sítí navržených pro rozpoznávání obrazu je sestaveny způsobem, který na výstupu sestavuje opět obraz, jen pozměněný. To znamená, že výstupní vrstva je pro naše účely, kde potřebujeme pouze zařazení do jedné z kategorií, příliš široká. To znamená, že bylo potřeba tuto architekturu pozměnit, aby generovala takový tvar výsledků, který požadujeme. Toto je příklad změn, které byly potřeba na UNet architektuře, která je jednou z architektur, které jsem využíval.

Dalším problémem, jež bylo třeba vyřešit pro zdárné dokončení této práce, bylo příliš obecné navržení architektury. Mnoho architektur je spíše obecným návrhem, doporučením, jaké vrstvy by měly jít po sobě, aby konvoluční síť lépe fungovala. Poté bylo nutné vyzkoušet několik kombinací, najít ideální množství bloků a hloubku sítě, aby síť fungovala co možná nejlépe.

Důležitou roli při trénování neuronové sítě hraje mnoho faktorů, například learning rate je třeba velmi pečlivě volit, aby se síť správně trénovala. Pokud je learning rate příliš velký, síť nedokáže najít minimum chybové funkce a potom diverguje od správného výsledku. V případě, že je naopak learning rate příliš malý, hrozí, že se bude síť trénovat příliš dlouho, což je hlavně u složitých problémů komplikace při experimentování, Pokud se bude trénovat síť dvakrát pomaleji, podstatně se sníží počet experimentů, které jsme schopni uskutečnit za určité časové období. To výrazně prodlužuje čas vývoje, obvzlášť u sítí, které se trénují delší dobu. Některé rozsáhlé sítě se trénují až v rámci dnů, v tomto případě je každé zpomalení učení velmi citelné. Naštěstí to není náš případ, trénování sítí navržených v této práci je podstatně kratší, veškeré trénování se odehrálo v rámci desítek minut. Další komplikací moc malého learning rate je uváznutí v lokálním minimu, posun vah je příliš malý a síť nedokáže lokální minimum chybové funkce překlenout. Jakákoliv změna vah s takto nízkým krokem vyústí k dočasnému zhoršení klasifikační schopnosti sítě. To znamená, že síť ztrácí schopnost se učit a dále zlepšovat. Pokud k tomuto uváznutí dojde nemá, smysl se snažit dále trénovat síť, proto je vhodné učení ukončit a vyhodnotit výsledky. Je velmi obtížné zjistit, zda síť při trénování skončila v lokálním minimu nebo narazila na minimum globální. Způsobem, jak toto vyřešit, je zkusit trénování s větší hodnotou learning rate a sledovat, zda se bude síť tentokrát chovat lépe. To můžeme opakovat, dokud se buď výsledky nezlepší, nebo dokud nezískáme pocit, že jsme narazili na limity sítě.

Důležitou součástí trénovacích skriptů je také proměná, která určuje počet trénovacích epoch. Epocha v kontextu neuronových sítí znamená jednu iteraci trénování, kde je na

vstupu každý obrázek z batche. Počet trénovacích epoch je další promněná, kterou je třeba volit velmi opatrně. Na první pohled se zdá, že příliš trénování nemůže síti uškodit, maximálně se přestane zlepšovat, ale opak je pravdou. Pokud je počet epoch příliš vysoký, může docházet k přetrénování sítě, což má negativní důsledky na kvalitu zpracování validačního datasetu a úspěšnost sítě v použití v praxi. Přetrénovaní v tomto případě vzniká kvůli vystavení sítě během trénovací fáze příliš mnohokrát těm samým obrázkům. Naopak pokud bude počet epoch příliš malý, síť nebude mít dostatek možností se požadované vlastnosti naučit. To opět vyústí ve zhoršení přesnosti klasifikace.

6.1 Pomocné skripty

Během trénování různých architektur sítí jsem použil několik skriptů, které implementují funkce, které sice nemusí přímo souviset s neuronovými sítěmi, avšak jsou naprosto zásadní pro ulehčení jejich vývoje.

loadData

Tento skript implementuje jedinou funkci, která má dva parametry. Prvním je cesta ke složce s trénovacími daty, druhým je cesta ke složce s daty testovacími. Funkce obrázky načte a ke každému přiřadí odpovídající třídu podle čísla v názvu složky, ve které se nachází. V této funkci se také provádí zamíchání obrázků, aby při trénování nebyly obrázky stejné třídy bezprostředně za sebou, a také dochází k normalizaci hodnot barev jednotlivých pixelů, aby se síť nemusela potýkat s příliš vysokými hodnotami. Výstupem této funkce je čtveřice polí. Konkrétně jde o trénovací data, testovací data, labely trénovacích dat a labely testovacích dat. Labely jsou pole za sebou jdoucích čísel, které určují, do jaké třídy obrázek patří.

test

Skript test obsahuje opět jen jedinou funkci. Jak už název napovídá, tato funkce implementuje testování modelu. Model v tomto případě obsahuje architekturu konvoluční neuronové sítě a naučené váhy jednotlivých neuronů. Vstupem do této funkce výše zmíněný model, načtená testovací data a testovací labely. Funkce vyhodnotí testovací data pomocí dodaného modelu a každému obrázku přiřadí podle výsledu vyhodnocení třídu. Poté se porovnává vypočítaná třída snímku s třídou dodanou v poli testovacích labelů a vypočte se accuracy. Accuracy se následně vypíše do konzole. Stejně jako při trénování jsou i při testování data rozdělena do batchů. Zde je to z toho důvodu, že na vyhodnocení všech snímků naráz nebyl dostupný dostatečný hardware.

measureModel

Funkce tohoto skriptu je podobná jako u skriptu test, rozdíl je v tom, že tento skript načítá již uložený a natrénovaný model a měří přesnost klasifikace jednotlivých tříd. Takto zjistíme, jaké má natrénovaná síť silné a slabé stránky. Zároveň tak získáme představu, kde je hlavní příčina nepřesností vzniklých při učení.

resize

Resize je skript, který byl vytvořen pro účely vytvoření datasetu. Konkrétně jde o změnu velikosti každého snímku na dodané cestě, zároveň tento skript změní typ obrázku na PNG,

aby s ním mohl pracovat skript loadData. Je potřeba, aby měly všechny obrázky stejnou velikost, protože šířka určitých vrstev sítě musí odpovídat množství dat, které dodáme v jednom vstupu, to znamená, že musím již při návrhu sítě znát formát dat, která budeme používat, nebo musíme použít skript, jako je tento. V této práci měním velikost obrázků ještě před tím, než jsou načteny, abych zrychlil experimenty se sítěmi.

delete

Poslední skript, který zde chci zmínit, je delete. Tato funkce slouží k mazání souborů z datasetu, které mají špatný typ.

6.2 Použité technologie

Programovací jazyk

Jako programovací jazyk jsem zvolil Python. Python je vysokoúrovňový programovací jazyk, který umožňuje velmi efektivní práci s daty a jednoduché zobrazení různých grafů [5].

PyTorch

PyTorch je open-source knihovna navržená pro práci se strojovým učením. Knihovna byla vyvinuta výzkumným týmem Facebooku, který zkoumá právě umělou inteligenci. PyTorch nabízí jednoduchou práci neuronovými sítěmi a optimalizaci na grafické kartě. Důležitým typem proměnné, kterou PyTorch hojně využívá, je Tensor. Tensor je podobný poli v známé knihovně Numpy, ale lze s ním pracovat i na grafické kartě. V rámci této knihovny jsou již implementované funkce, které jsou naprosto zásadní pro výpočty v průběhu trénování. Zároveň zajišťuje snadnou spolupráci s knihovnou Numpy, kterou také využívám v této práci.

Numpy

Numpy je knihovna pro jazyk Python, která je byla vytvořena pro práci s poli. Operace v této knihovně jsou optimalizované. V této práci ji využívám pro jednoduché matematické operace nad celým načteným datasetem.

Matplotlib

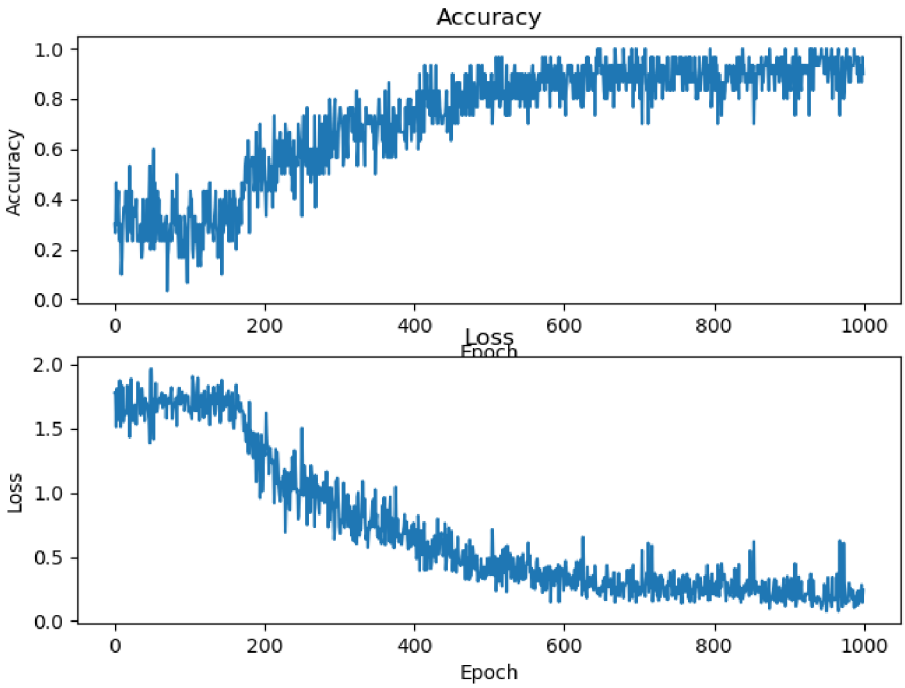
Matplotlib je knihovna, která implementuje funkce na vykreslení dat do grafů a histogramů. Tuto knihovnu využívám pro grafickou prezentaci výsledků a vizualizaci v průběhu vývoje této práce.

Kapitola 7

Experimenty a výsledky

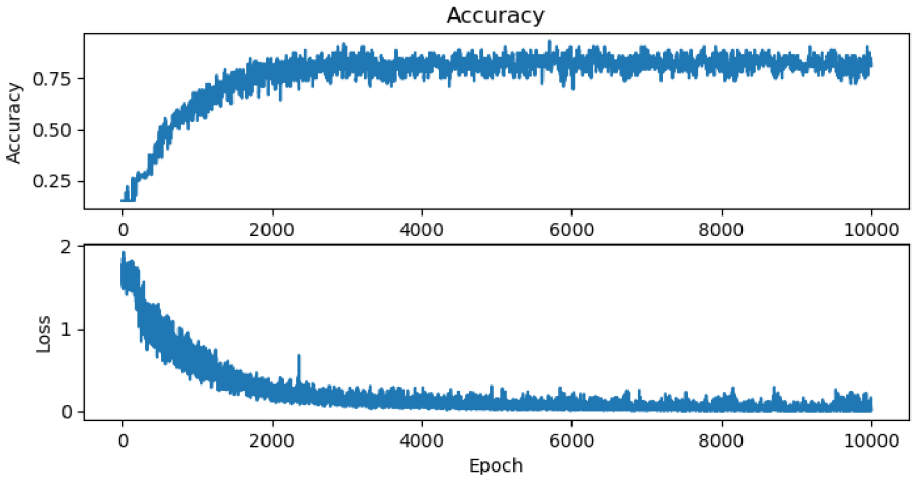
7.1 Experimenty

V této podkapitole jsou popsány způsoby a výsledky experimentů s implementovanými architekturami. Konkrétně jsem u každé architektury měnil promněné learning rate a počet epoch a sledoval jejich chování během trénování a testování. Sledoval jsem hlavně tři ukazatele, které mi napovídaly, kterým způsobem je vhodné síť pozměnit. Jedná se o vývoj hodnot chybové funkce a vývoj přesnosti na testovacích datech a výsledná přesnost na testovacích datech. Dohromady nám tyto hodnoty napoví, jestli není síť přeučená a jestli máme moc velký nebo malý learning rate. Nakonec víme, kterým směrem se máme vydat, abychom dokázali plně využít potenciál dané architektury. Například pokud máme stabilně vysokou přesnost při trénování a nízkou na datech testovacích, znamená to, že je síť přeučená a ztrácí schopnost generalizovat. To lze vyřešit třeba snížením počtu epoch. Nebo pokud se přesnost na testovacích datech dlouhodobě nelepší, může to znamenat, že máme příliš malý learning rate a síť uvízla v lokálním minimu chybové funkce.



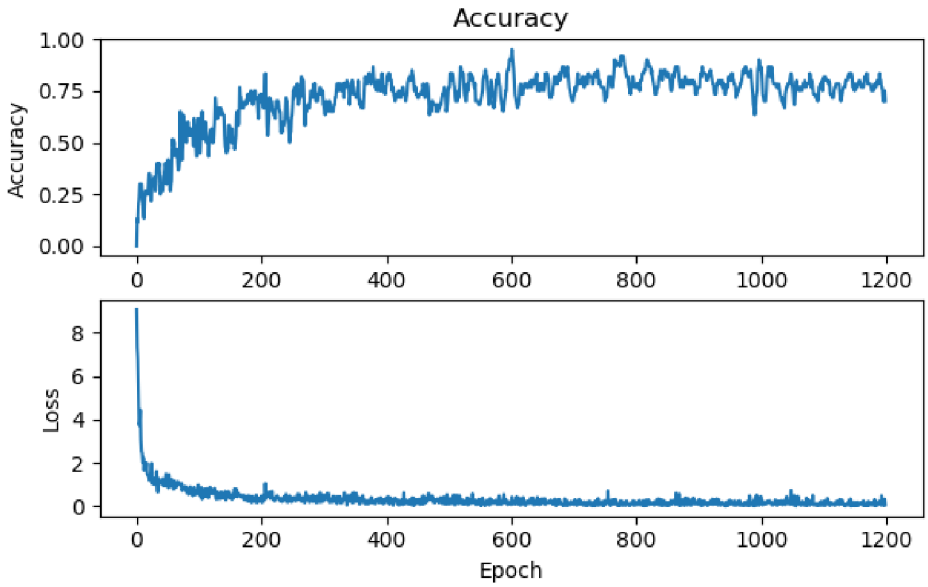
Obrázek 7.0: Graf experimentu s VGG3.

Například byla trénována síť VGG3 na 1000 epochách s learning rate nastaveným na 0.0005. Přesnost takto natrénované sítě na testovacích datech byla 0,875. Tento výsledek je poměrně dobrý a tomu odpovídají i ostatní naměřené hodnoty, jak je vidět na obrázku 7.0. Tento tvar křivek je téměř ideální.

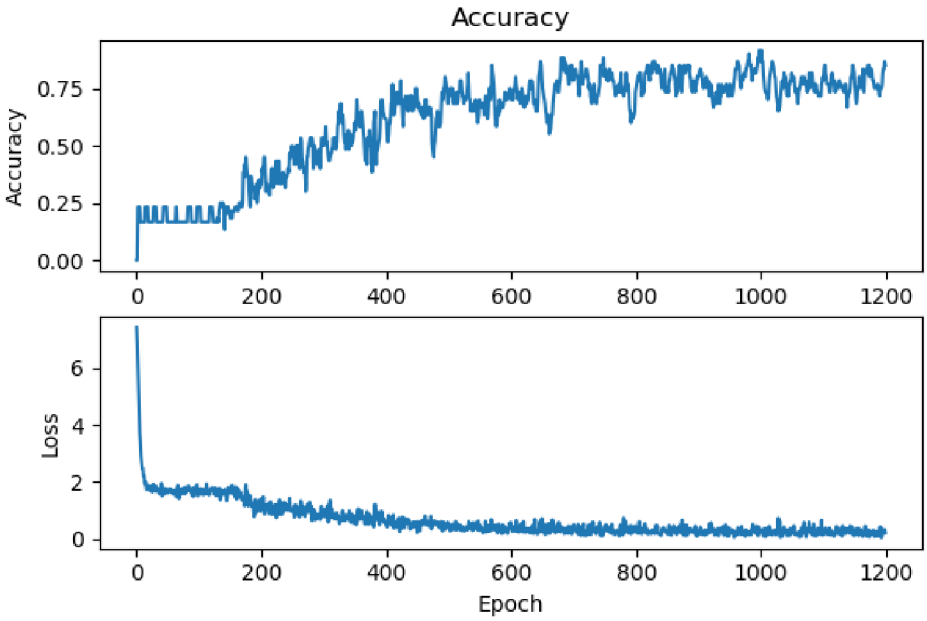


Obrázek 7.1: Graf experimentu s VGG3.

Na obrázku číslo 7.1 je zase vidět, že se výsledek trénování už nijak nezlepšuje většinu času, a proto je zde zbytečně mnoho epoch.



Obrázek 7.2: Graf experimentu s předtrénovaným Resnet



Obrázek 7.3: Graf experimentu s Resnet.

Poslední, co bych chtěl v této kapitole ukázat, je vliv předtrénování na učení sítě. Jak můžeme vidět na obrázku číslo 7.2, který znázorňuje vývoj trénování na předtrénované síti Resnet, zpočátku stoupá přesnost na trénovacích datech rychleji než na stejné síti, která předtrénována nebyla. Tuto síť reprezentuje obrázek 7.3.

7.2 Výsledky

V této kapitole budu prezentovat nejlépe naučené sítě od každé architektury a porovnávat jejich výsledky. Zároveň bych chtěl u každé sítě ukázat, v jakých kategoriích dělala nejvíce chyb. Měření přesnosti na kategoriích je prováděno na menším datasetu, takže součty přesností neodpovídají přesnosti na celém testovacím datasetu.

Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
-5 až 5	5 až 10	10 až 20	20 až -20	-20 až -10	-10 až -5

Tabulka 7.0: Rozdělení do tříd podle natočení.

Všechny hodnoty jsou uvedeny ve stupních a jednotlivé hodnoty určují zařazení snímku do třídy.

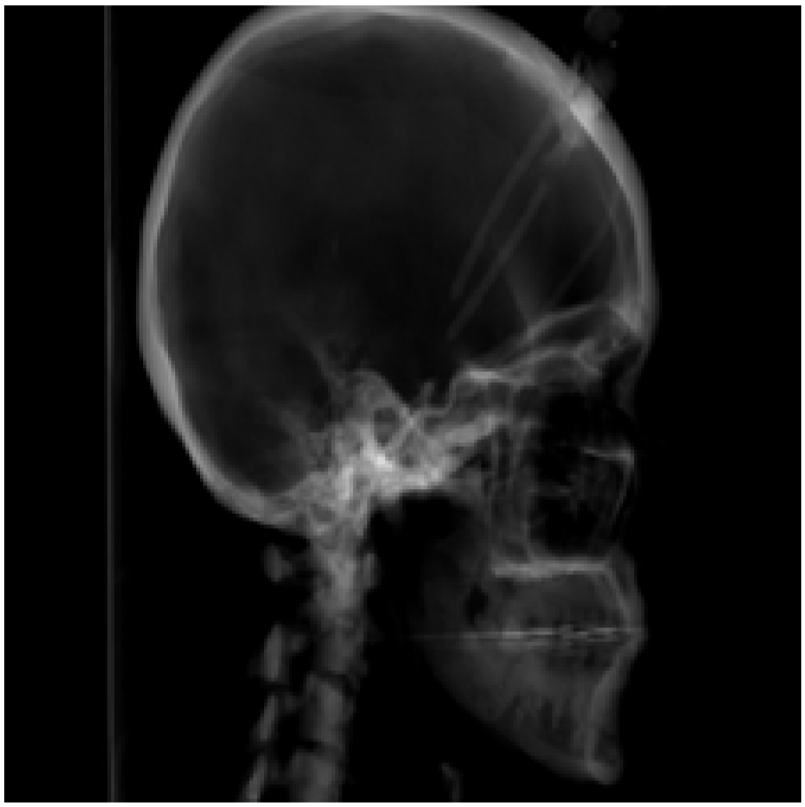
Výsledky VGG3

Nejlepší naměřená přesnost sítě, která používá VGG3 architekturu, na testovacích datech byla 96 procent.

VGG3	Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
Accuracy	94%	91%	94%	100%	99%	97%
Precision	83%	73%	84%	100%	100%	92%
Recall	95%	73%	79%	100%	93%	86%
F-measure	88%	73%	81%	100%	96%	89%

Tabulka 7.1: Metriky sítě VGG3 naměřené pro třídy.

Zde můžeme vidět přesnost natrénované sítě na jednotlivých třídách, je tedy zřejmé, že se síť plete v kategoriích 1 a 2. Tyto kategorie jsou zaměňovány za sebe navzájem. Tyto kategorie spolu sousedí. Není to tedy tak závažná chyba, jako záměna za třídu, která je na opačném konci, co se týče rotace.



Obrázek 7.4: Příklad špatně klasifikovaného snímku sítí VGG3.

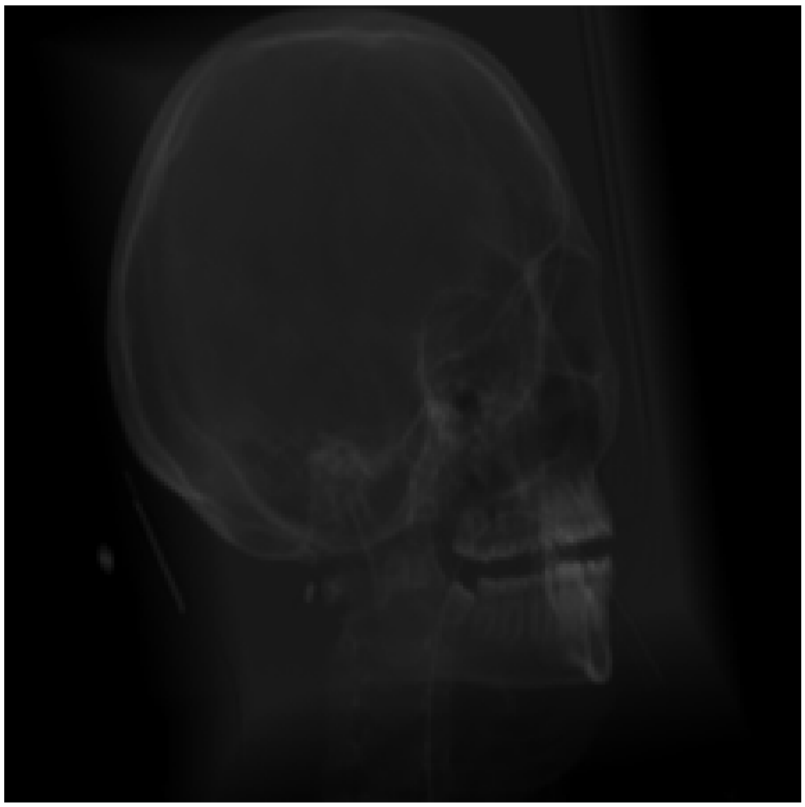
Výsledky VGG4

Tato síť dosáhla během testování přesnosti 97%. To je nejlepší výsledek, kterého jsem v rámci experimentování na všech implementovaných architekturách dosáhl.

VGG4	Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
Accuracy	93%	92%	97%	100%	99%	97%
Precision	77%	83%	88%	100%	100%	92%
Recall	94%	67%	93%	100%	93%	86%
F-measure	85%	74%	90%	100%	96%	89%

Tabulka 7.2: Metriky sítě VGG4 naměřené pro třídy.

Jak je vidět z tohoto měření na síti VGG4, síť se pletla hlavně v prvních třech kategoriích, které zároveň i mezi sebou zaměňovala.



Obrázek 7.5: Příklad špatně klasifikovaného snímku sítí VGG4.

Výsledky Half UNet

Half UNet architektura dosahuje z architektur, které jsem využil, nejhorších výsledků. Její přesnost na testovacích datech je 62%.

Half UNet	Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
Accuracy	66%	89%	97%	98%	87%	84%
Precision	36%	63%	100%	88%	100%	0%
Recall	84%	71%	77%	100%	20%	-
F-measure	50%	67%	87%	93%	33%	-

Tabulka 7.3: Metriky sítě Half UNet naměřené pro třídy.

V tabulce můžeme vidět, že většina špatně zařazených snímků je z tříd 1, 2 a 6. Síť tyto třídy mezi sebou zaměňuje. Tyto třídy jsou z pohledu rotace hlavy hned vedle sebe a je na nich nejjemnější rozlišení, se kterým si tato sít moc dobře neporadí. U kategorie nebyl zařazen správně ani jeden pozitivní obrázek. Z toho důvodu nebylo možné spočítat metriky jako je recall a f-measure.

Výsledky Resnet

Poslední měřenou sítí je Resnet. Resnet má přesnost na testovacím datasetu 90%. Stejně jako u ostatních architektur byla většina chyb u kategorií 1, 2 a 6.

Resnet	Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
Accuracy	97%	98%	100%	100%	95%	93%
Precision	93%	100%	100%	100%	80%	80%
Recall	93%	89%	100%	100%	89%	80%
F-measure	93%	94%	100%	100%	84%	80%

Tabulka 7.4: Metriky sítě Resnet naměřené pro třídy.

Porovnání

Nejlepších výsledků dosahuje síť VGG4 s VGG3. VGGNet se tedy osvědčil nejvíce. Naopak nejhorší v celkové přesnosti na testovacích datech byla síť Half UNet architektury, která za všemi ostatními podstatně zaostává.

VGG3	VGG4	Half UNet	Resnet
96%	97%	62%	90%

Tabulka 7.5: Přesnost jednotlivých sítí

Zajímavé je také porovnání přesnosti na jenotlivých třídach mezi jednotlivými sítěmi.

Accuracy	Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
VGG3	94%	91%	94%	100%	99%	97%
VGG4	93%	92%	97%	100%	99%	97%
Half UNet	65%	89%	97%	98%	87%	84%
Resnet	97%	98%	100%	100%	95%	93%

Tabulka 7.6: Porovnání accuracy mezi sítěmi a třídami.

Z toho je vidět, že nejproblémovějšími jsou kategorie 2 a 6, které si sítě pletou s kategorií 1. To je způsobeno tím, že v této oblasti natočení je rozdělení do tříd v malých intervalech. To znamená, že jsou si snímky v různých kategoriích velmi podobné a dochází tak k záměně. Například rozdíl v natočení mezi kategoriemi může být jeden stupeň a to může spadat do chyby s anotování snímků.

Precision	Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
VGG3	82%	73%	85%	100%	100%	92%
VGG4	77%	83%	88%	100%	100%	92%
Half UNet	36%	67%	100%	88%	100%	0%
Resnet	93%	100%	100%	100%	80%	80%

Tabulka 7.7: Porovnání precision mezi sítěmi a třídami.

Precision určuje, s jakou přesností rozpoznáváme pozitivní obrázky dané kategorie. Obecně můžeme vidět, že největší problémy mají sítě s rozpoznáváním tříd 1 a 2.

Recall	Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
VGG3	95%	73%	79%	100%	93%	86%
VGG4	94%	67%	93%	100%	92%	86%
Half UNet	84%	71%	77%	100%	20%	-
Resnet	93%	89%	100%	100%	89%	80%

Tabulka 7.8: Porovnání recall mezi sítěmi a třídami.

Z této tablky je zřejmé, že sítě mají největší problém s zachytáváním snímků, patřících do tříd 2 a 6.

F-measure	Třída 1	Třída 2	Třída 3	Třída 4	Třída 5	Třída 6
VGG3	88%	73%	81%	100%	96%	89%
VGG4	85%	74%	90%	100%	96%	89%
Half UNet	50%	67%	87%	94%	33%	-
Resnet	93%	94%	100%	100%	84%	80%

Tabulka 7.9: Porovnání f-measure mezi sítěmi a třídami.

F-measure odpovídá zjištěním z předchozích metrik. Je to dané hlavně tím, že je vypočítán jako jejich kombinace. Dává nám tak komplexnější pohled na schopnosti sítí.

Binární klasifikátor

Vzhledem k potencionálnímu využití těchto sítí pro klasifikaci rentgenových snímků je důležité si uvědomit, že důležitější než doporučení korekce je, zda je snímek vhodný pro kefalometrickou analýzu. Pokud budeme zkoumat výsledky sítě z tohoto hlediska, zajímá nás jen s jakou přesností síť zařazuje snímky do třídy číslo 1. Tato třída totiž odpovídá snímku, kde je hlava ve správné pozici. Proto můžeme interpretovat výsledek implementovaných sítí jako pozitivní v případě zařazení do třídy 1 a negativní pro všechny ostatní. Tímto způsobem vytvoříme simulaci binárního klasifikátoru. Samozřejmě je to pouze simulace, a proto nelze naměřit určité metriky, které bývají typické. Jedná se hlavně o ROC křivku.

Třída 1	Accuracy	Precision	Recall	F-measure
VGG3	94%	83%	95%	88%
VGG4	93%	77%	94%	85%
Half UNet	66%	36%	84%	50%
Resnet	97%	93%	93%	93%

Tabulka 7.10: Porovnání metrik mezi sítěmi na binárním klasifikátoru.

Z této tabulky je zřejmé, že většina sítí dokáže spolehlivě řešit i tento problém. Výjimkou je Half UNet, která má s tímto rozlišením větší problémy.

Celkové výsledky

Z výše uvedených tabulek se dá říct, že nejlépe fungovaly sítě VGG, mezi kterými je jen minimální rozdíl. Naměřená celková přesnost je pro VGG3 96% a pro VGG4 je to 97%. Obě sítě se chovají velmi podobným způsobem. Resnet má také velmi dobré výsledky (90%), nejlépe ze všech sítí rozpoznává třídu 1 (-5 až 5 stupňů), kde dosahuje 97%. Záleží tedy na konkrétním využití, kterou síť nebo jejich kombinaci využít.

Kapitola 8

Závěr

Cílem této práce bylo vytvořit program založený na konvolučních neuronových sítích, který dokáže klasifikovat rentgenové boční snímky hlavy určené pro kefalometrickou analýzu. Klasifikace probíhá tím způsobem, že program rozpozná, zda je snímek vhodný a v případě, že není, navrhne změnu polohy hlavy pro nápravu. Doporučení na nápravu se odvozuje z toho, do jaké kategorie byl snímek sítí zařazen. Nesnažím se tedy zjistit přesnou rotaci hlavy, ale stačí mi rozmezí několika stupňů.

Abychom dosáhli tohoto efektu, byl dataset rozdělen do 6 tříd, kde každá odpovídá rozmezí natočení hlavy. Pro stupně natočení, kam jsem očekával, že bude spadat naprostá většina klasifikovaných snímků, jsem navrhl podstatně jemnější rozdělení intervalů než pro stupně, u kterých očekávám, že se budou vyskytovat jen v naprostém minimu případů. Veškeré snímky byly také rozděleny na dataset trénovací a testovací. Trénovací dataset má celkem ve všech kategoriích 1867 snímků. Testovací dataset je o velikosti 90 snímků. Snímky byly vygenerovány z CT skenů kvůli nedostatku volně dostupného materiálu, který by odpovídal potřebám vývoje tohoto programu. Dataset, na kterém byly sítě trénovány, je také nejspíše nejslabším místem navrženého řešení. Pokud by byly dostupné skutečné rentgenové snímky, které by pokryly všechny kategorie, byl by výsledek pravděpodobně lepší.

Celkem jsem použil 3 architektury, které byly navrženy na klasifikaci obrazových dat. Z těchto architektur jsem vytvořil 4 sítě, se kterými jsem poté experimentoval za účelem dosáhnutí co nejlepších výsledků. Dvě sítě jsou typu VGGNet, jedna je založena na architektuře UNet a poslední je typu Resnet. Síť založená na UNet byla pozměněna, aby výstup odpovídal požadavkům zadání, a polovina původní architektury byla odstraněna. V rámci této zprávy ji nazývám Half UNet. Sítě založené na architektuře VGGNet se liší v počtu bloků, které jsou použity. Blokem je v této architektuře nazývána sekvence vrstev, které se opakují. Po experimentech s trénováním implementovaných sítí jsem dospěl u každé sítě k jisté přesnosti. U VGG3 je to 96%, VGG4 klasifikuje s přesností 97% a Resnet s přesností 90%. Nejhůře řeší zadaný problém síť Half UNet, která dosahuje přesnosti pouhých 60%. Výsledky na poslední jmenované síti by šlo pravděpodobně zlepšit předtrénováním.

Přesnost jednotlivých sítí by šlo zlepšit vytvořením většího datasetu. Také by pomohlo, pokud by byly sítě trénovány na reálných datech místo uměle vygenerovaných.

Další zajímavý směr, kterým by se dalo vydat při dalším vývoji, je vyvinutí programu, který by úhly natočení hlavy přímo počítal místo pouhého řazení do intervalů. To by se poté dalo použít v kombinaci s mým řešením a přesnost výsledku ještě navýšit. Zároveň by to vedlo ke zpřesnění výsledku a mé řešení by mohlo sloužit k validaci tohoto měření.

Dosažené přesnosti však hodnotím pozitivně a to hlavně kvůli tomu, že chybně zařazené snímky byly klasifikovány do kategorií, které intervalově přímo sousedí se správnou. Proto nebudou dopady špatné klasifikace v praxi tak výrazné.

Literatura

[1] URL https://stats.stackexchange.com/questions/330223/how-resnet-increasing-the-dimension

[2] Advanced Deep Learning Architectures. [Online; navštíveno 03.05.2019]. URL https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/

[3] Layers of a Convolutional Neural Network. [Online; navštíveno 05.04.2019]. URL https://wiki.tum.de/display/lfdv/Layers+of+a+Convolutional+Neural+Network/#LayersofaConvolutionalNeuralNetwork-PoolingLayer

[4] Neural Networks Explained. [Online; navštíveno 03.05.2019]. URL https://www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html

[5] Python Documentation. [Online; navštíveno 10.01.2018]. URL https://www.python.org/

[6] ROC curve analysis. [Online; navštíveno 05.04.2019]. URL https://www.medcalc.org/manual/roc-curves.php

[7] Specify Layers of Convolutional Neural Network. [Online; navštíveno 05.04.2019]. URL https://www.mathworks.com/help/nnet/ug/layers-of-a-convolutional-neural-network.html

[8] Al-barazanchi, H.; Qassim, H.; Verma, A.: Residual CNDS. 08 2016.

[9] Dronzeková, M.: Hluboké neuronové sítě pro analýzu medicínských obrazových dat. 2018, vedoucí: Ing. Michal Španěl, Ph.D. https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=181098.

[10] He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. 2015, arXiv:1512.03385.

[11] He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. 2015, arXiv:1512.03385.

[12] Ioffe, S.; Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015, arXiv:1502.03167.

[13] Škandera Juraj: Detekce významných bodů v medicínských obrazech pomocí hlubokých neuronových sítí. 2018, vedoucí: Ing. Oldřich Kodym http://www.fit.vutbr.cz/study/DP/BP.php?id=21193&file=t.

[14] Maschtakow, P. A.-c. S. L.; Tanaka, J. L. O.; Rocha, J. A. C. d.; aj.: Cephalometric analysis for the diagnosis of sleep apnea: a comparative study between reference values and measurements obtained for Brazilian subjects. *Dental Press Journal of Orthodontics*, ročník 18, 06 2013: s. 143 – 149, ISSN 2176-9451.
URL http://www.scielo.br/scielo.php?script=sci_arttext&pid=S2176-94512013000300023&nrm=iso

[15] Ronneberger, O.; Fischer, P.; Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015, arXiv:1505.04597.

[16] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014, arXiv:1409.1556.

[17] Sorower, M. S.: A Literature Survey on Algorithms for Multi-label Learning. 2010.
URL https://pdfs.semanticscholar.org/6b56/91db1e3a79af5e3c136d2dd322016a687a0b.pdf

[18] Červíček, K.: ALGORITMICKÉ OBCHODOVÁNÍ NA BURZE S VYUŽITÍM UMĚLÝCH NEURONOVÝCH SÍTÍ. 2016, vedoucí: Ing. Igor Szőke, Ph.D. https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=132171.

Příloha A

Obsah přiloženého paměťového média

Adresář Latex obsahuje kódy k vytvoření technické zprávy.
Adresář Scripts obsahuje zdrojové kódy k pomocným skriptům.
Adresář models obsahuje natrénované modely neuronových sítí.
Skript VGG3.py osahuje kód pro vytvoření a a natrénování VGG3 sítě.
Skript VGG4.py osahuje kód pro vytvoření a a natrénování VGG4 sítě.
Skript HalfUNet.py osahuje kód pro vytvoření a a natrénování HalfUNet sítě.
Skript Resnet.py osahuje kód pro vytvoření a a natrénování Resnet sítě.
Soubor poster.pdf obsahuje vytvořený plakát.

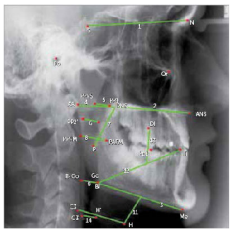
Příloha B

Plakát

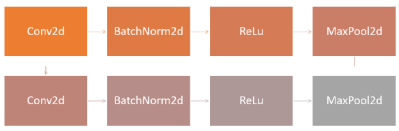
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Hluboké neuronové sítě pro analýzu medicínských obrazových dat

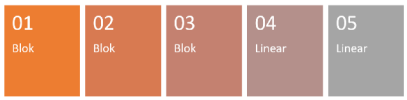
Autor: Tomáš Bíl
Vedoucí: Michal Španěl



Úkolem je zjistit výchylku v natočení hlavy pacienta z rentgenových snímků. Pokud je výchylka příliš velká, snímek již není vhodný pro kefalometrickou analýzu. Nejlepších výsledků dosahovala VGG architektura.



Architektura bloku



Architektura sítě

Dataset byl vygenerován z CT snímků, kvůli nedostatku volně dostupných vhodných rentgenových snímků. Dataset byl rozdělen do 6 tříd podle úrovně natočení. Síť se poté učila tyto třídy rozpoznávat.

Nejlepší dosažená přesnost na testovacím datasetu činí 97%. Nejvíce si síť pletla třídu -5° až 5° natočení oproti ideálnímu úhlu s třídami, které s ní bezprostředně sousedí.