Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ 2017 Tomáš Kolář

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš KOLÁŘ**
Osobní číslo:	**A15N0108P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Název tématu:	**Odhad pozice v těle pacienta ve snímcích získaných pomocí trojrozměrných zobrazovacích metod**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování:

1. Navrhněte metodu pro hrubé určení pozice v těle pacienta ve snímcích z trojrozměrných lékařských zobrazovacích metod. Pozici určujte vzhledem k významným anatomickým strukturám. Pro řešení využijte konvoluční neuronovou síť (CNN).
2. Rozeberte dostupná řešení konvolučních neuronových sítí a jedno zvolte.
3. Navrženou metodu porovnejte s dosavadními metodami pro určování pozice.

Rozsah grafických prací: **dle potřeby**

Rozsah kvalifikační práce: **40-50 stránek A4**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

NEKULA, Josef; HEŘMAN, Miroslav, et al. Radiologie. 3. vyd. Olomouc : Univerzita Palackého v Olomouci, 2005. Dotisk 2008. ISBN 978-80-244-1011-7. S. 205. (cs)
Sonka M., Hlavac V. , Boyle R.: Image Processing, Analysis, and Machine Vision, 3rd edition, Thomson Learning, Toronto, April 2007, 821 p, ISBN 049508252X (2nd edition Brooks/Cole, Pacific Grove, CA, 1999, 1st edition Chapman & Hall, London 1993, 4th edition scheduled for 2013).

Vedoucí diplomové práce: **Ing. Miroslav Jiřík**
Nové technologie pro informační společnost

Datum zadání diplomové práce: **3. října 2016**

Termín odevzdání diplomové práce: **21. května 2017**

Doc. RNDr. Miroslav Lávička, Ph.D.
děkan

L.S.

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 3. října 2016

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 28. 8. 2017 _____

podpis

Poděkování

Děkuji Ing. Miroslavu Jiříkovi Ph. D, vedoucímu této diplomové práce, za jeho trpělivost a rady.
Tato práce vznikla za podpory projektů CERIT Scientific Cloud (LM2015085) a CESNET (LM2015042) financovaných z programu MŠMT Projekty velkých infrastruktur pro VaVaI.

Abstrakt

Cílem této práce je za pomoci konvolučních neuronových sítí určit polohu řezu, na snímcích z počítačové tomografie, v těle pacienta, vzhledem k výrazným anatomickým strukturám v lidském těle.

Klíčová slova: konvoluční neuronová síť, keras, python, klasifikace, počítačová tomografie, augmentace,

Abstract

The object of this thesis is to estimation position in a patient's body from images from computed tomography, due to significant anatomic structure.

Keyworlds: convolutional neural network, CNN, keras, python, classification, computed tomography, augmentation

Obsah

1 Úvod **1**

1.1 Výpočetní tomografie . . . 2

1.1.1 Radonova transformace . . . 3

1.2 Počítačové vidění . . . 4

1.3 Neuronové sítě . . . 5

1.3.1 Historie neuronových sítí . . . 5

1.3.2 Biologický neuron . . . 7

1.3.3 Umělý neuron . . . 8

1.3.4 Učení neuronových sítí . . . 9

1.3.5 Dopředná neuronová síť . . . 10

1.3.6 Rekurentní neuronové sítě . . . 10

1.3.7 Konvoluční neuronové sítě . . . 11

2 Využívané technologie **13**

2.1 Git . . . 13

2.2 Python . . . 14

2.3 Lisa . . . 15

2.4 Dostupné implementace konvolučních neuronových sítí pro Python . . . 16

2.4.1 Caffe . . . 16

2.4.2 Theano . . . 16

2.4.3 Tensorflow . . . 17

2.4.4 Keras . . . 17

2.4.5 Torch . . . 18

2.4.6 Lasagne . . . 18

2.4.7 DeepLearning4j . . . 18

2.5 Použitá data . . . 19

2.5.1 SLIVER07 . . . 19

2.5.2 IRCAD . . . 20

2.6 Augmentace dat . . . 21

3 Návrh metody určení umístění řezu vzhledem k jaternímu parenchymu **22**

3.1 Použitá augmentace . . . 23

3.2 Kontext řezu . . . 25

3.3 Vstupní data 26
3.4 Příprava dat 27

4 Experiment **28**
4.1 Počet filtrů 29
4.2 Velikost filtrů 31
4.3 Počet konvolučních vrstev 33
4.4 Počet kompletních vrstev 35
4.5 Počet neuronů v plně propojené vrstvě 37
4.6 Výsledná neuronová síť 39

5 Závěr **41**

Seznam obrázků

1 Vyšetření pomocí CT 2
2 Radonova transformace 3
3 Struktura biologického neuronu 7
4 Jednoduchý model neuronu 8
5 Dopředná neuronová síť 10
6 Dopředná neuronová síť 10
7 Řez z datasetu IRCAD 22
8 Řez s provedenou augmentací 23
9 Vstupní data pro využití kontextu 25
10 Vstupní data pro využití kontextu s augmentací 26
11 Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu filtrů 29
12 Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu filtrů. 0 metoda není využita, 1 metoda je využita 30
13 Přesnost trénování vzhledem k počtu filtrů, po jednotlivých částech dat. . . 30
14 Přesnost klasifikace vzhledem k augmentaci, kontextu a velikosti filtrů. 0 metoda není využita, 1 metoda je využita 31
15 Přesnost klasifikace vzhledem k augmentaci, kontextu a velikosti filtrů. 0 metoda není využita, 1 metoda je využita 32
16 Přesnost trénování vzhledem k velikosti filtrů, po jednotlivých částech dat . 32

17 Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu vrstev. 0 metoda není využita, 1 metoda je využita 33
18 Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu vrstev. 0 metoda není využita, 1 metoda je využita 34
19 Přesnost trénování vzhledem k počtu vrstev, po jednotlivých částech dat . 34
20 Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu kompletních vrstev. 0 metoda není využita, 1 metoda je využita 35
21 Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu kompletních vrstev. 0 metoda není využita, 1 metoda je využita 36
22 Přesnost trénování vzhledem k augmentaci, kontextu a počtu kompletních vrstev, po jednotlivých částech dat 36
23 Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu a počtu neuronů v propojené vrstvě. 0 metoda není využita, 1 metoda je využita . . . 37
24 Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu a počtu neuronů v propojené vrstvě. 0 metoda není využita, 1 metoda je využita . . . 38
25 Přesnost trénování vzhledem k augmentaci, kontextu a počtu neuronů v propojené vrstvě, po jednotlivých částech dat 38
26 model neuronové sítě 39
27 Přesnost trénování vzhledem k augmentaci a kontextu pro konečný model neuronové sítě. 0 metoda není využita, 1 metoda je využita 40

1 Úvod

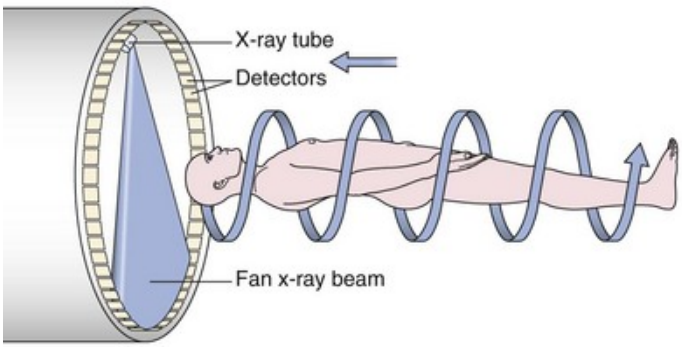
Cílem této diplomové práce je navrhnout, implementovat a otestovat program určený pro klasifikaci řezu trojrozměrného obrazu, pořízeného při vyšetření, prováděném za pomoci výpočetní tomografie, dle jeho pozice v těle, vzhledem ke zvolenému významnému orgánu. Samotná implementace bude provedena za pomoci konvolučních neuronových sítí. Implementovaný program je určen jako doplněk programu Lisa(LIver Surgery Analyser) vyvíjeného na katedře kybernetiky Západočeské univerzity v Plzni pod vedení Ing. Miroslava Jiříka, za přispění studentů katedry. Navržený algoritmus je veřejně k dispozici na GIThubu vedoucího(www.github.com/mjirik) v repozitáři metalisa. Tento program je určen pro segmentaci jater z obrazů pořízených pomocí trojrozměrných medicínských zobrazovacích metod, hlavně za pomoci výpočetní tomografie.

Funkce implementovaného programu je klasifikace vstupního řezu snímku z výpočetní tomografie, vzhledem k určenému orgánu. Klasifikace je prováděna do tří tříd. Jednotlivé třídy reprezentují stavy, kdy řez je pod orgánem, řez je v orgánu a řez je nad orgánem. Díky této klasifikaci je možné určit na vstupní trojrozměrném snímku rozsah orgánu postupně v jednotlivých osách a získat tak výsledný obalový kvádr celého orgánu. Tím je pak možné výrazně snížit výpočetní náročnost a zjednodušit práci obsluze při segmentaci jater, v případě že jako orgán játra zvolíme, pomocí programu Lisa, díky zmenšení oblasti v které se segmentuje z celého trojrozměrného obrazu na menší kvádrovou oblast.

Úvodní část této diplomové práce je věnována teoretickému podkladu, týkajícího se počítačového vidění neuronových sítí a získávání dat, následující část popisuje metodiku použitou při tvoření práce. Třetí část obsahuje popis experimentu s jednotlivými grafy, následující část pak jednotlivé výsledky experimentů shrnuje.

1.1 Výpočetní tomografie

Vstupní data pro námi řešenou úlohu jsou výsledkem vyšetření pomocí výpočetní tomografie. Výpočetní tomografie[9], taktéž známá jako CT, je lékařská 3D zobrazovací metoda. Zařízení využívá, podobně jako rentgen, rentgenové záření. Hlavní rozdílem oproti rentgenu je, že výsledkem vyšetření není jeden snímek, ale několik řezů spojených do 3D snímku těla. Vyšetření probíhá postupným zasouváním lehátka s pacientem do prstence přístroje. V prstenci je na jedné straně umístěn zdroj radiačního rentgenového záření, rentgenka. U rentgenových paprsků dochází při průchodu různých tkání v těle ke ztrátě jejich intenzity. Největší ztráty intenzity vznikají při průchodu záření skrz kosti, nejmenší pak při průchodu vzduchem. Typické hodnoty denzity v CT snímku jsou -1024 pro vzduch, 0 pro vodu, +1024 pro kosti, -60 až -120 pro svaly a 0 až 100 pro orgány. Na opačném konci prstence je umístěno několik snímačů rentgenových vln, které měří intenzitu vln, procházejících tělem. Prstenec kolem pacienta rotuje a snímá tedy velké množství intenzit pro každý řez. V původních verzích CT byl pokaždé pacient posunut o určenou vzdálenost a prstenec se kolem něj jednou otočil. V moderních přístrojích již dochází k nepřetržitému pohybu lehátka s pacientem, jak je viděl na obrázku 1.

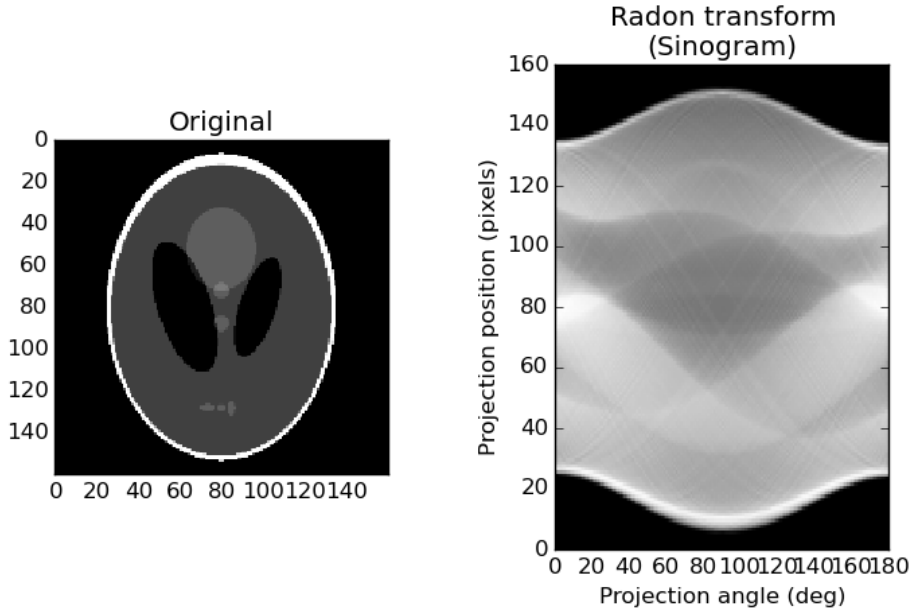


Obrázek 1: Vyšetření pomocí CT
Zdroj: http://clinicalgate.com

Výsledný 3D snímek je pak vypočítán, proto je CT označováno jako výpočetní tomografie a nikoliv špatně počítačová tomografie, pomocí radonové transformace z jednotlivých intenzit sejmutých z detektorů během vyšetření. Je nejvhodnější pro poúrazová vyšetření, protože dobře zobrazuje čerstvá krvácení.

1.1.1 Radonova transformace

Radonova transformace[4] je matematická transformace, která se používá pro získání výsledného snímku z jednotlivých zachycených intenzit. Pro tento výpočet se používá její zpětná varianta. Tuto transformaci popsal v roce 1917 Johann Radon. Z nastavení CT je známý úhel o který je v každém kroku, mezi zaznamenáním jednotlivých intenzit na detektorech, pootočen prstenec. Dále jsou k dispozici jednotlivé intenzity, které byly v těchto krocích zaznamenány. V pravé části na obrázku 2 jsou vidět jednotlivé intenzity zaznamenané detektory, jako sloupcové vektory. Následně je vytvořen integrální obraz. Z každého vektoru intenzit je vytvořen obraz, přes celou plochu, složením stejného vektoru vedle sebe, ten je následně otočen o úhel ve kterém byl vektor sejmut detektory. Takto se sečtou všechny obrazy pro vektory získané z jednoho otočení prstence. Následně je tento integrální obraz zprůměrován a výsledný obraz odpovídá levému části na obrázku 2



Obrázek 2: Radonova transformace

Zdroj: http://scikit-image.org

1.2 Počítačové vidění

Klasifikace je jednou z metod počítačového vidění. Tento obor vznikl v šedesátých letech minulého století. Počítačové vidění je jednou z oblastí umělé inteligence a má mnoho různých oborů. Úlohou klasifikace je ze vstupního obrazu rozeznat a vybrat jednu z možných tříd do které patří. Jeden ze známých příkladů klasifikace je postaven na takzvané MNIST databázi ručně psaných čísel. V této databázi je 70 000 ručně psaných čísel, každé v rozlišení 20 na 20. Úkolem je klasifikovat určit jaké číslo je na obraze.

Další oblastí počítačového vidění je segmentace obrazu. Cílem této úlohy je ve vstupním obraze nalézt takové části, které odpovídají jednotlivým předmětů v reálném světě, které byly na snímku zachyceny. Jednou z nejlehčích úloh segmentace je prahování. Je to metoda segmentace závislá na jasu. Je používaná pro oddělení pozadí a popředí v zachycené obraze.

Detekce je další oblastí počítačového vidění. V určitých případech potřebujeme detekovat v obraze určený předmět. Jedním z úkolů na detekci je detekce obličeje, případně předmětu určených specifikací, například v aplikacích řízení výrobních procesů. Další úlohou, která částečně vychází z detekce, je sledování. V případě této úlohy musí být vstupem více obrazů.

V současnosti jsou jedním z nejpokrokovějších oborů počítačového vidění konvoluční neuronové sítě. Konvoluční neuronové sítě jsou v dnešní době schopné v určitém smyslu zastoupit všechny ostatní obory počítačového vidění. Například konvoluční neuronová sít pro klasifikaci obrazů z MNIST databáze je dne schopna klasifikovat s přesností přes 99%. Konvoluční neuronové sítě jsou dnes prakticky největším odvětvím celého oboru umělé inteligence a počítačového vidění.

1.3 Neuronové sítě

Neuronové sítě jsou součástí oboru umělé inteligence. Jejich hlavní myšlenka je postavena na funkci reálných neuronů v mozku. Za dobu jejich vzniku je považován rok 1943, kdy byl popsán první model umělého neuronu. Dlouhou dobu byly pokládány neuronové sítě za obor, který je možno použít na všechny typy problémů, ale na každých z těchto problémů existoval lepší prostředek pro jeho vyřešení. Až v posledních několika letech došlo k masovému rozšíření neuronových sítí, který byl spojen hlavně s výrazným pokrokem v oblasti hardwaru, a tedy zvýšení výpočetní síly moderních počítačů, a díky výraznému zrychlení internetového připojení, díky kterému je možné sítě trénovat na obrovském množství dat, kterých je na internetu dostupné, pro obecné typy rozpoznávání, prakticky neomezené množství. Existuje velké množství typů neuronových sítí, například konvoluční, dopředné, rekurentní a mnoho dalších.

1.3.1 Historie neuronových sítí

První model neuronu byl publikován v roce 1943 Warrene McCullochem a Walterem Pittsem [12]. Výstupní hodnoty tohoto modelu byli převážně binární(-1, 0, 1).
V článku bylo dokázáno, že sítě vytvořené z těchto modelů neuronů dokáží počítat aritmetické a logické operace. Dalším pokrokem v této oblasti byl publikován v roce 1949 Donaldem Hebbem v knize "The Organization of Behavior"[8]. V ní byl popsán postup pro výpočet synapse neuronů. Ten byl odvozen z myšlenky existence podmíněných reflexů, které existují u všech myslících tvorů a byly popsány I. P. Pavlovem v roce 1903 v knize Experimentální psychologie a psychopatologie zvířat.
Tento postup sloužil k výpočtu vah mezi jednotlivým neurony pro možnost učení neuronových sítí na příkladech. V následných dvou desetiletí nedošlo k žádnému výraznému pokroku v oblasti neuronových sítí. V roce 1951 byl vytvořen první neuropočítač Snark, který ale nikdy nenašel uplatnění.

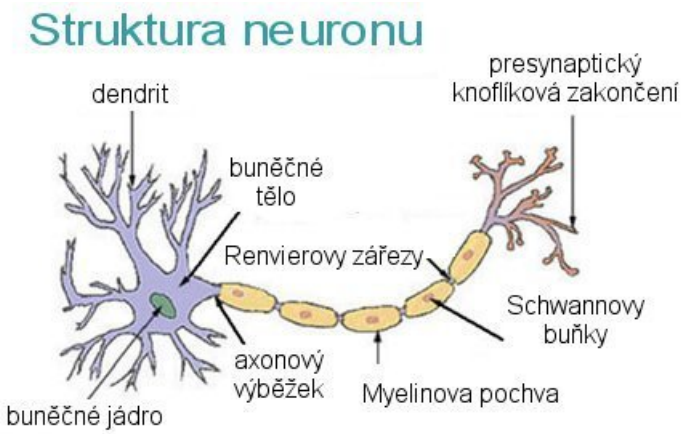
Dalším výrazným pokrokem v této oblasti bylo zobecnění modelu neuronu pro reálné číselné parametry, které publikoval v roce 1957 Frank Rosenblatt, ten byl pojmenován perceptron. Rosenblatt také pro perceptron navrhl učící algoritmus, který dokázal natrénovat stejné váhy sítě pro libovolné startovní váhy, a napsal knihu "Principles of neurodynamics"[13]. V následujících letech navrhl Rosenblatt spolu s Charlesem Wightmanem první neuropočítač s reálným využitím. Mark I Perceptron byl určen pro rozpoznávání znaků abecedy. Dalším pokrokem v oblasti neuronových sítí je pak nový typ modelu neuronu ADALINE. Učící algoritmus ADALINE se používá dodnes.

V následujícím období došlo k velkému odlivu zájmu z oblastí neuronových sítí, hlavně z důvodu malé využitelnosti při řešení reálných problémů. Oboru neuronových sítí také nepomohla negativní kampaň vedená Marvinem Minským, který se podílel na vývoji Snarku, a Seymourem Papertem. Ti v knize "Perceptrons"popsali problém perceptronu s počítáním funkce XOR. Tu je možno počítat vícevrstvou neuronovou sítí, ale v té době neexistoval algoritmus, který by byl schopen upravit váhy v takové síti. V knize byla též zpochybněna možnost vzniku takového učícího algoritmu. Další výzkum probíhá v minimální míře a téměř výhradně mimo území Spojených státu amerických, tedy v Japonsku a Evropě.

V roce 1986 byl publikován učící algoritmus, jehož vznik Minský a Papert zavrhli, skupinou vědců PDP. Tento "backpropagation"algoritmus je dnes používaný v 80% všech implementacích neuronových sítí. V roce 1987 se pak v San Diegu konala první konference zaměřená čistě na neuronové sítě.

1.3.2 Biologický neuron

Neuron je základním stavebním prvkem lidského mozku. Lidský mozek je tvořen bílou a šedou kůrou mozkovou. Šedá kúra je tvořena neurony a bílá kůra je tvořena nervovými vlákny. Podle [1] je neuron rozdělen do několika částí. První nejdůležitější součástí neuronu je buněčné jádro, které na základě vstupních signálů od předešlých neuronů a jiných částí těla. Na základě vstupních signálů pak generuje výstup dle aktivační funkce neuronu. Dendrit je pak nervové zakončení, které do neuronu přivádí tyto vstupní signály. Výstupní signál je z neuronu odváděn přes axon do synaptických zakončení, na kterých probíhá předání informace napojenému dendritu dalšího neuronu. Nervové vzruchy jsou v mozku a tedy i mezi neurony přenášeny jako elektrické impulzy. Na synaptickém zakončením dochází k vypuštění neurotransmiteru v odpovídajícím množství velikosti výstupního signálu. Neurotransmiter je vypuštěn do okolí a je pohlcen dendritem jiného neuronu. Dendrit dle množství zachyceného neurotransmiteru vygeneruje elektrický impulz, který slouží jako vstupní signál pro rozhodování neuronu. Množství přeneseného neurotransmiteru závisí na synaptické propustnosti mezi neurony. Jednotlivé synapse jsou poté rozděleny na excitační a inhibiční. Inhibiční synapse posílají do neuronu kladné signály a inhibiční záporné singály. Buněčné jádro neuronu obsahuje zděděné informace, tedy DNA. Mezi dendrity a synapsemi jednotlivých neuronů existuje pak určitá paměť. Po každém průchodu signálu dochází ke změně synaptické propustnosti mezi danými neurony. To odpovídá procesu učení.

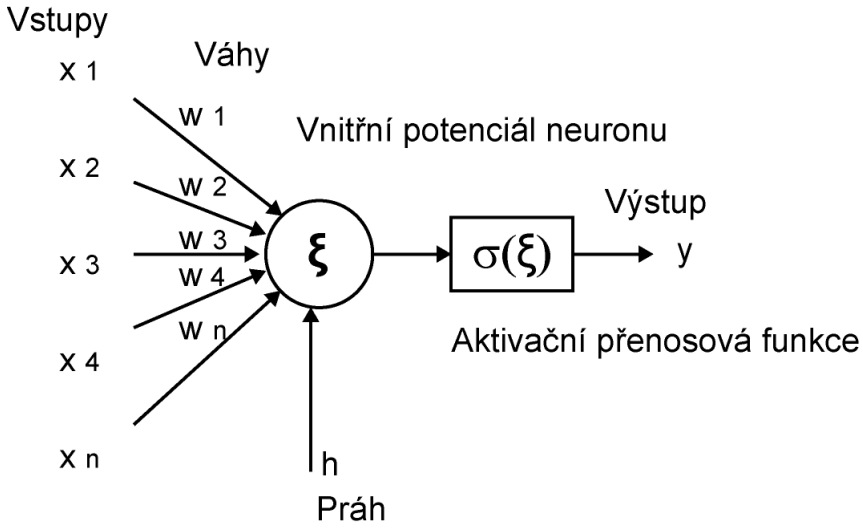


Obrázek 3: Struktura biologického neuronu

Zdroj: https://commons.wikimedia.org/wiki/File:Neuron_(cesky)-1.svg

1.3.3 Umělý neuron

Umělý model neuronu, perceptron, vychází ze zjednodušení modelu neuronu biologického. Byl popsán v roce 1957 Frank Rosenblatt. V tomto případě je tělo neuronu nahrazeno sumou vstupních signálů. Synapse a dendrity jsou nahrazeny přímým spojením neuronů. Synaptická propustnost je přenesena na váhu spojení jednotlivých neuronů. Výstupní signál neuronu je pak závislý na typu aktivační funkce neuronu, sumě vstupních signálů a jeho citlivosti. Klasické aktivační funkce umělých neuronů jsou skokové, sigmoidální či lineární. Model neuronu je zobrazen na obrázku 4.



Obrázek 4: Jednoduchý model neuronu
Zdroj: http://portal.matematickabiologie.cz

Výstup takového neuronu lze vypočítat dle vzorce:

$$o_j = \phi(\sum_1^n (x_n * w_n) - \theta)$$

Kde ϕ je aktivační funkce perceptronu, x_n je vstupní signál z n-tého perceptronu z předchozí vrstvy, w_n je váha spojení z n-tým předchozím perceptronem a θ je citlivost perceptronu. Síť s jedním neuronem, takzvaný perceptron, je schopna rozdělit stavový prostor pomocí přímky na dva podprostory. Je tedy schopna klasifikovat vstupní data do dvou tříd.

1.3.4 Učení neuronových sítí

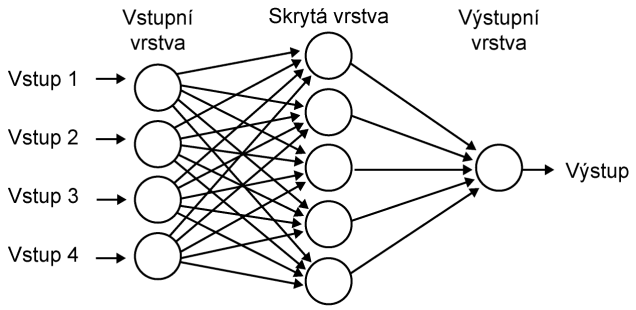
Před použitím neuronové sítě je potřeba provést proces učení, při které dochází ke změně parametrů neuronové sítě. Parametry, které se mění, jsou váhy jednotlivých vstupů neuronů a prahy neuronů.

1.3.4.1 Učení s učitelem Při učení s učitelem jsou poskytnuta neuronové síti dva odlišné typy dat. První jsou vstupy, které jsou přivedeny na vstup neuronové sítě. Druhou informací je pak požadovaný výstup neuronové sítě na k němu patřící vstupní data. Cílem je nastavit váhy tak, aby byl rozdíl mezi požadovaným výstupem a reálným výstupem minimální. Změna vah je prováděna bud sekvenčně, tedy po každém průchodu vstupních dat, nebo dávkově, tedy po průchodu určitého množství dat. Algoritmem který se používá nejčastěji pro trénování parametrů sítě je backpropagation, ten převádí zpětně chybu na výstupu, za pomoci parciálních derivací, na změnu parametrů sítě.

1.3.4.2 Učení bez učitele V tomto případě chybí požadovaný výstup. Změna vah a citlivosti perceptronu tedy nezávisí na výstupu, protože ten není s čím porovnávat. V takovém případě si síť vstupní data rozřazuje do několika tříd podle rozhodovacího kriteria, většinou dle eukleidovské vzdálenosti jednotlivých vstupních vektorů. Parametry perceptronů jsou pak upravovány tak, aby byl výstup sítě pro podobné vektory, s malou eukleidovskou vzdáleností, velmi podobný.

1.3.5 Dopředná neuronová síť

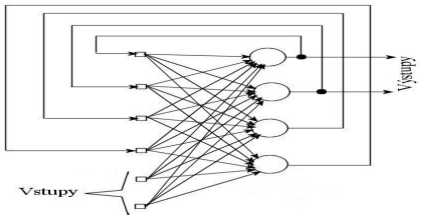
Jedním ze základních typů neuronových sítí jsou dopředné neuronové sítě. Dopředná neuronová síť je složena z několika vrstev neuronu, kdy výstup každého neuronu z předchozí vrstvy je vstupem každého neuronu z následující vrstvy. Síť má vstupní a výstupní vrstvu. Může také mít několik skrytých vrstev mezi nimi.



Obrázek 5: Dopředná neuronová síť
Zdroj: http://portal.matematickabiologie.cz

1.3.6 Rekurentní neuronové sítě

Dalším typem jsou rekurentní neuronové sítě. U rekurentních neuronových sítí dochází narozdíl od dopředných neuronových sítí k šíření signálů i proti směru neuronové sítě. Každému neuronu jsou, oproti klasické dopředné neuronové síti, přivedeny na vstup výstupy všech neuronů v daně vrstvě sítě, jak je vidět na obrázku 6.



Obrázek 6: Dopředná neuronová síť
Zdroj: http://cw.cz-slo-spolecnost.eu

U dopředné neuronové sítě požadujeme, aby po jejím natrénování byl výstup pro stejně vstupní data totožný. Rekurentní síť je pak použitelná pro případ, kdy potřebujeme vzít v potaz i časový kontext, tedy výsledek předchozích dat v minulém časovém okamžiku.

1.3.7 Konvoluční neuronové sítě

Konvoluční neuronové sítě vznikly především pro zpracování 2D dat. Jsou typem dopředné konvoluční sítě. Poprvé byli popsány v roce 1987 Kunihiko Fukushimou v článku Neocognitron [7]. Konvoluční neuronové sítě vycházejí stejně jako klasické neuronové sítě na podobnosti s lidským mozkem. Základem pro popsání konvolučních neuronových sítí byl výzkum Davida Huntera Hubela a Torsten Wiesela, ti v padesátých a šedesátých letech minulého století zkoumali aktivitu části mozku, zpracovávajícího vizuální vjemy, koček a opic. Při tomto výzkumu vizuálního kortexu promítali zvířatům na zeď černobílé rohy a snímali aktivitu neuronů. Závislost aktivity neuronů na promítaném obrazu pak popsaly v roce 1968[10]. Zjistili, že některé neurony jsou maximálně excitované při promítání přesného úhlu na určité místě a jiným stačí když je takovýto úhel promítán do určité oblasti. Stejně tak reagovaly neurony rozdílně na jiné hrany. Některé neurony tedy slouží pro detekci svislý hran jiné pro detekci vertikálních hrany, či rohů. Z této práce pak vznikl nápad vytvořit neuronovou síť, která by měla podobné vlastnosti.

1.3.7.1 Konvoluční vrstva Hlavní součástí konvoluční neuronové sítě je konvoluční vrstva, ve které jsou na obraz aplikovány konvoluční masky, takzvaného filtru, které slouží k extrakci příznaků. Vznikne tedy několik obrazů, nazývaných příznakové plochy, odpovídající počtu konvolučních masek. V dalších vrstvách se konvoluční masky aplikují na všechny obrazy vzniklé v předchozí vrstvě, dochází tedy k výraznému zvyšování počtu obrazů a tedy i parametrů. V případě konvoluční neuronové sítě se trénují právě konvoluční masky. V každé další vrstvě hledají konvoluční masky větší detaily. V první vrstvě se většinou natrénují detektory hran, rohů a dalších obecných tvarů.

1.3.7.2 Subsamplingová vrstva

Tato vrstva je vkládána mezi konvoluční vrstvy a je určena pro snížení počtu parametrů sítě. V současné době je nejpoužívajší maxpooling. Při jeho aplikaci je příznaková plocha rozdělena na stejné části a do výsledného obrazu je z každé části vložena jen nejvyšší hodnota. V případě že je v oblasti detekována nějaká hrany je nejspíše i v ostatních bodech této oblasti a nedochází tedy k výrazné ztrátě užitečné informace. Většinou je maxpooling použit na oblasti o velikosti 2*2. Obraz tedy poté má poloviční rozměry a dojde ke snížení potřebných parametrů o 75%.

1.3.7.3 Plně propojené vrstvy

Po poslední subsamplingové vrstvě jsou všechny příznakové mapy převedeny na jednoduchý vektor, za kterým následuje již klasická neuronová síť, kdy je každý neuron následující vrstvy spojen s každým neuronem vrstvy předchozí. Poslední, výstupní, vrstva neuronové sítě může mít jeden či více neuronů. V případě že jde o úkol klasifikace do 2 tříd bývá na výstupu jeden neuron. V případě klasifikace do více tříd bývá na výstupu počet neuronů roven počtu tříd a výstupem jednotlivých neuronů je pravděpodobnost klasifikace vstupního obrazu do třídy odpovídající neuronu.

2 Využívané technologie

2.1 Git

Git je volně dostupný open source verzovací program určení pro vývoj aplikace, na které pracuje najednou více lidí. Pro takový vývoj nejsou vhodné lokální kopie kódu, kdy je složité sdílet jednotlivé změny s ostatními vývojáři. U centralizovaných systému pak vzniká problém s prácí více vývojářů na stejné části kódu, kdy dochází ke konfliktům. Jednotlivý vývojáři pracují na lokální verzi vyvíjené aplikace a po dokončení této práce odešlou změny na server s aktuální verzí aplikace, zbylí vývojáři si poté svojí verzi aktualizují. Git a další verzovací aplikace, subversion(SVN), Mercurial a Concurrent Versions System (CVS), zajištují distribuci nejnovějších změn ve vývojové verzi aplikace a řeší konflikty způsobené současnou práci více vývojářů na stejné části kódu. Stejně tak ukládá jednotlivé verze programu, tak aby bylo možno se k nim kdykoliv vrátit, v případě neočekávaného problému v změněném kódu. V takovém systému má pak každý člověk s aktuální verzí aplikace v počítači zálohu veškeré práce a v případě že dojde ke ztrátě verze uložené na serveru, ať už z jakéhokoliv důvodu, může každý z těchto vývojářů obnovit vývojovou verzi bez jakýhkoliv ztrát, včetně všech předchozích verzí. Pro práci na této diplomové práci byl použit verzovací systém Git[2]. Git má původ v komerčním programu Bit-Keeper, který byl využíván při programování jádra operačního systému linux. Po jeho zpoplatnění, vyvinul tvůrce Linuxu Linus Torvalds sytém Git. Git uchovává při změnách snímky změněných souborů, narozdíl od jiných verzovacích programů, které uchovávají seznam změn v souboru. V případě změn od dvou různých vývojářů ve stejném souboru dochází ke konfliktu. Často je tento konflikt vyřešen automaticky pomocí sloučení změn(merge). K tomu odchází ve chvíli kdy jsou změny na různých místech souboru. V případě že jsou změny na stejných místech souboru je nutné manuální sloučení, kdy jeden z vývojářů vybere změny, které mají být zachovány.

2.2 Python

K vývoji aplikace pro tuto diplomovou práci byl použit programovací jazyk Python[6]. Python je vysokoúrovňový skriptovací programovací jazyk. Je open source a poskytuje instalační balíky pro majoritně zastoupené operační systémy(Windows, Mac OS, Linux). Ve valné většině Linuxových distribucí je dokonce Python předinstalován. Byl navržen v roce 1991 Guidem van Rossumem. V současné době jsou používány hlavně dvě verze Pythonu a to 2.7 a nejnovější Python 3.6. Python se stále vyskytuje ve dvou verzích z důvodu zpětné nekompatibility verze 3.x, při jejím vydání se část komunity rozhodla zůstat na verzi 2.x. Pro vývoj této aplikace je použit Python ve verzi 2.7. Hlavní výhodou Pythonu je obrovské množství dostupných knihoven pro téměř jakýkoliv účel. Pro vývoj webových aplikací je to například framework Django. Pro zpracování obrazu je určena knihovna openCV, která je sice napsána v programovacím jazyku C/C++ ale má i interface pro Python. Další významnou oblastí, ve které je Python využíván, jsou výzkum a numerické výpočty. Mezi nejznámnější knihovny z této oblasti patří NumPy, SciPy a Pandas. Numpy je knihovna pro matematické výpočty, v multidimenzionálních polích a maticích, ve vědeckých oblastech. SciPy je taktéž knihovna pro matematické výpočty ve vědeckých oblastech obsahující například funkce pro integrování, rychlou furierovu transformaci, interpolaci či optimalizaci. Pandas je knihovna určená pro zpracování a analýzu dat.

2.3 Lisa

Program vyvíjený v rámci této diplomové práce je určen pro integraci do programu Lisa(LIver Surgery Analyser) vyvíjeného na katedře kybernetiky Západočeské univerzity v Plzni pod vedením Ing. Miroslava Jiříka, za přispění studentů katedry. Lisa je vyvíjena ve spolupráce spolu s Lékařskou fakultou Univerzity Karlovy v Plzni. Hlavním účelem programu Lisa je z trojrozměrného snímku, získaného metodami medicínského zobrazování, hlavně snímky získané vyšetřením pomocí počítačové tomografie, získat segmentaci jater. Dalším z cílů je pro operaci resekce jater určit velikost resektátu, tedy velikost části jater, která musí být při této operaci odebrána z důvodu přerušení výživy této části jater, přerušením krevního řečiště, které tuto část vyživuje. Tato část je určena za pomoci modelu vnitřního uspořádání krevního řečiště v játrech. Při segmentaci jater je využíváno několika různých metod. Lisa obsahuje nástroj na prohlížení trojrozměrných snímku z medicínských zobrazovacích metoda, kdy jsou postupně zobrazovány jednotlivé řezy daty. Tato funkce je zároveň používaný při interaktivní segmentaci jater, kdy uživatel v některých řezech označí jedním tlačítkem myši části, které náleží játrům a druhým tlačítkem naopak označí části, které játrům neodpovídají. Následně přichází automatická segmentace jater, podle uživatelem označených pomocných dat. Po dokončení segmentace dostane uživatel opět možnost označit nesegmentované části jater a segmentované části, které játrům neodpovídají. A algoritmus segmentaci podle těchto změn upraví. V segmentovaných jsou pak pomocí dalších algoritmů segmentovány cévy, jejichž model se poté využívá pro výpočet velikosti resektátu, pří daném řezu během resekce jater.

2.4 Dostupné implementace konvolučních neuronových sítí pro Python

2.4.1 Caffe

Caffe[11] je knihovna vyvíjená Berkley Vision and Learning Center. Narozdíl od jiných knihoven je určena čistě na zpracování obrazových dat. Jednou z jejích největších výhod je rychlost zpracování dat, díky možnosti provádět výpočty na grafických kartách. Díky tomu je možné docílit rychlosti až 60 milionů obrazů za den. Samotná knihovna je napsána v programovacích jazycích C++ a C s API pro Python a Matlab. Samotný model neuronové sítě je čistě textový soubor s příponou .prototxt, není tedy nutné psát žádný kód. Caffe je používáno převážně ve vědeckých a akademických kruzích. Často je používáno pro ladění již existující konvolučních sítí, navrhnutých v jiných knihovnách. Je stále méně používané. Původní autor Caffe nyní za podpory Facebooku vyvíjí Caffe 2.

2.4.2 Theano

Theano[14] je jedna z knihoven pro neuronové sítě napsaná přímo v Pythonu. Díky tomu je možné využít pro běh knihovny NumPy. Stejně jako u Caffe lze použít pro výpočty grafických karet. Stejně jako Caffe je velmi používané v akademických kruzích. Samotná tvorba modelu není příliš intuitivní. Většinou se tedy používá spíše jako základní stavební kámen dalších knihoven, které přinášejí lepší API s jednodušší syntaxí. Theano v těchto knihovnách slouží jako část starající se o samotné počítání, modely jsou pak psány dle specifikací knihoven které jsou na něm postaveny. Theano je používáno jako backend pro Keras a Lasagne. Je distribuováno s licencí BSD, je tedy vhodné pro komerční použití.

2.4.3 Tensorflow

Tensorflow[13] je open source knihovna vyvinutá společností Google jako alternativa pro Theano. Narozdíl od Theana je Tensorflow naprogramován v programovacích jazycích C a C++ a má Python API. Na vývoji pracovalo několik autorů Theana. Nevýhodou Theana oproti Tensorflow je pomalejší běh. Výhodou je podpora distibuovaného výpočtu, díky které je trénování na více grafických kartách výrazně jednodušší. Narozdíl od dalších zmíněných knihoven obsahuje podporu i pro jiné metody umělé inteligence než neuronové sítě. Tensorflow také nemá komerční podporu. V současné době je tedy určen pouze pro výzkum umělé inteligence a není pravděpodobné že dojde ke změně.

2.4.4 Keras

Keras[3] je knihovna původně vytvořená zaměstnancem Googlu Françoisem Cholletem. Keras jako takový není přímo knihovnou poskytující plnou podporu neuronových sítí. Je to nadstavba, která používá jíž existující knihoven neuronových sítí a poskytuje uživateli možnost jednoduššího, rychlejšího a intuitivnějšího návrhu neuronové sítě. Jako základ pro keras je možné v současné době použít jednu z následujících knihoven: TensorFlow, Theano či CNTK. Keras podporuje jak konvoluční tak rekurentní neuronové sítě. Stejně tak podporuje výpočty jako na procesorech tak na grafických kartách. Jednou z hlavních výhod je rychlé prototypování neuronové sítě a následné provádění experimentů. Hlavní výhoda spočívá v modulárnosti a nezávisloti kódu na použité backend knihovně. Keras je použitelný pro komerční účely v závislosti na použité backend knihovně.

2.4.5 Torch

Torch je další z open source knihoven zabývajících se implementací neuronových sítí. Stejně jako Caffe 2 je i Torch podporován Facebookem. Hlavním rozdílem oproti ostatním knihovnám je jazyk použitý při vývoji. Torch je naprogramováný částečně ve skriptovacím jazyce Lua a částečně v jazyce C. Narozdíl od ostatních zmíněných knihoven nemá Torch vlastní API pro Python. Pro práci s Torch v Pythonu existuje samostatná API knihovna PyTorch. Stejně jako Theano je Torch šířen pod licencí BSD, díky které je možno ho využívat v komerčních projektech. Díky použití Lua jako základního stavebního kamene je Torch schopný držet rychlostně krok s Theanem.

2.4.6 Lasagne

Lasagne[5] je velmi podobný Kerasu. Narozdíl od Kerasu poskytuje podporu pouze pro Theano jako backend. Pro modelování neuronových sítí v Kerasu není potřeba umět používat syntaxi Theana. Oproti tomu Lasagne zachovává možnost využívat syntaxi Theana a více si samotný model přizpůsobit. V současné době dochází k naprosto vyjímečným aktualizacím a využití Lasagne opadá. Je šířeno pod MIT licencí a je tedy vhodné i pro komerční účely.

2.4.7 DeepLearning4j

DeepLearning4j je open source, distribuovaná knihovna určená pro Java Virtual Machine. Jedná se o nejvýznamnější implementaci neuronových sítí pomocí javy. Má API jak pro Javu tak pro Python. Právě Python API je založené na Kerasu a je tedy možné importovat modely navržené za pomoci kerasu do DeepLearning4j. DeepLearning4j je šířen pod Apache 2.0 licencí a je tedy vhodný pro komerční využití.

2.5 Použitá data

2.5.1 SLIVER07

Jedním z použitích datasetů pro trénování a validaci byl zvolen dataset SLIVER07. Ten byl vytvořen za účelem ohodnocení algoritmů pro segmentaci jater v soutěži, která byla součástí workshopu 3D Segmentation in the Clinic: A Grand Challenge pořádaného v roce 2007 sdružením The Medical Image Computing and Computer Assisted Intervention(MICCAI). Tato soutěž měla dvě oddělené části a to online a offline část. V online části byly pro každý soutěžní tým na webu workshopu dostupná trénovací a testovací data. Celkem byly k dispozici čtyři sety trénovacích dat po pěti snímcích, ke kterým byl dodán i soubor obsahující požadovanou segmentaci, a dva testovací sety taktéž po 5 snímcích. Vyhodnocení soutěže probíhalo na výsledcích jednotlivých soutěžních algoritmů, které dosáhli na testovacích snímcích, odeslaných pořadatelům. Během samotného workshopu pak probíhala také offline část soutěže na nových testovacích datech. Pro obě části soutěže byl použit stejný hodnotící systém. Výsledné skóre bylo ovlivněno objemovou přesahovou chybou, relativním rozdílem absolutních objemů segmentací, průměrnou symetrickou povrchovou vzdáleností, kvadratickou střední hodnotou symetrické vzdálenosti a maximální symetrickou vzdáleností. V našem případě byly využity všechny čtyři trénovací data sety včetně vzorových segmentací, které nám umožňují automatickou tvorbu dat pro trénování a validaci neuronové sítě, avšak pouze pro určení polohy jater, jelikož jediným orgánem segmentovaným v tomto data setu jsou játra. Pro stažení tréninkových dat je potřeba se registrovat do soutěže, která je stále otevřena pro nové účastníky.

2.5.2 IRCAD

Další použitým datasetem je IRCAD(3D Image Reconstruction for Comparison of Algorithm Database), který obsahuje anonymizovaná medicínská data, snímky z CT, ve formátu DICOM. Dataset obsahuje dvacet snímků CT, na kterých je zachyceno deset mužů a deset žen. Dataset je určen především jako testovací data pro algoritmy, které jsou trénovány k vyhledávání nádorů na játrech. Každý snímek je také doplněn o ruční segmentaci několika orgánů, případně další dílčích součástí organizmu a případných nádorů. Každý snímek datasetu obsahuje segmentaci jater, ostatní segmentace nejsou přítomné vždy. Z celkových dvacet snímků se nádory vyskytují v 75% procentech případů, tedy na patnácti snímcích. Oproti od SLIVER datasetu, kde jsou všechny snímky uloženy do 3D snímku formátu mhd, v tomto datasetu jsou snímky reprezentované jednotlivými horizontálními řezy původním snímkem, které jsou zabaleny do souborů formátu zip. To přináší mírně složitější zpracování oproti datasetu SLIVER. V tomto případě je nutné nejdříve všechny součásti datasetu rozbalit a následně z nich vybrat zipy snímků pacienta a snímků jednotlivých masek, rozdělených do složek, pojmenovaných podle segmentované časti organizmu. Stejně jako u datasetu SLIVER jsou primární oblastí zájmu tohoto datasetu játra a tedy v jednotlivých snímcích zabírají výraznou část játra. Tedy většina trénovacích dat jsou v tomto případě řezy jater.

2.6 Augmentace dat

Augmentace dat je operaci, při které jsou z existujícího datasetu generovány další data pro trénování neuronové sítě. Jejím hlavním důvodem je zabránění overfitingu při trénování neuronové sítě. Overfiting je případ, kdy je neuronová síť trénována na datech, která nejsou dostatečně rozmanitá a tedy obecná. To vede na situaci, kdy dochází při trénování k dobrému rozlišení trénovacích dat, ale pro nová data výsledná spolehlivost naprosto neodpovídá trénovací přesnosti. V takovém případě dochází k natrénování některých filtrů konvoluční vrstvy na hodnoty pixelů v jednotlivých místech odpovídající jednotlivým trénovacím datům, která nová nikdy neviděná data, jejichž klasifikace je naším cílem, neobsahují. Augmentace umožňuje získat z dostupných trénovacích dat maximální informaci. Existuje mnoho možností augmentování dat pro trénování různých druhů neuronových sítí. Pro obrazová data jsou použity různé translace, rotace či změny škálování již existujících obrazů v datasetu. Další metodou je zašumění jednotlivých obrazů. Jednou z možností augmentace jednotlivých obrazů je stranové, vertikální či horizontální, překlopení obrazů. Další možností je rotace obrazů, kolem jejich středů o zvolený úhel čí přiblížení a ořezání obrazu. Je možné také posunout obraz do stran, případně vybrat jen jeho náhodnou část a zbytek ořezat. U obrazů z kamer je pak někdy také používána augmentace jasu, pro simulaci různých denních dob zachycení obrazu. Zašumění obrazu pak simuluje různé chyby kamery, či snímání jinou kamerou, než kterou byla nasnímána trénovací data. Pro každou úlohu jsou vhodné jiné druhy augmentace, v závislosti na možnosti objevení takových dat v reálném provozu aplikace. Například pro trénování neuronové sítě pro řízení autonomního automobilu, nemá smysl vertikálně obracet dostupná data, protože taková data se v reálné úloze neobjeví.

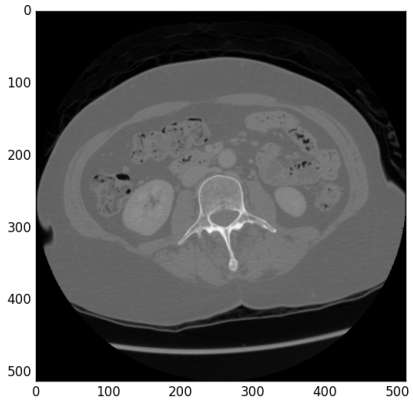
3 Návrh metody určení umístění řezu vzhledem k jaternímu parenchymu

Všechna vstupní data pro trénování a klasifikaci, z data setů SLIVER a IRCAD, jsou prezentována jako matice obsahující celá čísla. Tato čísla jsou již během vyšetření pomocí počítačové tomografie normována do rozmezí -1024 až 1024. Odpovídají tedy Hounsfieldově stupnici. Obecně pixely s velkou zápornou hodnotou odpovídají okolnímu vzduchu, případně plicím, pixely s hodnotou blízkou nule odpovídají měkké tkáni, tedy tuku a tkáni orgánů a pixely s velkou kladnou hodnotou odpovídají kostem.

Pro návrh úlohy bylo počítáno s několika omezeními. Ačkoliv jsou obecně všechny snímky z vyšetření pomocí počítačové tomografie normalizovány tak, aby jednotlivé řezy šli od spodní části lidského těla, v některých případech tomu tak není, pro tento úkol je to brána jako nutná podmínka funkce, vzhledem k použité augmentaci. Další požadavkem na vstupní data je poloha pacienta, kdy úloha počítá s pozici pacienta na zádech s omezením na točením o několik stupňů na oba boky.

Celá metoda je navržena obecně pro jakýkoliv obecně odlišitelný významný anatomický útvar. Proto bylo pro realizaci zvoleny konvoluční neuronové sítě, u kterých při změně požadovaného anatomického útvaru dojde k novému trénování.

Pro samotnou diplomovou práci je zvolenou implementací konvolučních neuronových sítí pro Python, vzhledem k jeho bezproblémové funkci na strojích metacentra, knihovna Keras. Jako backendová knihovna je vzhledem k lepší licenci zvoleno Theano.



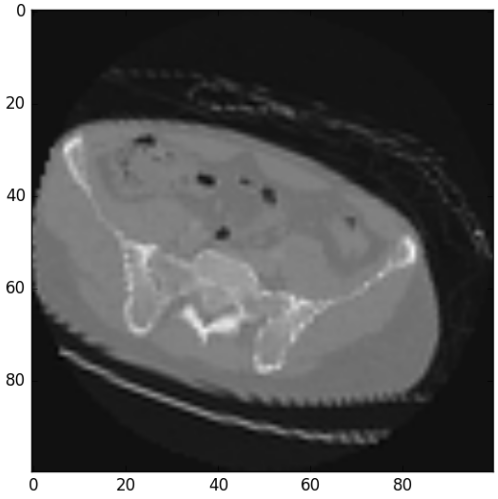
Obrázek 7: Řez z datasetu IRCAD

3.1 Použitá augmentace

Pro klasifikaci pozice snímku CT dle umístění vzhledem k danému orgánu, je potřeba vyhodnotit jednotlivé možné augmentační metoda a použít ty, které jsou pro tento problém vhodné. Jako omezení pro správnou funkci natrénované neuronové sítě je uvedeno zachycení snímku při poloze pacienta na zádech. Proto je zbytečné augmentovat data pomocí vertikálního prohození stran. Další zbytečnou metodou augmentace, pro tento úkol, je horizontální prohození stran, vzhledem ke vstupním datům. Formát DICOM totiž normalizuje směry jednotlivých os celého obrazu. Osa Z jde od nohou směrem k hlavě. Nikdy by tedy nemělo dojít k tomu, aby byla vstupní data obrácena.

Obraz výpočetní tomografie není přesným průmětem zachycených dat, při vyšetření. Během vyšetření totiž dochází ke konstantnímu posunu těla pacienta skrz prstenec zařízení. Radonova transformace počítá s použitím několik snímků, kdy se lehátko s pacientem neposouvá do prstence přístroje. Pro rekonstrukci celého snímku ze získaných dat je tedy nutné provést výpočet aproximace hodnoty jednotlivých obrazových bodů. Poté je výsledný snímek filtrován, tedy je z něj odstraněn šum. Přidání šumu do obrazu, jako forma jeho augmentace, je tedy opět naprosto zbytečné.

Lehátko je v každém zařízení pevné a nedá se sním hýbat ani v ose x ani v ose y. Toto vylučuje jako platnou formu augmentace i jakýkoliv posun řezu v jakékoliv ose. Stejně tak je vyloučené požít jakékoliv skosení obrazu, protože takové obrazy také nejsou obecně možnými výstupy výpočetní tomografie.

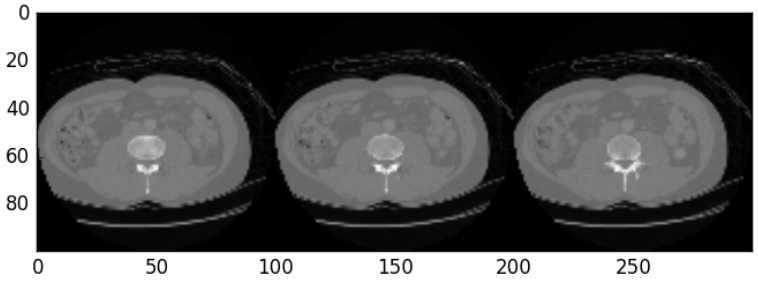


Obrázek 8: Řez s provedenou augmentací

Poslední možností augmentace tak zbývá natočení obrazu. To je jediná možná augmentace. V této diplomové práci je prováděna klasifikace řezu ve všech třech osách. Augmentace otáčením obrazů je vhodná jen pro řezy v transverzální rovině, tedy pro řezy, které jsou kolmé na páteř. Ostatní dvě roviny podle kterých jsou řezy brány jsou, frontální rovina, která dělí tělo na přední a zadaní část, a sagitální rovina, která dělí tělo na pravou a levou část. Pro řezy v těchto dvou rovinách není vhodná ani augmentace pomocí otáčení, protože vyšetření pomocí výpočetní tomografie takové řezy v těchto rovinách nikdy neposkytne.

3.2 Kontext řezu

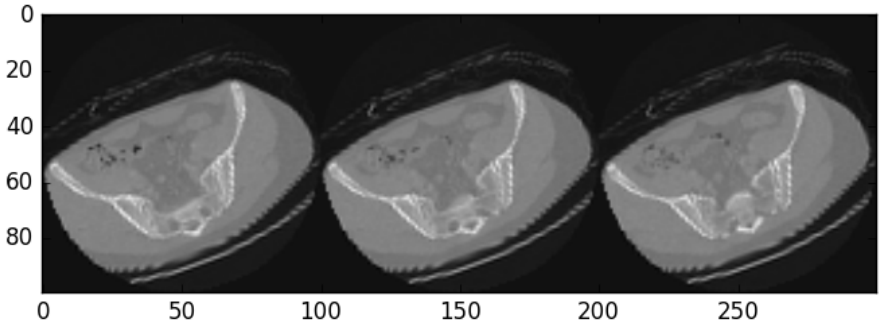
Obraz z CT je ve své podstatě trojrozměrným obrazem. Jedním z požadavků na funkci programu, vytvořenému k této diplomové práci, bylo odhadnout pozici dané řezu snímku získaného za pomoci CT, vzhledem k významným anatomickým strukturám. Vstupem samotné konvoluční neuronové sítě je tedy vždy minimálně jeden řez. Vstupem samozřejmě může být pro určení obalového kvádru určeného orgánu i celý trojrozměrný snímek, pořízení Výpočetní tomografií. Pro vylepšení přesnosti určení pozice právě u těchto trojrozměrných vstupních dat, byla použita znalost i okolních řezů určovaného řezu, tedy kontext toho řezu. Proto je možné natrénovat konvoluční neuronovou sít právě včetně přecházejícího a následujícího řezu. Omezení dané možností že na vstupu prediktivní části mohou být i jednotlivé řezy, znemožňuje použít trojrozměrnou neuronovou sít. Proto bylo nutné implementovat metodu využití kontextu, tak aby byla použitelná u dvou rozměrné neuronové sítě. Jedinou možností je tedy umístit všechny řezy do jednoho vstupního obrazu. Vstupní data pro trénování i predikci jsou pak vedle postupně poskládaný předešlý, určovaný a následující řez. Nezdálo se být vhodné ukládat takto vytvořená vstupní data jako samostatný dataset. Bylo tedy využito funkce knihovny Keras pro vytvoření generátoru vstupních dat v reálném čase během trénování či predikce. Generátor ze vstupních dat, která jsou seřazená a délky jednotlivých trojrozměrných snímků, generuje data skládáním snímků vedle sebe. V případě generování trénovacích dat je nejdříve na každý řez zvlášť aplikována augmentace a pak jsou teprve všechny řezy složeny do výsledného vstupního obrazu konvoluční neuronové sítě. Vstupem generátoru je také list s označením příslušnosti jednotlivých řezů k odpovídajícímu výstupu. Generátor pak generuje dvojice řezů s kontexty a odpovídajícím označením. V případě kdy je vstupním řeze první či poslední řez z jednoho trojrozměrného snímku je odpovídající chybějící snímek nahrazen kopií snímku který klasifikujeme.



Obrázek 9: Vstupní data pro využití kontextu

3.3 Vstupní data

Vstupní data, ať již ta určená pro trénování konvoluční neuronové sítě, čí ta která chceme klasifikovat, jsou jednotlivé řezy, ať již samostatné nebo v podobě celých trojrozměrných snímku z výpočetní tomografie, jsou uloženy v souboru datového typu *.tiff. Pro funkci neuronové sítě potřebujeme tyto řezy zpracované a pro účely trénování k nim odpovídající požadované označení příslušnosti do správné třídy. Primární účelem je hledání obalového kvádru obecně libovolného orgán, kde primárním orgánem zájmu jsou játra. Z tohoto důvodu bylo pro tuto klasifikaci určeno rozdělení do tří tříd. První třídou jsou řezy, které se nacházejí nad orgánem, u kterého chceme určit obalový kvádr. Další třídou jsou řezy, na kterých je přitom požadovaný orgán. A poslední třídou jsou řezy, které se nacházejí pod požadovaným orgánem. Výstup neuronové sítě může tedy klasifikovat řez do tři různých tříd. Výstupní vrstva konvoluční neuronové sítě bude tedy mít 3 neurony. Pro každou třídu je potřeba vytvořit odpovídající vektor požadovaných výstupů těchto tří neuronů, určený pro učení s učitelem. Každé třídě odpovídá vektor, který má dvě hodnoty nulové a na místě výstupního neuronu klasifikujícího odpovídající třídu pak jedničku. Samotná trénovací data je pak možné vkládat jako jednotlivé řezy či celé trojrozměrné snímky. Všechna řezy použité pro trénování jednoho modelu, či řezy pomocí něj klasifikované, musí splňovat požadavek na stejný rozměr šířky a výšky řezu, kvůli velikosti vstupní vrstvy navržené konvoluční neuronové sítě. V případě trojrozměrných dat na posledním rozměru, tedy hloubce, nezáleží. Ke každému datasetu je pak odpovídající soubor s příponou *.csv, který obsahuje veškeré zařazení jednotlivých řezů do určených tříd.



Obrázek 10: Vstupní data pro využití kontextu s augmentací

3.4 Příprava dat

Pro přípravu vstupních dat ze zmíněných datasetů je naprogramován skript *liver_data_preparation.py*. Dataset IRCAD je potřeba před samotným zpracováním ještě upravit, protože všechny trojrozměrné snímky jsou v něm uloženy v souborech po jednotlivých řezech. Proto je nejdříve potřeba jednotlivé řezy spojit do jednotlivých souborů reprezentujících jednotlivé trojrozměrné snímky z výpočetní tomografie. Při připravování dat pro konvoluční neuronovou síť z datasetu IRCAD je tedy nejprve volána funkce, která tyto řezy spojí. Stejně tak jsou na jednotlivé řezy rozděleny všechny trojrozměrné snímky obsahující segmentace jednotlivých orgánů. Tyto segmentace, používané pro určení třídy, jsou uloženy v složkách s názvem orgánu. Každý ze spojených trojrozměrných obrazů je pak samostatně načten a pro potřeby trénování konvoluční neuronové sítě a klasifikace je otočen dle požadované osy. Následně je pro každý obraz, podle požadovaného zadaného vstupního rozměru každého řezu, přepočítán na požadované rozlišení, kdy počet řezů není změněn. Během vykonávání této funkce na jednom obrazu je pak každému řezu přiřazen odpovídají popis třídy. Je tedy zároveň načten spojený obraz, obsahující segmentaci požadovaného orgánu, a postupně po jednotlivých řezech původního obrazu, dochází ke kontrole počtu různých hodnot v odpovídajícím řezu segmentace. Segmentace je složena z pixelů, které mají hodnotu odpovídající segmentaci či neodpovídající segmentaci. Celá segmentace je tedy dvouhodnotová. Při kontrole každého řezu je tedy kontrolován počet unikátních hodnot řezu. Tím je rozlišeno zda řez obsahuje požadovaný orgán, či ne. Zda je řez pod nebo nad orgán je určeno, vzhledem k návaznosti jednotlivých řezů, dle toho zda již byl v předchozích řezech obrazu orgán nalezen. Jednotlivé třídy jsou logicky v obrazu obsaženy v třech oblastech, kdy dochází pouze ke dvěma změnám. Samotné označení tříd je pak obsaženo v souboru s příponou *.csv, kdy pro každý obraz obsahuje soubor tři řádky. Každý řádek značí rozmezí řezů, které odpovídají třídě, která je slovně popsaná na řádku.

4 Experiment

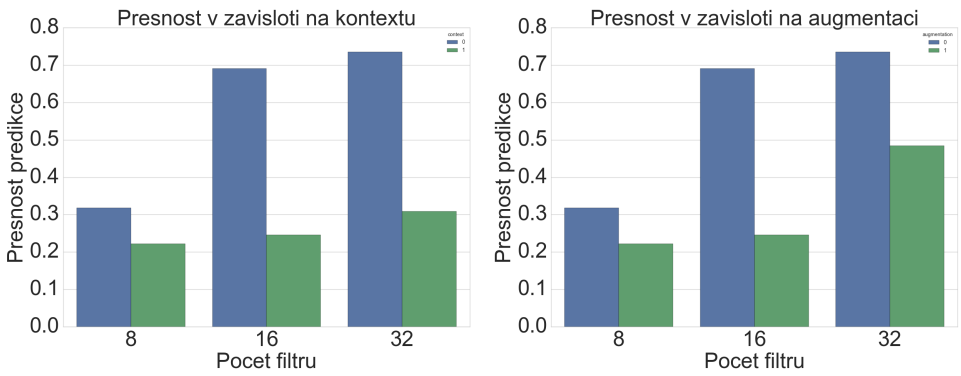
Pro účely porovnávání výsledků jednotlivých navržených modelů konvolučních neuronových sítí v závislosti na zvolených parametrech sítě byl navržen experiment. Při tomto experimentu jsou jako vstupní data zvolená pro trénování sítě použity snímky z datasetu IRCAD. Jako testovací data jsou pak použita data z datasetu SLIVER07.

Model konvoluční neuronové sítě je vytvořen dle parametrů uložených v souboru *config.txt*. V případě že tento soubor neexistuje a jako argument programu předán jiný, dochází k použití defaultního modelu napsaného přímo v programu.

Vstupní vrstvou modelu konvoluční neuronové sítě je první konvoluční vrstva. Jedna část konvoluční neuronové sítě obsahuje určený počet konvolučních vrstev následovanou vrstvou aktivačních neuronů. Poslední vrstvou v této části je pak vrstva maxpoolingu, která zmenšuje rozměr celé sítě a tedy i počet parametrů dalších vrstev. Po několika těchto částech následujtuto lohukonvoluční neuronové sítě e propojení pomocí vrstvy Flatten, která plní funkci převedení výsledku klasické konvoluční vrstvy do plně propojené vrstvy, což je vektor, který kdy je neuron v následující vrstvě spojen se všemy v předchozí. Následuje další plně propojená vrstvou. Aktivační vrstva, která navazuje na předchozí, je následována vrstvou dropout, která nastaví, během trénovaní konvoluční neuronové sítě, některé náhodné vstupy neuronů v následující vrstvě na nulu. Tato vrstva pomáhá s problémem overfittingu konvoluční neuronové sítě během jejího trénování. Následuje poslední vrstva s třemi výstupními neurony. Pro každý obraz je výstupem vektor s pravděpodobností příslušnosti k jednotlivým třídám.

4.1 Počet filtrů

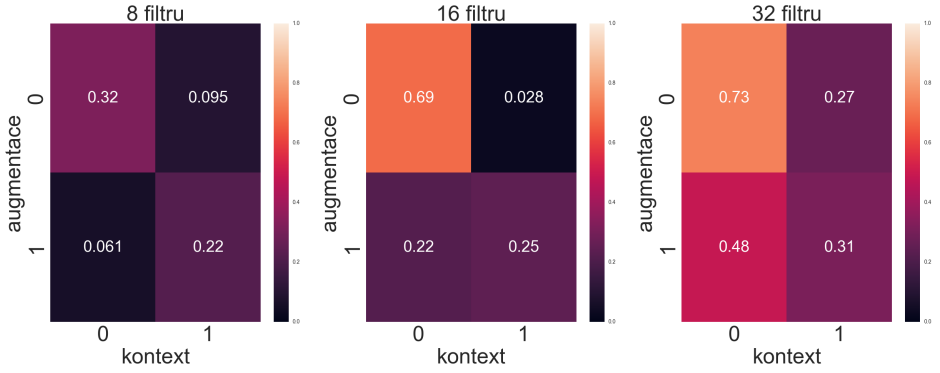
Počet filtrů v jednotlivých konvolučních vrstvách neuronové sítě je parametr, který určuje kolik filtrů je natrénováno pro každou konvoluční vrstvu. Každý z filtrů je natrénován pro vyhledávání jiných klíčových tvarů ve vstupním obraze. V případě že je pro vrstvu nastaveno málo filtrů nejsou ze vstupního obrazu vybrány všechny důležité informace. A v případě že je pro konvoluční vrstvu neuronové sítě trénováno příliš mnoho filtrů může dojít, při omezeném množství vstupních trénovacích dat, k natrénování modelu, tak že je závislý na hodnotě jednoho pixelu, který je náhodou ve většině trénovacích dat velmi podobný a výsledná klasifikace odpovídá požadované třídě. Ale obecně to tak v neznámých vstupních datech není. Testovanými parametry tohoto experimentu byly zvoleny počty filtrů 8, 16 a 32.



Obrázek 11: Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu filtrů

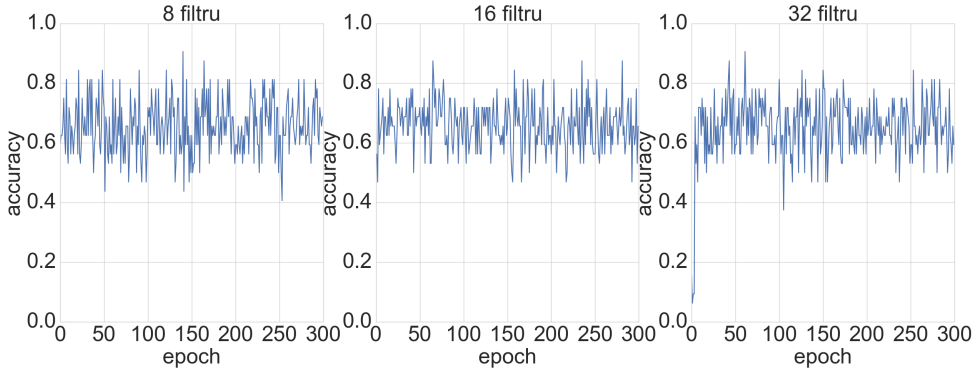
Na obrázku 11 je srovnání maxim výsledných přesností natrénované neuronové sítě, vztaženého k počtu filtrů v jednotlivých konvolučních vrstvách a použití augmentace nebo kontextu u konvoluční neuronové sítě. Modré sloupce jsou bez použití kontextu popřípadě augmentace. Zelené odpovídají případům použití kontextu či augmentace.

Na obrázku 12 je pak výsledná přesnost klasifikace na validačním datasetu SLIVER v závislosti na použití augmentace a kontextu. Narozdíl od předchozích grafů jsou zde uvedeny přesnosti v závislosti na použití jak augmentace tak kontextu. V jednotlivých experimentech se také výrazně projevuje náhodnost trénování neuronové sítě. V případě, kdy je pří trénování použita augmentace dochází k velkým rozdílům v uspěšnosti klasifikace při opakovaných trénováních. Nejlepších výsledků při klasifikaci je dosaženo při použití 32 filtrů.



Obrázek 12: Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu filtrů. 0 metoda není využita, 1 metoda je využita

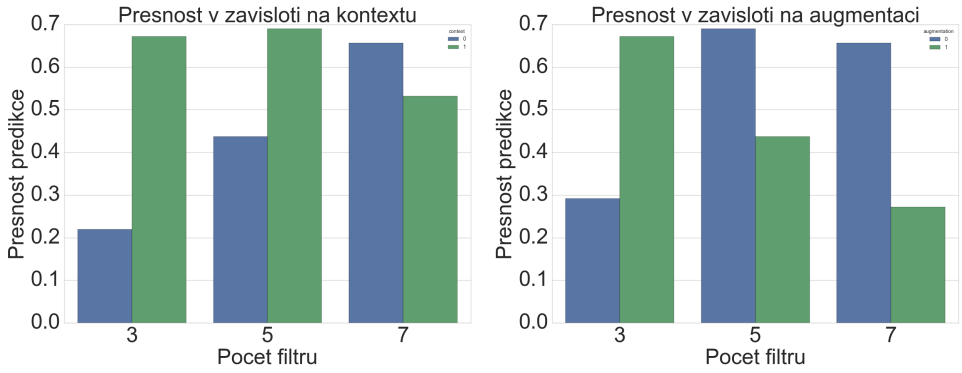
Na obrázku 13 jsou uvedeny přesnosti během trénování konvoluční neuronové sítě v průběhu jednotlivých batchů, pro trénování s augmentací bez kontextu pro zadané množství filtrů. Pro trénování ať již s kontextem nebo augmentací jsou jednotlivé epochy brány jako trénování po jednotlivých batchích, tedy 32 vstupních obrazech. V takovém případě nedochází v grafu učení k hladkému průběhu.



Obrázek 13: Přesnost trénování vzhledem k počtu filtrů, po jednotlivých částech dat.

4.2 Velikost filtrů

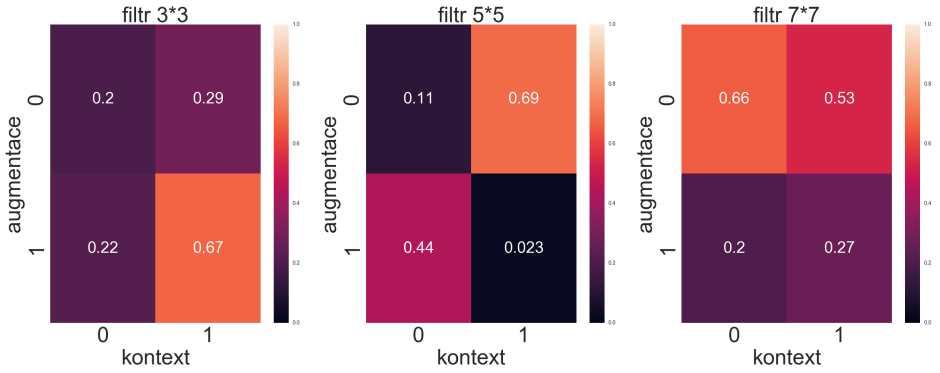
Kromě počtu filtrů v jednotlivých konvolučních vrstvách neuronové sítě je dalším důležitým parametrem velikost těchto jednotlivých filtrů. Čím vetší filtr tím větší specifické tvary může neuronová vrstva detekovat. Bohužel platí také přímá úměrnost mezi velikostí filtrů a počtem parametrů k natrénování. Počet parametrů potřebných k natrénování pak souvisí i s časovou náročností trénování a klasifikace navržené konvoluční neuronové sítě. Jako testované parametry byly zvoleny velikosti filtru 3, 5 a 7.



Obrázek 14: Přesnost klasifikace vzhledem k augmentaci, kontextu a velikosti filtrů. 0 metoda není využita, 1 metoda je využita

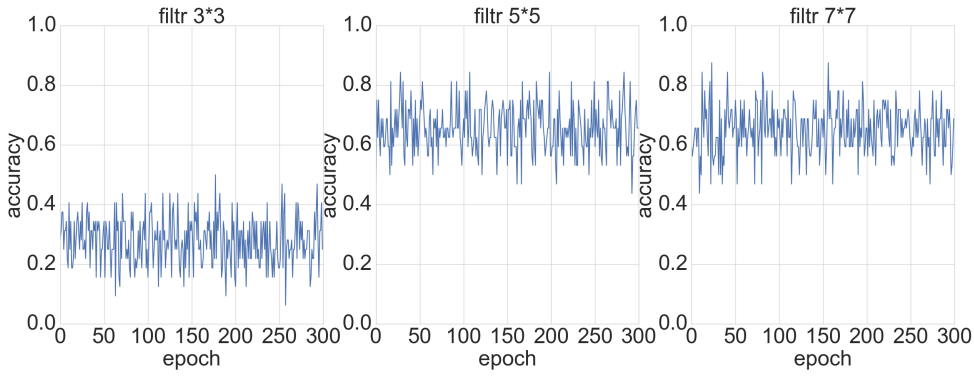
Na obrázku 14 je srovnání maxim výsledných přesností natrénované neuronové sítě, vztaženého k velikosti filtrů v konvolučních vrstvách a použití augmentace nebo kontextu u konvoluční neuronové sítě.

Na obrázku 15 je pak výsledná přesnost klasifikace na validačním datasetu SLIVER v závislosti na použití augmentace a kontextu. Narozdíl od předchozích grafů jsou zde uvedeny přesnosti v závislosti na použití jak augmentace tak kontextu. Bez ohledu na použití augmentace či kontextu se nejlépe jeví použití filtru s velikostí strany 7.



Obrázek 15: Přesnost klasifikace vzhledem k augmentaci, kontextu a velikosti filtrů. 0 metoda není využita, 1 metoda je využita

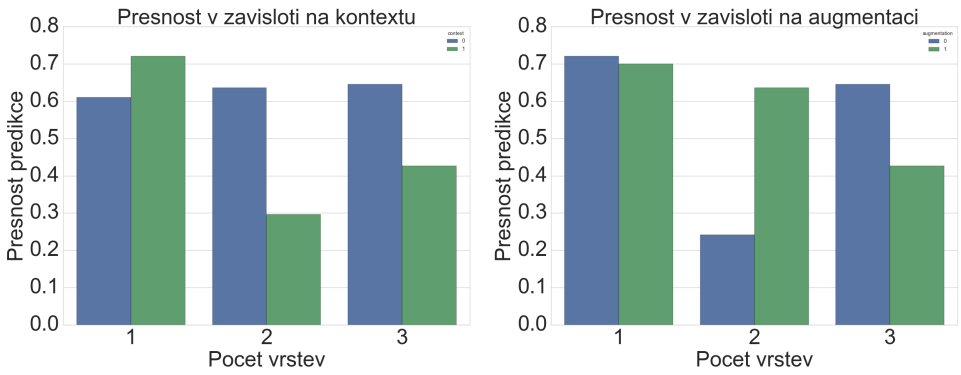
Na obrázku 16 jsou uvedeny přesnosti během trénování konvoluční neuronové sítě v průběhu jednotlivých batchů, pro trénování s augmentací bez kontextu pro zadané množství filtrů.



Obrázek 16: Přesnost trénování vzhledem k velikosti filtrů, po jednotlivých částech dat

4.3 Počet konvolučních vrstev

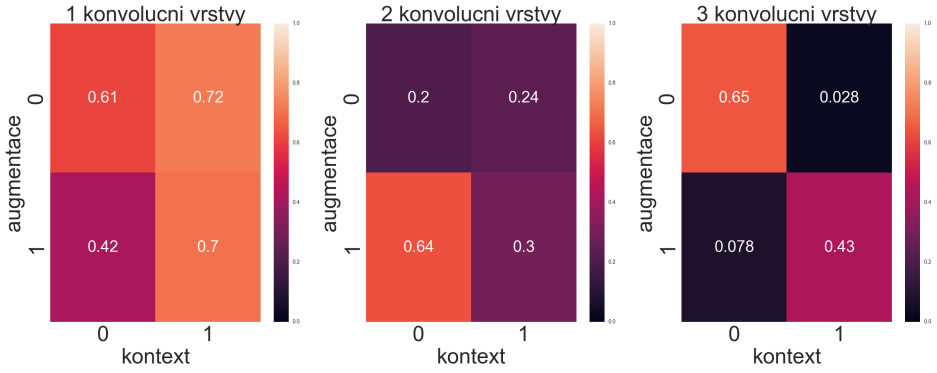
Tento meta parametr udává počet konvolučních vrstev, které jdou bezprostředně za sebou, mezi předchozí a následující aktivační a maxpoolingovou vrstvou. Ve většině případů jsou používány maximálně dvě konvoluční vrstvy bezprostředně za sebou. V našem případě je pro zachování stejného postupu jako u ostatních parametrů použity až tři konvoluční vrstvy. Stejně jako u dalších meta parametrů roste s počtem konvolučních vrstev, počet parametrů sítě, které je potřeba natrénovat. Tedy roste i výpočetní složitost celé operace trénování. Stejně tak roste i složitost klasifikace, i když téměř zanedbatelně.



Obrázek 17: Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu vrstev. 0 metoda není využita, 1 metoda je využita

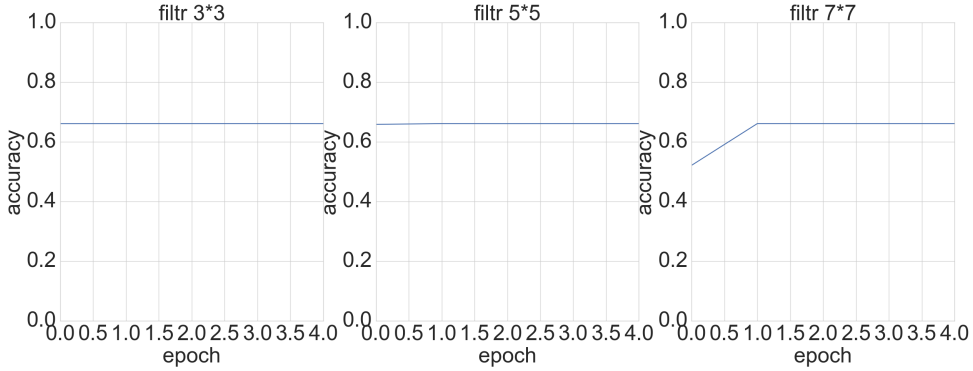
Na obrázku 17 je srovnání maxim výsledných přesností natrénované neuronové sítě, vztaženého k velikosti filtrů v konvolučních vrstvách a použití augmentace nebo kontextu u konvoluční neuronové sítě. Z těchto grafů je patrné že pro klasifikaci je nejvhodnější použít pouze jednu konvoluční vrstvu mezi aktivační a maxpoolingovou vrstvou.

Na obrázku 18 je pak výsledná přesnost klasifikace na validačním datasetu SLIVER v závislosti na použití augmentace a kontextu. Narozdíl od předchozích grafů jsou zde uvedeny přesnosti v závislosti na použití jak augmentace tak kontextu. V případě tohoto parametru sítě se jako nejlepší možnost, vzhledem k přesnosti klasifikace jeví použití jedné konvoluční vrstvy.



Obrázek 18: Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu vrstev. 0 metoda není využita, 1 metoda je využita

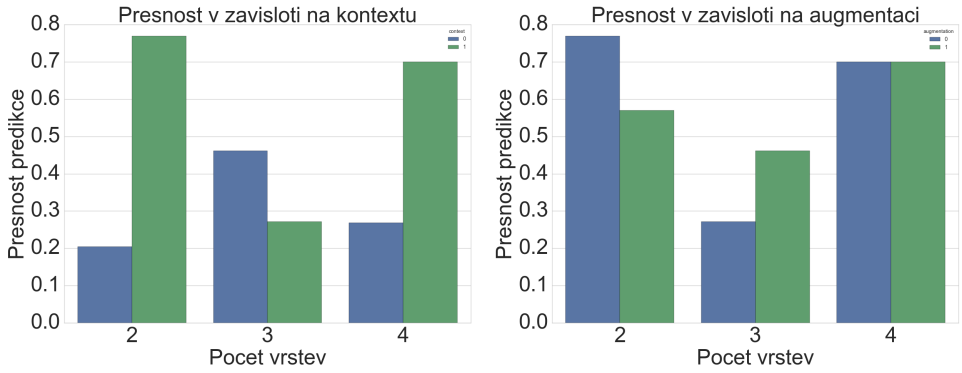
Na obrázku 19 jsou uvedeny přesnosti během trénování konvoluční neuronové sítě v průběhu jednotlivých průchodech celého data setu. Narozdíl od grafů předchozích meta parametrů sítě, je v tomto případě použit graf pro trénování bez použití kontextu či augmentace. Je vidět rozdíl v počtu epoch, kdy pří každé epoše je trénování provedena na celém trénovacím data setu.



Obrázek 19: Přesnost trénování vzhledem k počtu vrstev, po jednotlivých částech dat

4.4 Počet kompletních vrstev

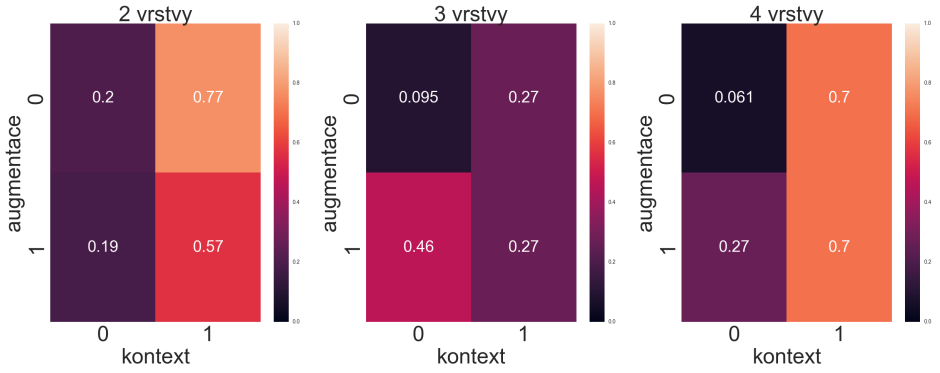
Každá kompletní vrstva navržené konvoluční neuronové sítě je tvořena několika konvolučními vrstvami, následovanými aktivační vrstvou a poslední vrstvou maxpoolingu. Po poslední kompletní vrstvě následuje převod na plně propojené vrstvy pomocí vrstvy flatten. Čím více kompletních vrstev model konvoluční neuronové sítě obsahuje, tím komplexnější třídy dokáže neuronová síť klasifikovat. Zvolenými parametry pro testování jsou 2, 3 a 4 kompletní vrstvy.



Obrázek 20: Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu kompletních vrstev. 0 metoda není využita, 1 metoda je využita

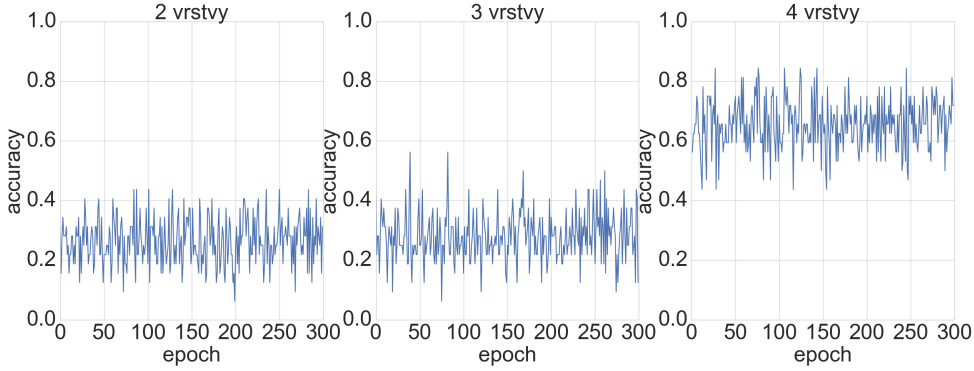
Na obrázku 20 je srovnání maxim výsledných přesností klasifikace natrénované neuronové sítě, vztaženého k počtu kompletních vrstev a použití augmentace nebo kontextu u konvoluční neuronové sítě.

Na obrázku 21 je pak výsledná přesnost klasifikace na validačním datasetu SLIVER v závislosti na použití augmentace a kontextu. V tomto případě jsou si, co do obecnosti správné klasifikace, výsledky velmi podobné pro 2 a 4 vrstvy. Jako nejlepší parametr jsou zvoleny 2 vrstvy, které mají celkově téměř stejnou přesnost a nezatěžují trénování tolik co se týká náročnosti.



Obrázek 21: Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu kompletních vrstev. 0 metoda není využita, 1 metoda je využita

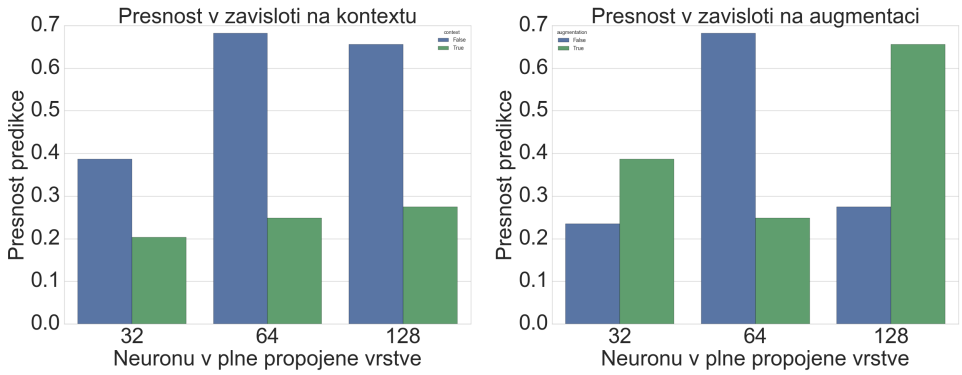
Na obrázku 22 jsou uvedeny přesnosti během trénování konvoluční neuronové sítě v průběhu jednotlivých batchů, pro trénování s augmentací a kontextem pro zadané množství filtrů.



Obrázek 22: Přesnost trénování vzhledem k augmentaci, kontextu a počtu kompletních vrstev, po jednotlivých částech dat

4.5 Počet neuronů v plně propojené vrstvě

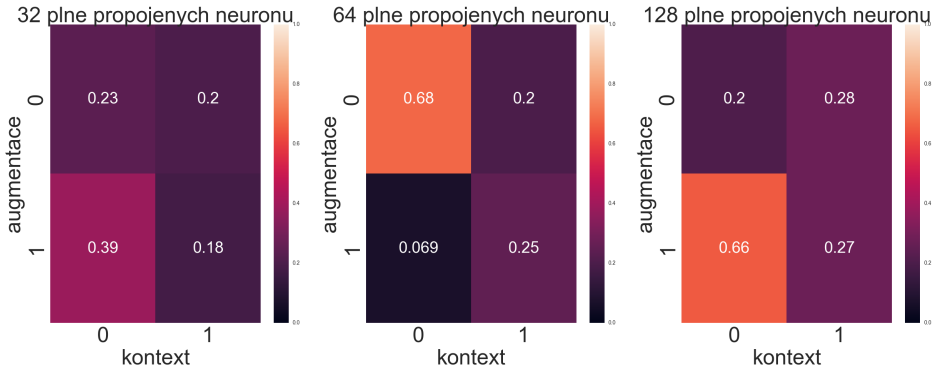
Neurony v plně propojené vrstvě mají na svém vstupu výstupy všech neuronů v předchozí vrstvě. Jednotlivé konvoluční vrstvy vždy obsáhnou pouze část předchozího obrazu. Plně propojené vrstvy jsou pak schopné spojit příznaky, které jsou například na opačných koncích obrazu. Počet neuronů v plně propojené vrstvě pro experiment byl zvolen 32, 64 a 128 neuronů.



Obrázek 23: Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu a počtu neuronů v propojené vrstvě. 0 metoda není využita, 1 metoda je využita

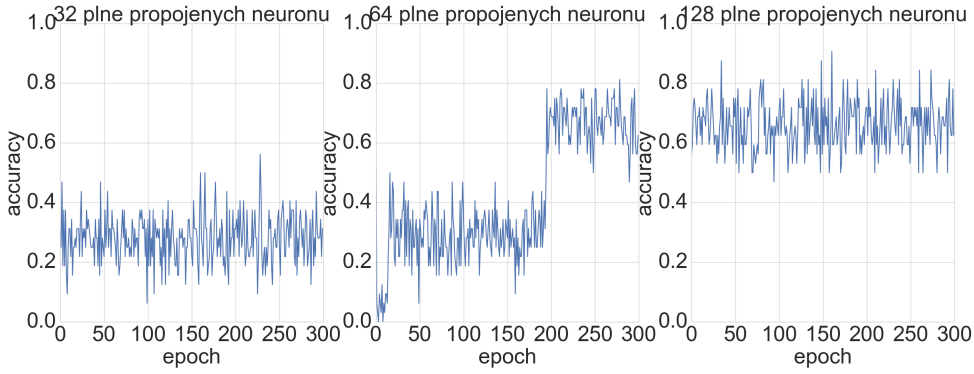
Na obrázku 23 je srovnání maxim výsledných přesností klasifikace natrénované neuronové sítě, vztaženého k počtu kompletních vrstev a použití augmentace nebo kontextu u konvoluční neuronové sítě.

Na obrázku 24 je pak výsledná přesnost klasifikace na validačním datasetu SLIVER v závislosti na použití augmentace a kontextu. Pro tento meta parametr sítě je zjevně nejlepší hodnota 128 neuronů v plně propojené konvoluční vrstvě.



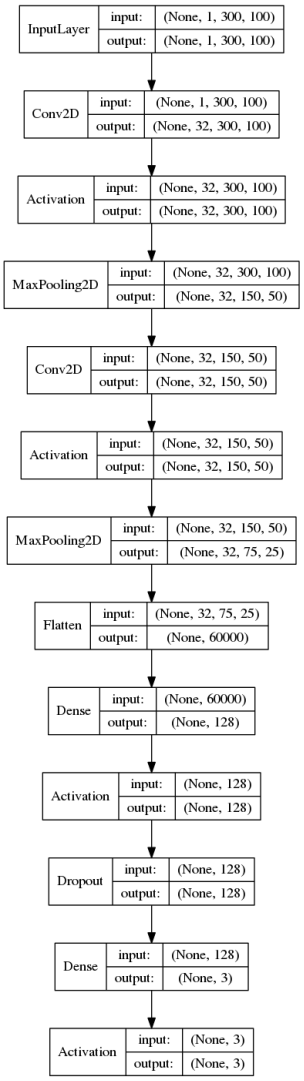
Obrázek 24: Přesnost klasifikace vzhledem k augmentaci, kontextu a počtu a počtu neuronů v propojené vrstvě. 0 metoda není využita, 1 metoda je využita

Na obrázku 25 jsou uvedeny přesnosti během trénování konvoluční neuronové sítě v průběhu jednotlivých batchů, pro trénování s augmentací bez kontextu pro zadané množství filtrů.



Obrázek 25: Přesnost trénování vzhledem k augmentaci, kontextu a počtu neuronů v propojené vrstvě, po jednotlivých částech dat

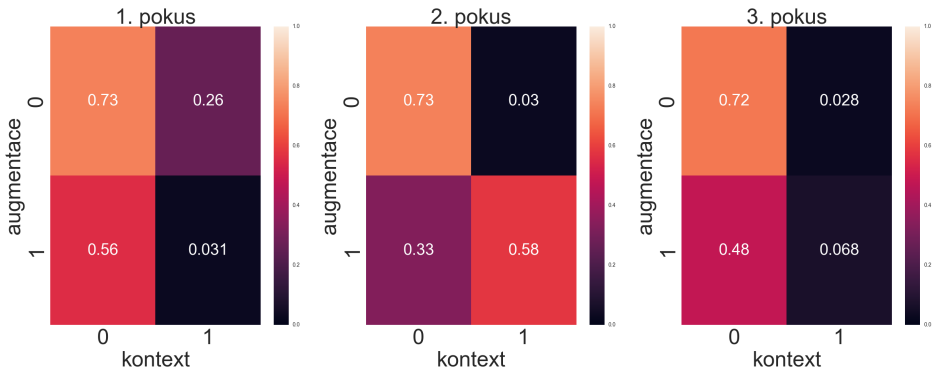
4.6 Výsledná neuronová síť



Obrázek 26: model neuronové sítě

Pro výsledný model konvoluční neuronové sítě, zvolený pro klasifikaci řezů kolmých na páteř, byli zvoleny meta parametry získané z výše uvedených experimentů. Tedy konvoluční neuronovou síť tvoří dvě vrstvy, každá složení z jedné konvoluční vrstvy následované vrstvou aktivační a poslední součástí je maxpoolingová vrstva. Po těchto dvou vrstvách následuje převod na plně propojené vrstvy. Další neuronová vrstva sítě má 128 neuronů. Výsledná klasifikace je pak v poslední vrstvě určena ze tří výstupním neuronu, kdy výstup každého z nich určuje pravděpodobnost zařazení vstupního řezu do odpovídající třídy. Tento model neuronové sítě je vidět na obrázku 26, včetně dimenzí jednotlivých vrstev. Tento model byl vygenerován za pomoci kerasu pro rozměry vstupních dat 100 na 100 pixelů a pro zapnuté využití kontextu řezu, proto má vstupní vrstva rozměr 300 na 100 pixelů, to odpovídá třem řezů složeným vedle sebe. Pro tuto neuronovou síť je pak proveden stejný experiment jako byl proveden při návrhu jednotlivých parametrů této sítě. Nejprve byla tato síť natrénována za pomoci dat z data setu IRCAD a následně došlo ke kontrolní klasifikaci dat z data setu SLIVER. Tento proces je, stejně tak jako v předešlých případech, proveden pro všechny možné kombinace použití kontextu a augmentace. Pro ověření stability klasifikace jednotlivých model pro jednotlivé kombinace kontextu a augmentace je celý proces pro všechny kombinace proveden třikrát.

Výsledky jednotlivých nezávislých pokusů jsou pak zobrazeny na obrázku 27.



Obrázek 27: Přesnost trénování vzhledem k augmentaci a kontextu pro konečný model neuronové sítě. 0 metoda není využita, 1 metoda je využita

Z těchto výsledků jednotlivých realizací pokusu je zjevné, že využití kontextu řezu není pro tuto úlohu naprosto relevantní, kdy pouze při jedné realizace, která využívá kontextu, je úspěšnost klasifikace větší než poloviční v ostatních případech je z jednotlivých výsledků zjevné, že využití kontextu není pro tuto úlohu klasifikace vhodné. Experimenty, při kterých nebyl použit kontext, ukázaly výrazně větší přesnost a stabilitu klasifikace řezu. Při využití augmentace se přesnost klasifikace pohybuje jen okolo padesáti procent. To je ale nejspíše dáno charakterem řezů, obsažených v testovacím data setu SLIVER. V tomto datasetu jsou všechny snímky pozicované stejně jako v trénovacím data setu IRCAD, tedy všichni pacienty leží přímo na zádech a případné úpravy jednotlivých filtrů natrénovaných pomocí augmentovaných, tedy o několik stupňů pootočených dat, se neprojeví. Nejlepší přesnosti a zároveň stability klasifikace řezu bylo dosaženo při trénování konvoluční neuronové sítě bez použití kontextu a augmentace. Takto modelovaná konvoluční neuronová síť si drží přesnost klasifikace při každém individuálním tréninku na stejných datech. Z jednotlivých

5 Závěr

Cílem této diplomové práce bylo navrhnout a následně implementovat program, který ze snímku získaných při vyšetření provedeném pomocí počítačové tomografie určí polohu daného řezu v těle pacienta, vzhledem k obecné, vybrané anatomické struktuře. V tomto případě jsou jako tato anatomicky významná struktura zvolena játra, vzhledem k budoucí začlenění celého kódu do programu Lisa, jehož funkce je ve většině případů spojená s hledáním a analýzou jater.

Při úlohách, kde není dostupný odpovídající počet trénovacích dat se používá metoda augmentace dat. Ta využívá existující data k vytvoření dat nových odlišných dat, jejich základní úpravou, například v tomto případě jejich otočením o několik stupňů kolem středu řezu. Zároveň pomáhá augmentace zamezit přetrénování. Pro zlepšení výsledně klasifikace bylo navrženo využití kontextu řezu, tedy okolních řezů řezu určeného pro trénink či klasifikaci. Vstupní obraz konvoluční neuronové sítě při tréninku nebo klasifikaci je pak složen z předchozího, trénovaného či klasifikovaného a následujícího řezu. Pro generování těchto upravených dat jsou vytvořeny zvláštní metody.

Byl navržen experiment pro volbu struktury sítě pro klasifikaci řezů vzhledem k jaternímu parenchymu. Během tohoto experimentu bylo použito 4159 řezu z data setu SLIVER pro klasifikaci a 5646 řezů z data setu IRCAD pro trénink sítě v hlavní ose kolmé na páteř.

Pro zvolení meta parametrů neuronové sítě bylo provedeno několik experimentů, při kterých byla porovnávána přesnost klasifikace natrénovaných neuronových sítí pro různé hodnoty jednotlivých parametrů, pro všechny možné kombinace použití augmentace a kontextu řezu. Následně byl pro takto navrhnutou struktury konvoluční neuronové sítě proveden stejný experiment, který byl pro odstranění vlivu náhodnosti, hlavně v případě použití augmentace, třikrát. Konvoluční neuronová síť, která nevyužívá ani kontextu ani augmentace, vykazuje nejlepší výsledky a to stabilní úspěšnost klasifikace 73%.

Hlavním problémem je hlavně malé množství trénovacích dat pro třídu řezů nad játry, kdy na některých snímcích z počítačové tomografie tyto řezy naprosto chybí a nedochází tedy k přesnému natrénování pro tuto třídu. Celá práce je navržena velmi obecně a je tedy možné ji použít pro segmentaci prakticky jakýkoliv obecně odlišitelný významných anatomických celků.

Reference

[1] I. A. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43:3–31, 2000.

[2] Scott Chacon. *Pro Git*. Apress, Berkely, CA, USA, 1st edition, 2009.

[3] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[4] Stanley R Deans. *The Radon transform and some of its applications*. Courier Corporation, 2007.

[5] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, et al. Lasagne: First release., August 2015.

[6] Allen Downey, Chris Meyer, and Jeffrey Elkner. *How to think like a computer scientist : learning with Python*. Green Tea Press, 2002.

[7] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[8] Donald O. Hebb. *The organization of behavior: A neuropsychological theory*. Wiley, New York, June 1949.

[9] J. Hsieh. *Computed Tomography: Principles, Design, Artifacts, and Recent Advances*. SPIE Press Monograph Series. SPIE, 2015.

[10] David H. Hubel and Torsten N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology (London)*, 195:215–243, 1968.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[12] Walter Pitts. The linear theory of neuron networks: the static problem. *Bull. Math. Biophys.*, 4:169–175, 1942.

[13] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.

[14] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.