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 LEARNING FOR MEDICAL IMAGE ANALYSIS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **KATEŘINA TRÁVNÍČKOVÁ**
AUTHOR

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

BRNO 2017

Zadání bakalářské práce/19460/2016/xtravn24

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Trávníčková Kateřina**
Obor: Informační technologie
Téma: **Hluboké neuronové sítě pro analýzu medicínských obrazových dat**
Deep Learning for Medical Image Analysis
Kategorie: Zpracování obrazu

Pokyny:

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

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Španěl Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá analýzou medicínských objemových dat pomocí konvolučních neuronových sítí. Vstupem analýzy jsou CT snímky lidských končetin, výstupem pak vysegmentované kontury dlouhých kostí, humeru a tibie. Cílem práce je nalezení vhodného nastavení konvoluční neuronové sítě pro co nejpřesnější výstup analýzy. Jako metrika úspěšnosti byla zvolena plocha pod Precision-Recall křivkou (AUC). Nejlepší dosažená úspěšnost se pohybuje kolem 88 % (0.8778 AUC). Pro implementaci řešení byl použit framework Caffe, resp. caffe modul pro skriptovací jazyk python.

Abstract

This bachelor thesis deals with medical volume data analysis using convolutional neural networks. The input of the analysis is a CT scan of human limbs and the output are segmented countours of long bones, humerus and tibia. The goal of this work is to find suitable convolutional neural network settings to achieve the best possible analysis output while the area under the Precision-Recall curve is used as the precision metric. The best accuracy reaches almost 88 % (0.8778 AUC). The implementation is based on Caffe framework, or python caffe module.

Klíčová slova

Konvoluce, konvoluční neuronové sítě, strojové učení, segmentace kontur objektu, medicínská objemová data, Caffe framework, pycaffe

Keywords

Convolution, convolutional neural networks, machine learning, object contour segmentation, medical volume data, Caffe framework, pycaffe

Citace

TRÁVNÍČKOVÁ, Kateřina. *Hluboké neuronové sítě pro analýzu medicínských obrazových dat*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Španěl Michal.

Hluboké neuronové sítě pro analýzu medicínských obrazových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Michala Šapněla, Ph.D.
Další informace mi poskytl pan Ing. Michal Hradiš, Ph.D.
Dataset pro trénování a vyhodnocení analýzy mi poskytla společnost 3Dim Laboratory, s.r.o. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Kateřina Trávníčková
15. května 2017

Poděkování

Chtěla bych poděkovat panu Ing. Michalovi Španělovi, Ph.D. za užitečné připomínky, ochotu a trpělivost během vedení mé bakalářské práce. Dále bych chtěla poděkovat panu Ing. Michalu Hradišovi, Ph.D. za vedení semináře zaměřeného na neuronové sítě a společnosti 3Dim Laboratory, s.r.o. za poskytnutí trénovacího datasetu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V současné době dochází díky pokročilým výpočetním technologiím k rychlému rozvoji oblasti umělé inteligence. Ta nalézá své uplatnění mimo jiné i v medicíně, kde je však limitována několika faktory. Pomineme-li vysoké požadavky na spolehlivost a s nimi spojený zdlouhavý proces nasazování nových postupů do medicínskké praxe, je stále třeba přihlédnout k problému nedostatku dat. Metody hlubokého učení zpravidla vyžadují rozsáhlé datasety, které jsou díky vyžadované profesionalitě anotací a díky omezené možnosti využití osobních zdravotních záznamů pacientů obtížně dostupné.

Tato práce se zabývá analýzou skenů výpočetní tomografie (CT skenů) pomocí konvolučních neuronových sítí a dotýká se tedy oblastí medicíny, umělé inteligence a zpracování obrazových dat. Přesněji se zaměřuje na detekci kontur, jejíž uplatnění blízce souvisí se segmentací a detekcí objektů. Hledanými konturami jsou v tomto případě kontury dlouhých kostí lidských končetin, humeru a tibie. Hlavním cílem práce je nalezení nejúspěšnějšího nastavení sítě, přičemž za metriku úspěšnosti byla zvolena plocha pod Precision-Recall křivkou označovaná dále jako AUC.

Bylo navrženo několik variant různých architektur založených na existujících řešeních, s přihlédnutím k prostorovému charakteru dat. Tabulka všech významnějších architektur je součástí přílohy. Nejlepší ze zde použitých architektur dosahuje téměř 90% úspěšnosti (0.8778 AUC). Architektura je tvořena pěti konvolučními vrstvami (což tvoří celkové zorné pole sítě o velikosti 32x32) následovanými rekurentní částí inspirovanou conditional random fields. V textu práce je uvedeno podrobnější vyhodnocení experimentů provedených nad dvěma rozdílně velkými datasety a jsou rozebrány možnosti dalšího pokračování této práce. Číselné výsledky nejvýznamějších experimetů lze nalézt v tabulkách, které jsou součástí šesté kapitoly.

Následující kapitoly obsahují stručný úvod do problematiky detekce resp. segmentace, konvolučních neuronových sítí a představení existujících řešení, ze kterých tato práce vychází. Dále jsou popsány zkoumané aspekty problému, například možnosti rozšíření omezeného datasetu, využití prostorového charakteru CT dat nebo použití rekurentní neuronové sítě inspirované conditional random fields (CRF).

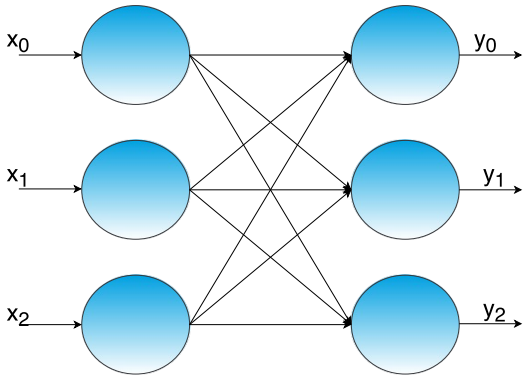
Kapitola 2

Umělé neuronové sítě

Umělé neuronové sítě jsou v dnešní době stále více zkoumaným odvětvím oboru umělé inteligence. Jejich historie sahá zhruba do doby, kdy v roce 1943 neurofiziolog W. McCulloch a matematik W. Pitts popsali způsob fungování liského mozku a pro ilustraci problému vytvořili model neuronové sítě pomocí elektrických obvodů [7]. Od té doby došlo s rozvojem výpočetní techniky a narůstajícím objemem a počtem trénovacích datasetů k vývoji umělých neuronových sítí od lineárního perceptronu, přes několikavrstvé plně propojené sítě až po aktuální stav, kdy existuje přehršel různých architektur specializujících se na řešení rozličných problémů a kdy lze trénování provádět na relativně snadno dostupných a výkonných grafických procesorech.

2.1 Hyperparametry, parametry a princip učení neuronových sítí

Lidský mozek se učí novým dovednostem opakovaným řešením problémů. Umělé neuronové sítě se snaží tento princip napodobit. V zásadě na trénování neuronových sítí lze pohlížet jako na hledání aproximace funkce, která pro určitý vstup dává požadovaný výstup. Takovýto model je definován svými parametry, jejichž hodnoty jsou hledány procesem trénování. U klasických plně propojených neuronových sítí patří mezi parametry tzv. váhy a biasy. Plně propojená síť je složena z neuronů, pro které platí že pro vstup x dávají vý-



Obrázek 2.1: Plně propojená neuronová síť.

stup y na základě rovnice 2.1, kde w je váha a b je bias daného neuronu. Tento výstup pak zpravidla bývá vstupem aktivační funkce, která má za úkol normalizaci výstupu a zároveň je nelinearitou nutnou pro umožnění aproximace složitějších problémů reálného světa.

$$y = \sum_{i=1}^{N} x_i \cdot w_i + b \tag{2.1}$$

Hyperparametry Pro odlišení od pojmu označujícího výše zmíněné parametry byl zaveden pojem hyperparametry, který je souhrnným označením pro veškerá nastavení týkající se architektury sítě i samotného procesu učení. Mezi hyperparametry patří například algoritmus pro trénování, loss funkce, použité vrstvy a jejich velikosti, počáteční nastavení parametrů, počet iterací a koeficient učení. Při hledání optimálního nastavení hyperparametrů je důležité nejen logické uvažování, ale také přehled o již existujících implementacích neuronových sítí a intuice vycházející z vlastních dřívějších experimetů.

Učení neuronových sítí Proces trénování je iterativní metoda a má dvě podstatné části. První částí je dopředný průchod, kdy je vstupní vrstvě sítě předložena informace a na základě hodnot parametrů jsou počítány výstupy jednotlivých vrstev až po vrstvu poslední. Výstup sítě je porovnán s požadovaným výstupem pomocí loss funkce a výsledná chyba je pak v druhé části procesu zpětně propagována sítí a je tak provedena odpovídající aktualizace hodnot parametrů podle rovnice 2.2, kde w_i označuje současnou hodnotu parametru a w_{i+1} hodnotu následující. Tento postup je (v případě konvergujícího trénování) iterativně opakován buď po pevně daný počet iterací, nebo dokud se síť ještě učí, což lze určit podle snižující se hodnoty loss funkce.

$$w_{i+1} = w_i + \Delta w_i \tag{2.2}$$

Optimalizátory Algoritmy pro trénování neuronových sítí jsou někdy označovány jako optimalizační algoritmy nebo optimalizátory. Zřejmě nejznámějším optimalizačním algoritmem je **gradient descent** nebo jeho varianta **stochastic gradient descent (SGD)**, kdy dochází k aktualizaci parametrů podle rovnice 2.3, kde g je gradient parametru v i-té iteraci a η je koeficient učení, jehož hodnotu je v případě použití SGD nutné experimentálně nalézt. Existují ale také optimalizační algoritmy, které nalezení hodnoty koeficientu učení částečně zajišťují samy. Jedním z těchto algoritmů je **AdaDelta**, což je také gradientní metoda, která však v průběhu trénování zohledňuje gradienty z nedávné historie trénování dané oknem o pevně stanované velikosti a podle jejich velikosti upravuje míru v jaké budou parametry aktualizovány [14].

$$\Delta w_i = -\eta g_i \tag{2.3}$$

Loss funkce Loss funkcí používanou pro výpočet chyby klasifikačních úkolů je **Logarithmic loss** (jinak také **Log loss** či **Cross–Entropy loss**)[1]. Obecný vzorec 2.4 pro výpočet chyby pomocí Log loss pro M tříd lze aplikovat i na binární problémy typu segmentace kontury, kde x_n je odhad pravděpodobnosti pro daný pixel a y_n příslušný label. Vzorec potom vypadá následovně 2.5. Díky zápornému logaritmu je penalizace obzvlášť vysoká u krajně nepřesných odhadů.

$$E = -\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} y_{nc} log(x_{nc}) \tag{2.4}$$

$$E = -\frac{1}{N}\sum_{n=1}^{N}[y_n log(x_n) + (1-y) log(1-x_n)] \tag{2.5}$$

Aktivační funkce Jak už bylo řečeno, aktivační funkce zajišťuje nelinearitu sítě, která by bez její přítomnosti byla v podstatě pouhým lineárním klasifikátorem, který není dobře použitelný k aproximaci složitých problémů. Dalším efektem aktivační funkce je normalizace hodnot získaných podle vzorce 2.2 a celkově tedy určení, při jakých hodnotách dochází k aktivaci neuronu. Při výběru aktivační funkce lze volit z relativně velkého množství používaných variant. Pro tuto práci byly vybrány dvě z nich, **ReLU** definovaná vzorcem 2.6, u které dochází k aktivaci pouze pro kladné honoty a **sigmoida** 2.7, která normalizuje hodnoty do rozsahu $\langle 0; 1\rangle$, což je ideální, pokud chceme aby výstupem sítě byl odhad pravděpodobností.

$$ReLU(y) = max(0, y) \tag{2.6}$$

$$\sigma(y) = (1 + exp(-x))^{-1} \tag{2.7}$$

Metriky pro vyhodnocení úspěšnosti Dvěma pomocnými metrikami, které jsou běžně používány k vyhodnocování experimentů z oblasti hlubokého učení jsou precision a recall. **Precision**, jak už název napovídá, je metrikou zhodnocující jak úspěšné bylo zařazení do správné třídy podle 2.8, kde TP znamená true positive a FP naopak false positive. Význam **recall** metriky pak udává, kolik bylo nalezeno hledaných vzorků z celkového množství podle 2.9 s FN znamenajícím false negative.

$$precision = \frac{TP}{TP + PF} \tag{2.8}$$

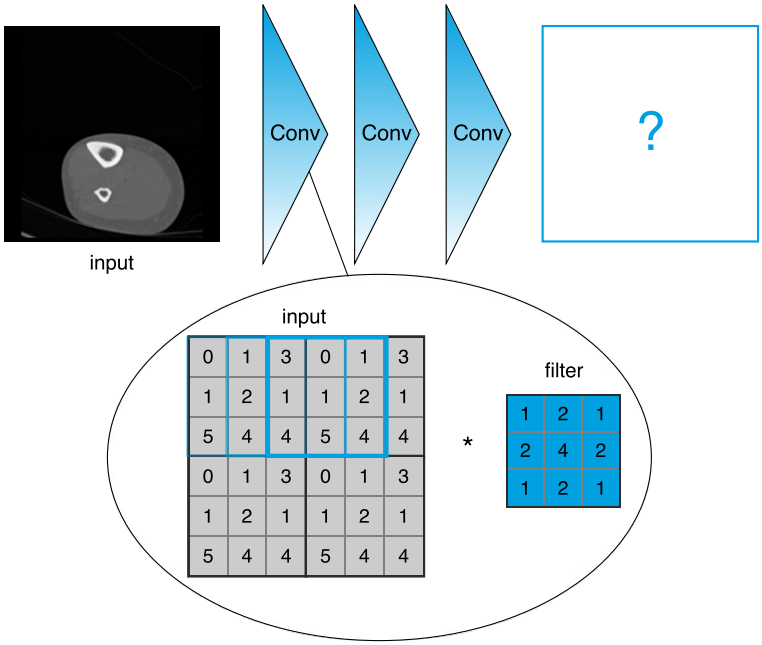
$$recall = \frac{TP}{TP + FN} \tag{2.9}$$

Výstupem sítě jsou odhady pravděpodobnosti příslušnosti do třídy kontura v rozmezí $\langle 0; 1\rangle$ pro každý pixel. Protože ze samotných odhadů pravděpodobnosti nelze jednoznačně určit, do které třídy pixel ve výsledku patří, je možné stanovit v tomto intervalu práh t, jehož překročení udává rozhranní mezi třídami pozadí a kontura. Odhady v intervalu $\langle 0; t\rangle$

[1] Viz dokumentace Cross–Entropy Loss vrstvy v BVLC Caffe http://caffe.berkeleyvision.org/doxygen/classcaffe_1_1SigmoidCrossEntropyLossLayer.html

pak připadnou do třídy pozadí, naopak odhady o hodnotách $\langle t; 1\rangle$ budou znamenat, že daný pixel náleží třídě kontura. Postupným nastavováním prahu od nejnižšího po nejvyšší a současným výpočtem metrik precision a recall získáme Precision–Recall křivku (PRC). Postupný vývoj PRC při trénování detekce kontur ukazuje obrázek 6.1. Je patrné, že čím dokonalejší je klasifikátor, tím více se precision i recall blíží hodnotě 1 současně u všech hodnot prahů a tím větší je plocha pod křivkou (AUC). V teoretickém případě dokonalého klasifikátoru by plocha byla $1 \cdot 1 = 1$, což znamená 100% úspěšnost.

2.2 Konvoluční neuronové sítě



Obrázek 2.2: Konvoluční neuronová síť.

Protože s narůstajícím počtem vstupů (v případě zpracování obrazu jednotlivých pixelů obrázku) roste i počet propojení a tedy počet parametrů, které je nutné trénovat, nejsou plně propojené sítě pro oblast zpracování obrazu vhodným řešením. Dalším výsledkem použití plně propojených architekutr je mnohdy nežádoucí přikládání důležitosti pozici v rámci obrazu. Proto je pro tuto oblast vhodnější varianta konvolučních neuronových sítí. Na rozdíl od plně propojených vrstev jsou váhy konvolučních vrstev uspořádány do matic resp. konvolučních jader s (obvykle) lichým rozměrem. Každá vrstva je tvořena konkrétním počtem těchto jader, která jsou se zvoleným krokem přikládána na vstupní data, přičemž je při každém přiložení prováděna operace konvoluce. Výsledek, jímž je pro každé přiložení jedna hodnota, je umístěn na odpovídající pozici výstupního obrazu vrstvy (obrázek 2.2). Počet výstupů konvoluční vrstvy pak odpovídá počtu jejích jader, které také bývají označovány jako filtry. Za hyperparametry konvolučních vrstev tedy lze považovat počet filtrů, velikost konvolučního jádra, krok konvoluce a v případě nutnosti také rámec nul (neboli padding), kterým lze zajistit, aby došlo k zachování stejné velikosti vstupu i výstupu.

Současně s konvolučními vrstvami bývají často používány **pooling** vrstvy. Typickým zástupcem je **max–pooling**, kdy je s daným krokem, stejně jako u konvolučních vrstev, posouváno po vstupu vrstvy okno dané velikosti. Tentokrát je ale hodnotou, která bude na odpovídajícím indexu výstupu vrstvy, maximum z tohoto okna. Tímto způsobem jsou data procházející sítí redukována pouze na nejsilnější aktivace, což potlačuje šum, zachovává jenom podstatnější informace a může tak napomáhat lepší klasifikaci. Oproti tomu využití poolingu pro segmentaci nemusí být vždy nejlepší řešení, protože tak dochází ke ztrátě informace o prostorovém kontextu a zároveň dochází k výraznému zmenšení rozlišení, což je následně třeba řešit různými upsampling metodami.

Využití těchto sítí je v dnešní době rozsáhlé. Podle současného vědce z oblasti strojového učení Andrewa Ng [2] lze předpokládat, že dnešní umělá inteligence je schopná zvládnout obvyklé úkoly, které průměrnému člověku trvají několik sekund. Konvoluční neuronové sítě už byly použity například k odhadu pozice člověka [10], hraní arkádových her [8] nebo generování objektů podle jejich popisu [2]. Klasickým a stále hodně diskutovaným úkolem pro konvoluční neuronové sítě je však klasifikace objektů [5]. Vzhledem k tomu, že detekce kontur pracuje s obrazovými daty a lze ji chápat také jako druh klasifikace, jsou konvoluční neuronové sítě slibným řešením i tohoto problému.

[2] Přednáška *Nuts and Bolts of Applying Deep Learning* v rámci Deep Learning School, Stanford https://www.bayareadlschool.org/

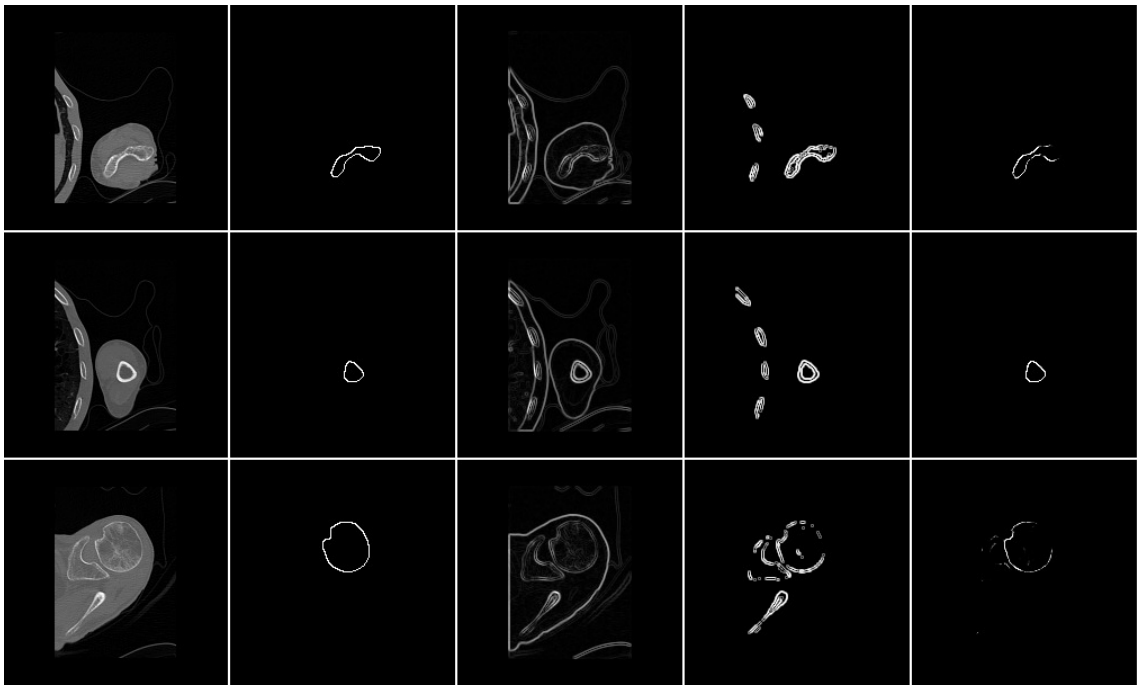
Kapitola 3

Metody pro detekci kontur a segmentaci obrazu

Obrázek 3.1: Detekce kontur, (zleva) původní obrázek, klasická detekce hran, detekce pomocí encoder-decoder sítě. Převzato z [13].

Pojmem kontura se rozumí rozhranní mezi zobrazeným objektem a jeho okolím resp. linka pixelů patřících objektu, která se tohoto rozhranní dotýká (obrázek 3.1). Kontury odpovídají hranicím celého reálného objektu na rozdíl od hran, které se vyskytují, kdekoliv se v obraze zlomově mění barva, jas, či textura, a mohou se tedy nacházet i uvnitř objektu či naopak nemusí být na hranicích objektů dostatečně patrné. Naivním přístupem k detekci kontur by mohlo být například použití prahování v kombinaci s hranovými filtry. Ukázku použití sobelova operátoru z python modulu scikit–image [12] společně s prahováním můžeme vidět na obrázku 3.2. Vzhledem k povaze konvolučních neuronových sítí, které dobře fungují jako detektory objektů a které jsou v podstatě spoustou potenciálních hranových filtrů, lze očekávat jejich úspěšné uplatnění i pro tento úkol.

Na problém detekce kontur i problém segmentace je možné pohlížet jako na klasifikaci, při které je každý pixel obrazu klasifikován zvlášť. To znamená, že pro každý pixel existuje zařazení do jedné ze stanovených tříd klasifikace. Label pro celý obraz má potom stejnou

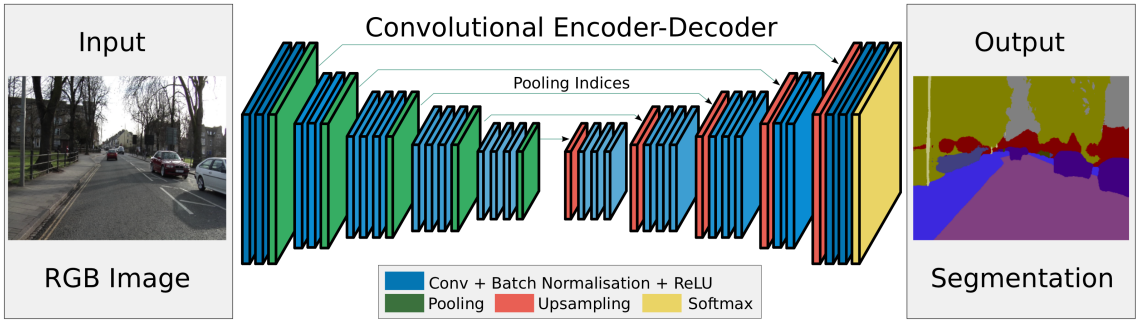


Obrázek 3.2: Zleva: vstupní vrstva CT skenu, snímek s anotací, ukázka použití sobelova operátoru, sobelův operátor použitý současně s prahováním a výstup nejlepší nalezené architektury neuronové sítě.

výšku a šířku jako obraz samotný. V případě segmentace může podle počtu hledaných objektů existovat několik tříd klasifikace. V případě detekce kontur existují třídy dvě – kontura a pozadí. Současná řešení problému segmentace, která jsou také hlavní inspirací této práce, jsou popsána v následující podkapitole.

3.1 Plně konvoluční šíť pro segmentaci (FCN)

V roce 2014 J. Long et al. adaptovali několik klasických klasifikačních sítí pro úkol semantické segmentace [6]. Plně propojená část na konci klasifikátoru je převedena na konvoluční a tím je dosaženo plně konvoluční architektury, jejímž výstupem jsou labely pro jednotlivé pixely tvořící label pro celý obraz. Protože průchodem relativně hlubokou konvoluční sítí s poolingem dochází k redukci výšky a šířky vstupu, je na konec sítě přidána dekonvoluční část tvořená pevně danými filtry pro bilineární interpolaci, která zajišťuje upsampling na velikost původního vstupu. Je zde také použita myšlenka kombinování informace z různě zanořených vrstev pro poskytnutí lokálního i globálního kontextu. Kanály feature map z nižších vrstev jsou redukovány pomocí 1x1 konvoluce a přičteny k feature mapám vstupujícím do dekonvoluční části. Na obrázku 3.6 je vidět rozdíl mezi dvěma variantami použití dekonvoluce, z nichž v prvním případě jsou dekonvoluční jádra trénována, zatímco v případě druhém je používá pouze bilineární interpolace.



Obrázek 3.3: Architektura sítě SegNet. Převazato z [1].

3.2 SegNet encoder-decoder

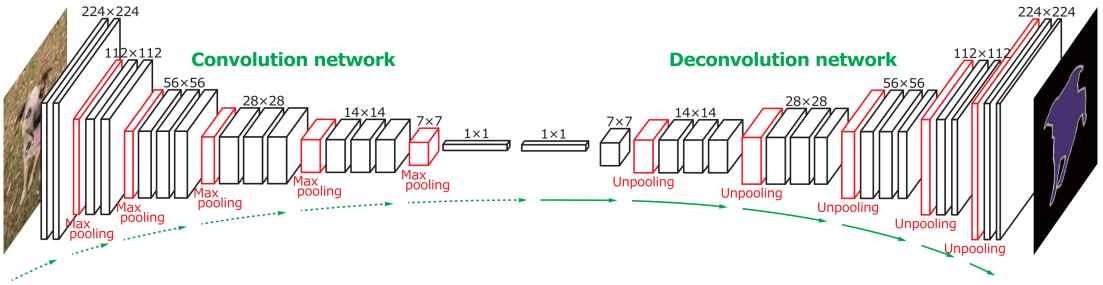
Encoder-decoder architektura je založená na následující myšlence: encoder je zde v podstatě běžná konvoluční síť, která má za úkol extrahovat (zakódovat) podstatné informace reprezentující vstup. Dekoder je síť složená z vrstev pro upsampling a zajišťuje převedení zakódované informace na požadovanou novou reprezentaci. V případě SegNetu [1] je architektura encoderu identická se sítí VGG16 [11] (kromě plně propojené části), decoder je v podstatě zrcadlovou sítí k encoderu. Pro upsampling decoder využívá indexy encoderových max-pooling vrstev, čímž mimo jiné snižuje počet trénovaných parametrů, a rovněž feature mapy z odpovídajících vrstev encoderu.

Autoři SegNetu navazují na FCN a zaměřují se na paměťově i výpočetně efektivní architekturu použitelnou pro (real-time) aplikace analyzující prostředí ulice. Výsledky jejich experimentů naznačují, že použití max-pooling indexů, případně použití neredukovaných feature map pro upsampling, vede k dobrým výsledkům segmentace. V rámci práce byl také proveden benchmark existujících architektur z něhož vybrané ukázky můžeme vidět na obrázku 3.6. Je patrné, že SegNet dosahuje nejdetailnější segmentace ze zde porovnaných architektur a je úspěšný i u drobných objektů, jakými jsou tyče lamp pouličního osvětlení.

3.3 DeconvNet

Autoři [9] upozorňují na fakt, že starší architektury pro segmentaci mají tendenci přikládat význam velikosti objektu. To má za následek, že velké objekty jsou někdy ve výsledné segmentaci rozbity na několik menších (špatně klasifikovaných) objektů, zatímco malé objekty jsou přehlídnuty a zaměněny za pozadí. Autoři se snaží tomuto problému předcházet trénováním sítě na „instancích“ , což jsou výběry z původních obrazů obsahující objekt.

Jejich architektura (obrázek 3.4) je v podstatě encoder-decoder, kdy jsou obě části vzájemně zrcadlové. Encoder (opět předtrénovaná VGG16), zde označován také jako extraktor příznaků, převádí vstupní obraz na mnohokanálovou reprezentaci o velikosti 1x1. Tato reprezentace je vstupem decoder části, která slouží jako „generátor tvarů“ a jejímž výstupem je výsledná maska segmentace. Decoder část je v tomto případě složena z unpoolingu, dekonvoluce a ReLU, a na rozdíl od dřívějších řešení (FCN) je i ona trénována. Unpooling (obdobně jako u SegNetu) využívá uložených indexů, díky kterým je možné zhruba zrekonstruovat původní tvar objektů. Po něm následuje dekonvoluční vrstva, která vyhlazuje hrubý výstup unpoolingu. Dále autoři používají batch normalization a dvoufázové tréno-



Obrázek 3.4: Architektura sítě DeconvNet. Převazato z [9].

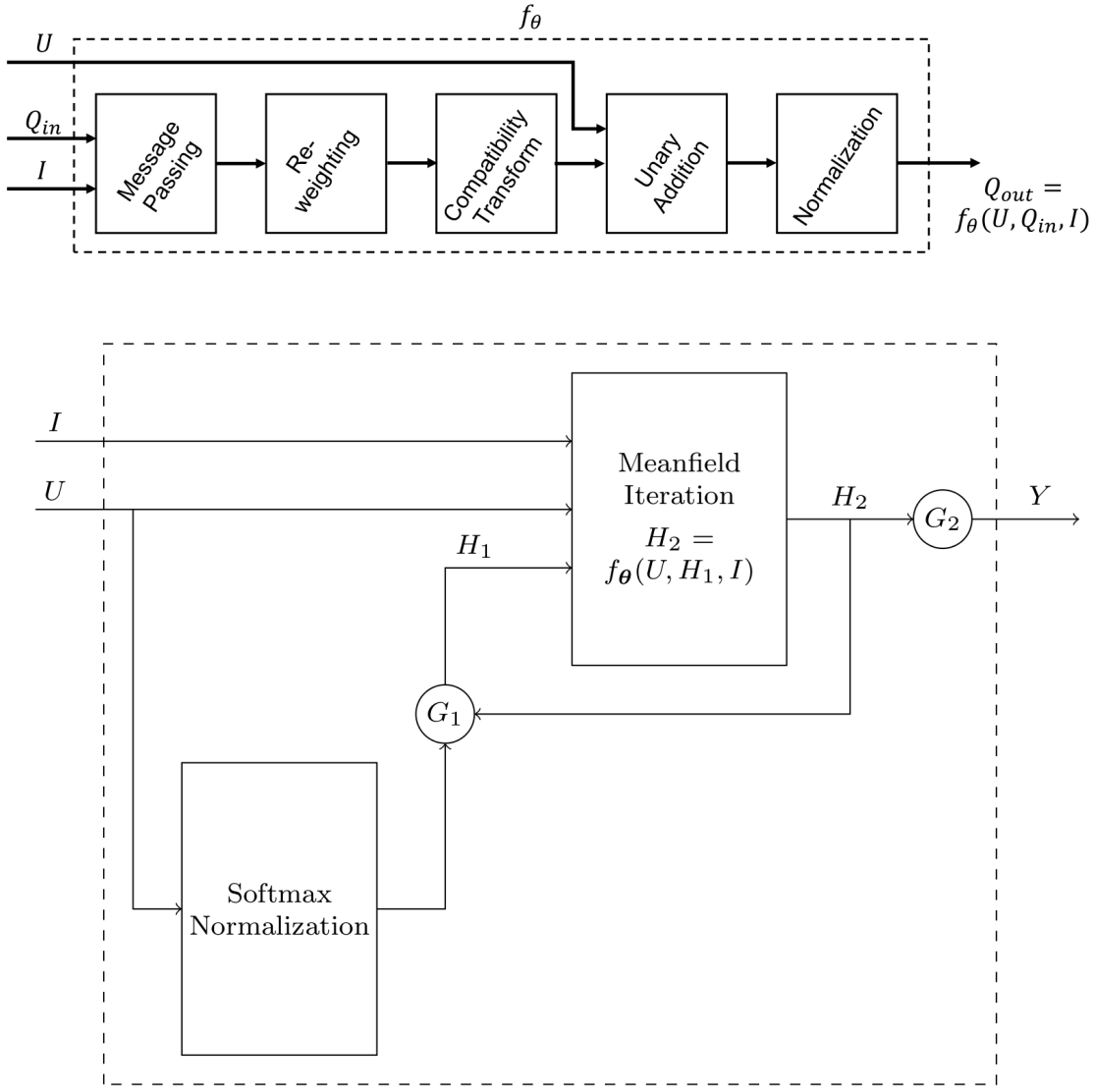
vání, kdy v první fázi je síť trénována na jednodušších případech (objekt je vycentrován a ořezán podle bounding-boxu) a na složitějších případech až ve fázi druhé. Tyto přístupy mají za úkol pomoci natrénovat hlubokou síť pomocí relativně malého datatsetu (celkem 12031 PASCAL trénovacích a testovacích obrazů).

3.4 Detekce kontur pomocí plně konvoluční encoder-decoder architektury

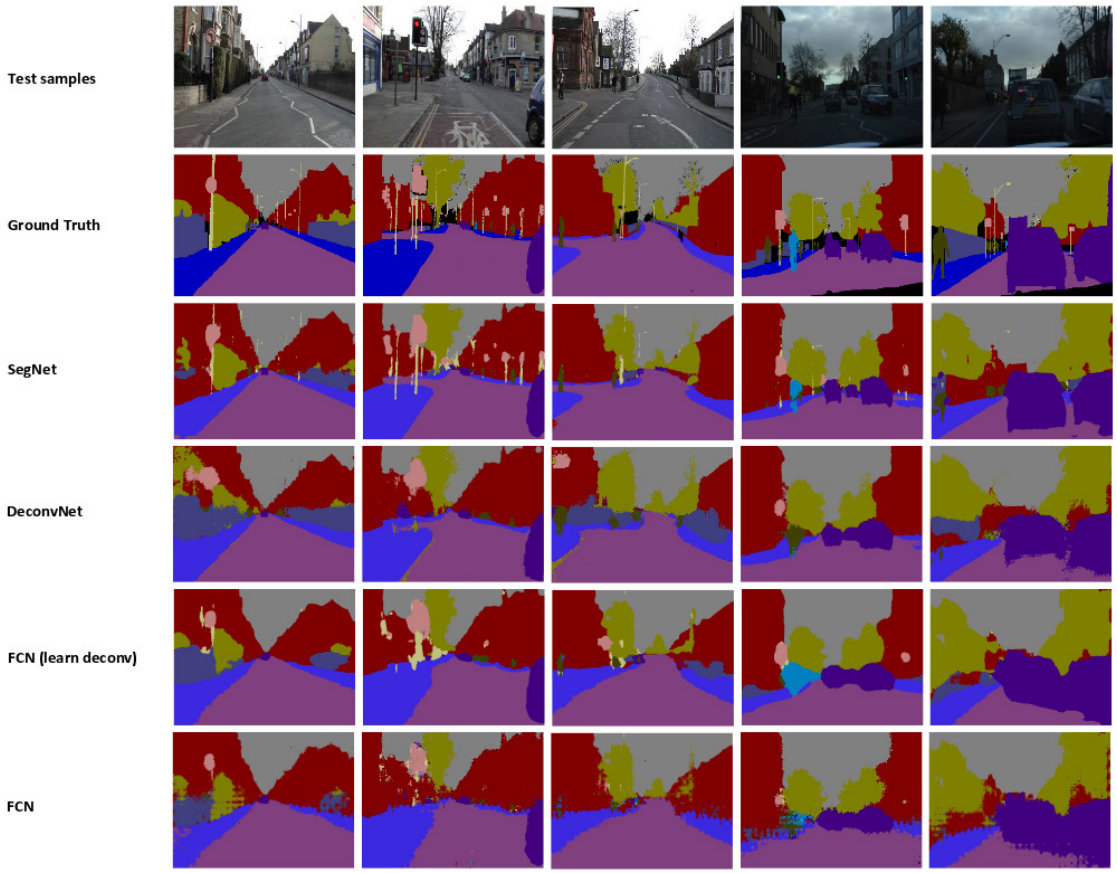
Podobného přístupu, jako autoři výše zmíněných architektur pro segmentaci, využívá [13] pro detekci kontur objektů jakými jsou postavy, zvířata a další běžně používané třídy klasifikace. Problém je formulován jako klasifikace pixelů do dvou tříd – kontura a non-kontura. Autoři připomínají, že i přes existenci spousty metod pro detekci hran je jen málo z těchto řešení zaměřených na detekci hran objektů, kdy méně významné hrany jsou ignorovány. Encoder je opět síť VGG16, která tentokrát zůstává fixována v průběhu trénování, kdy se tedy trénuje pouze decoder část. Protože hlubokou encoder-decoder síť může být těžké natrénovat, navrhují autoři „odlehčenou“ verzi encoderu a tedy nesymetrickou encoder-decoder architekturu. Kvůli nevyváženosti výskytu tříd kontura a non-kontura je použit class balancing 1:10. Autoři tvrdí, že jejich síť dosahuje oproti předchozím řešením problému detekce kontur výrazně lepší přesnosti.

3.5 Conditional Random Fields a jejich implementace pomocí konvolučních sítí

Conditional random fields jsou metodou často používanou pro segmentaci či vyhlazování a zpřesňování výsledků segmentace. Pro trénování CRF je možné použít mean–field algoritmus, který může být implementován jako rekurentní neuronová síť. Autoři [15] navrhují model složený ze dvou částí, klasické konvoluční sítě a CRF sítě, kde sada několika vrstev implementujících mean–field algoritmus představuje jednu iteraci tohoto algoritmu. Tato sada je pak opakovaně skládána za sebe, čímž je simulováno iterativní opakování mean–field algoritmu. Detailnější schéma použití konvoluční implementace CRF je patrné na obrázku 3.5.



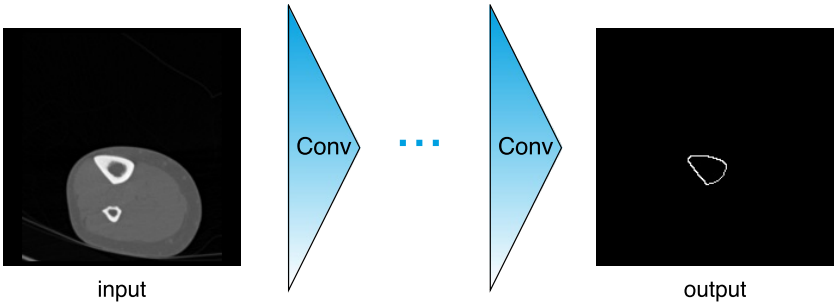
Obrázek 3.5: Mean–field algoritmus a jeho připojení ke konvoluční síti pro segmentaci. Převazato z [15].



Obrázek 3.6: SegNet Benchmark současných architektur pro segmentaci, převzato z [1] a upraveno.

Kapitola 4

Návrh konvoluční neuronové sítě pro detekci kontur v obrazech výpočetní tomografie

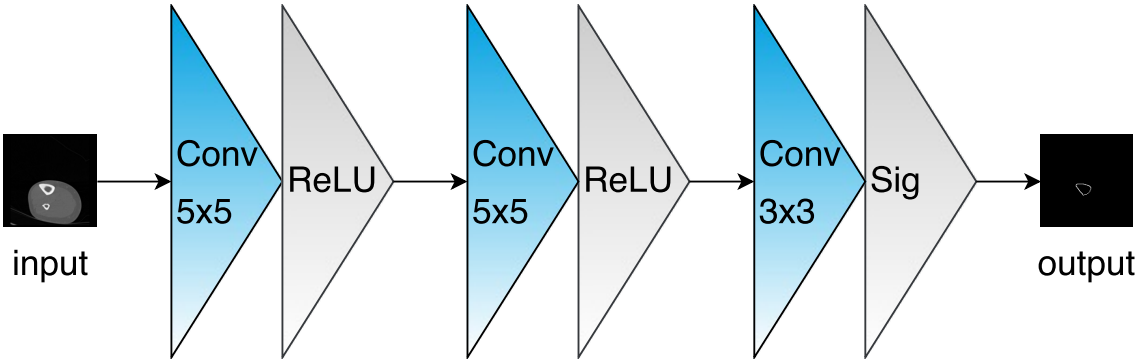


Obrázek 4.1: Ilustrace problému detekce kontur.

Jak už bylo řečeno, detekce kontur je prováděna na snímcích výpočetní tomografie neboli CT skenech. Cílem práce je experimentovat s různými druhy architektur konvolučních neuronových sítí a nalézt takové nastavení, které umožní co nejpřesněji segmentovat kontury dlouhých lidských kostí humeru a tibie. Ilustraci úkolu detekce kontury můžeme vidět na ukázkové vrstvě CT skenu tibie 4.1.

4.1 Data výpočetní tomografie

Výpočetní tomografie je jakousi rozšířenou variantou klasické rentgenové vyšetřovací metody. Pacient je v tomto případě snímán zařízením, které jej obíhá po kruhové trajektorii a umožňuje tak vypočítat prostorový model vnitřního složení jeho těla. Ten je následně jako matice čísel v Hounsfieldově jednotce uložen ve standardizovaném formátu DICOM pro uchovávání a síťový přenos biomedicínských obrazových dat. Hounsfieldova jednotka je obecně používaná jednotka radiodenzity s referenční hodnotou 0 pro destilovanou vodu. Relevantní hodnoty potom jsou od -1000 pro vzduch a cca 3000 pro kompaktní kost. Pro zobrazení CT dat je použit menší rozsah hodnot, protože běžné lidské oko nedokáže rozlišit 4000 stupňů šedi [16].



Obrázek 4.2: Baseline: základní použitá architektura.

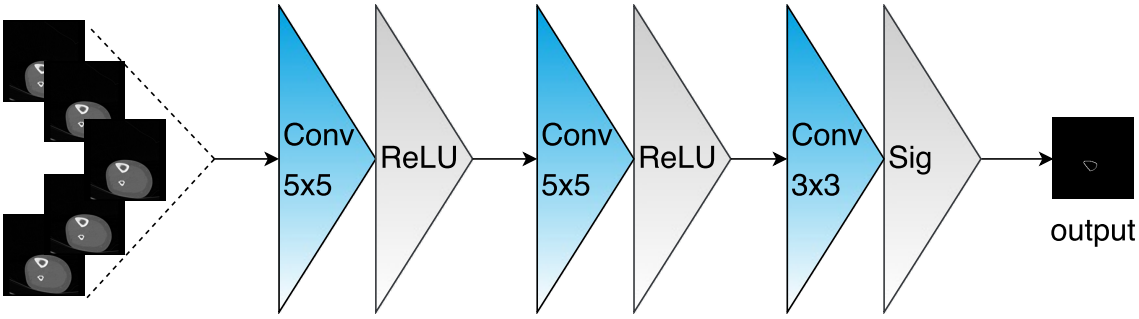
4.2 Architektury

Za účelem detekce kontur bylo navrženo několik následujících architektur, které čerpají z již existujících řešení a zároveň se snaží využít prostorového charakteru analyzovaných dat. Základní použitou architekturou je jednoduchá třívrstvá konvoluční síť dále označovaná jako **Baseline** (obrázek 4.2). Vrstvy této archutektury mají filtry (od vstupní vrstvy) 5x5, 5x5 a 3x3 (dále zkrácený zápis $\{5, 5, 3\}$), všechny s krokem 1. Vrstvy `conv1` `conv2` mají výstup 32 kanálů, čemuž odpovídá počet jejich filtrů. Po konvolučních vrstvách ve většině případů následuje aktivační funkce ReLU, definovaná jako $ReLU(x) = \max(0, x)$. V poslední vrstvě je ReLU nahrazena sigmoidou pro normalizaci odhadů pravděpodobností do rozsahu $\langle 0; 1 \rangle$. Vstupem sítě je minibatch o jednom kanálu a (při trénování) o výšce i šířce 200.

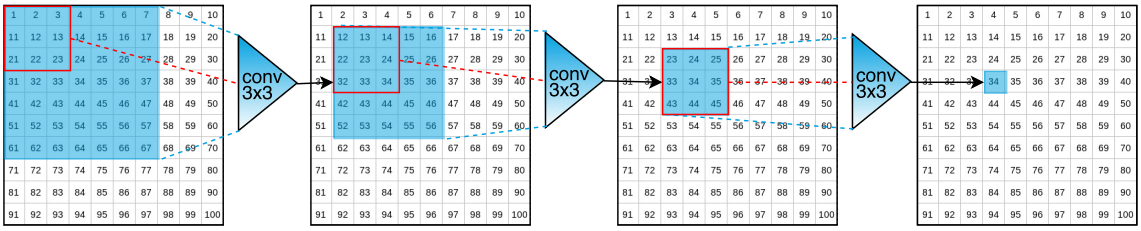
Využití prostorového charakteru dat Vstup sítě může samozřejmě mít i více kanálů, běžné jsou 3 kanály barevných RGB obrázků. Konvoluční jádro překrývá vždy všechny kanály zároveň. Pro vstup o třech kanálech by potom první vrstva Baseline architektury měla konvoluční jádra o rozměrech 3x5x5. Architektury **3ch** resp. **5ch** využívají prostorového charakteru CT snímků a vstupem sítě je v tomto případě 3 resp. 5 po sobě následujících vrstev skenu, předaných síti jako vstup o více kanálech (obrázek 4.3). Předpokládá se, že přidání kontextu ve směru další osy může pomoci přesnosti segmentace. Tento přístup se na použitých datasetech příliš neosvědčil, což je blíže popsáno v kapitole 6.

Dalším způsobem, jak přidat informaci o prostorovém kontextu, by mohlo být zavedení dalších tří kanálů, které by pro každý voxel udávaly vzdálenost od ručně označených konců kosti v jednotlivých osách. Toto řešení nebylo v rámci dosavadní práce implementováno a je tedy možným budoucím rozšířením.

Zorné pole Experimenty jsou prováděny také se zorným polem sítě (obrázek 4.4). Zorné pole jednoho filtru má stejnou velikost jako filtr samotný. Skládáním zorných polí při průchodu jednotlivými vrstvami sítě je tvořeno celkové zorné pole sítě. Baseline architektura má tedy zorné pole 11x11. Zvětšení zorného pole lze dosáhnout buďto větším rozměrem filtrů nebo přidáním dalších vrstev. Tímto rovněž přibývá počet parametrů sítě, což může také pomoci lépe aproximovat řešený problém. Architektura **BigKernels** je obdobou Baseline, avšak používá filtry o rozměrech $\{11, 5, 3\}$, čímž se zorné pole zvětšuje na 17x17. Oproti



Obrázek 4.3: Využití prostorového charakteru dat.

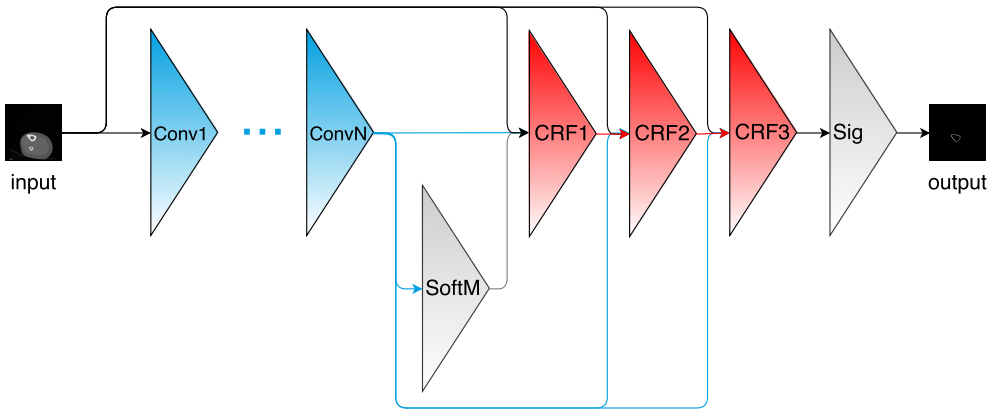


Obrázek 4.4: Znázornění zorného pole sítě.

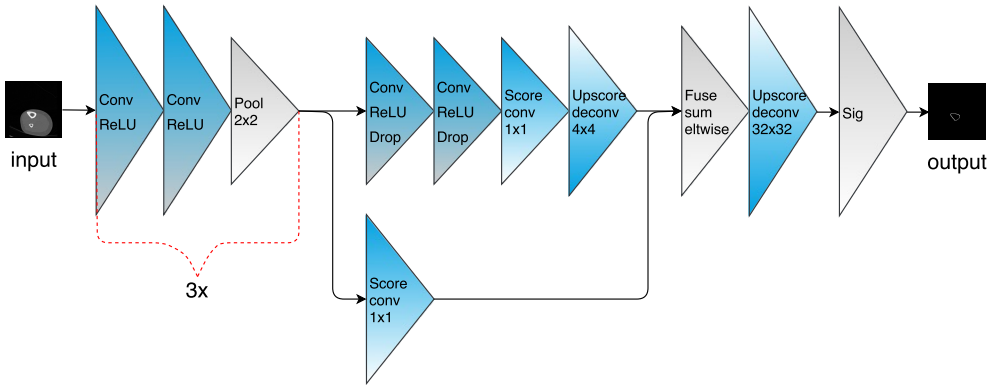
tomu architektura **Net5** je tvořena pěti vrstvami o filtrech $\{5, 5, 3, 3, 3\}$ a tedy zorným polem 15x15. Nakonec je použita i kombinace obou přístupů **Net5_bigKernels** s filtry $\{11, 11, 5, 5, 3\}$ a zorným polem 31, která dosahuje jedné z nejvyšších přesností.

Inspirace v CRF Pro segmentaci či zpřesnění výsledků segmentace se často používají metoda conditional random fields (CRF). Autoři článku [15] navrhli postup, jakým modelovat conditional random fields pomocí konvolučních sítí. Protože cílem této práce není dopodrobna modelovat CRF, je použita upravená varianta CRF, která na rozdíl od původního návrhu nekopíruje krok za krokem mean-field algoritmus. Místo toho je použito několik homogenních rekurentních vrstev (obrázek 4.6), přičemž informace vstupující do této části sítě jsou stejné jako u autory navrženého postupu.

Encoder-decoder architektury Jako okrajový experiment byla vyzkoušena také architektura obsahující dekonvoluční část. Vzorem pro tuto architekturu byla jedna ze sítí autorů [6]. Síť je celkově menší než původní varianta, přičemž bilineární interpolace v dekonvoluční části byla nahrazena trénováním dekonvolučních jader, jak doporučují autoři [1]. Výsledná architektura nebyla oproti ostatním navrženým příliš úspěšná, což může být způsobeno malým datasetem, který není dostačující pro natrénování větší sítě, která navíc nemá výhodu předtrénování, tak jak tomu je u FCN využívající předtrénované VGG16. Tato práce není cílená na experimenty s encoder-decoder přístupem a proto nebylo vynaloženo úsilí na zlepšení výsledků tohoto druhu architektury, ač by to mohlo být do budoucna vhodným rozšířením práce.



Obrázek 4.5: Architektura inspirovaná metodou conditional random fields.

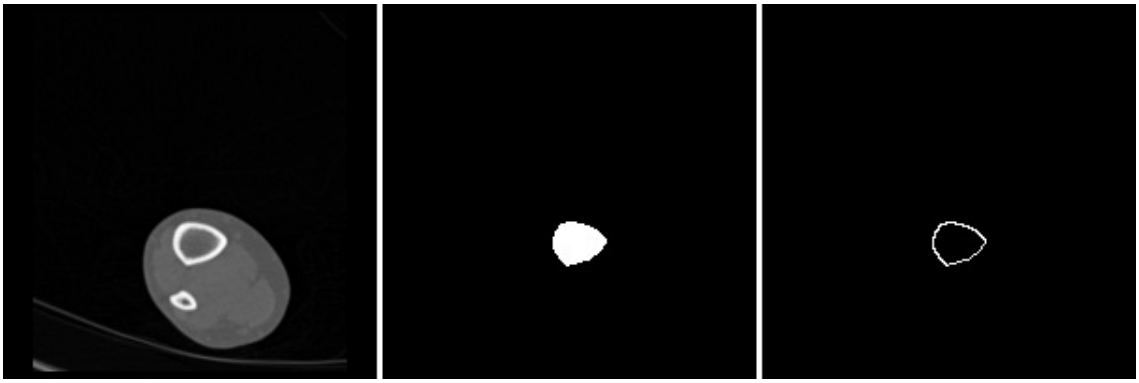


Obrázek 4.6: Architektura inspirovaná FCN encoder-decoder architekturou.

4.3 Datasety

Firma 3Dim s.r.o. poskytla pro tuto práci dva datasety CT snímků ve formátu DICOM. První dataset je tvořen snímky holenní kosti, neboli tibie. Dataset **Tibia** tvoří celkem 10 skenů, z nichž byl jeden (složený z 928 vrstev) vybrán jako testovací a zbytek (9 skenů resp. 8856 vrstev) jako trénovací podmnožina. Dataset **Humerus** obsahuje snímky kosti pažní, neboli humeru. Tento dataset tvoří 51 skenů a pro zachování podobného poměru jako u Tibie bylo zařazeno 5 skenů (resp. 3102 vrstev) do testovací podmnožiny a 46 skenů (37100 vrstev) do podmnožiny trénovací. Několik experimentů bylo provedeno také na **smíšeném** datasetu, ale vzhledem k rozdílnosti obou druhů snímků nebylo dosaženo příznivých výsledků a od rozsáhlejších experimetů bylo upuštěno.

Anotace K oběma datasetům přísluší nejen samotné CT snímky, ale také ručně vytvořené anotace daných kostí. Původní anotace určené pro segmentaci byly pro potřeby experimentů upraveny na pouhé kontury. Význam anotace je následující: pro každý voxel původního skenu existuje jeho label označující, zda tento patří do třídy „kontura“ (hodnota „1“) nebo ne (hodnota „0“) (obrázek 4.7).



Obrázek 4.7: Zleva: vrstva CT skenu, původní anotace pro segmentaci kosti, nová anotace pro detekci/segmentaci kontury kosti

Rozšíření datasetů Pro trénování hlubokých konvolučních sítí se běžně používají relativně objemné datasety. Příkladem může být ImageNet [1], který dosahuje celkovým počtem obrázků řádu milionů. U problémů, které nejsou tak často zkoumány (a neexistují tedy snadno dostupná data o takovémto objemu), jsou uplatňovány různé metody umělého rozšíření datasetů. Manipulace s daty musí pochopitelně být taková, aby nedocházelo ke vzniku nereálných případů a nebyla tak natrénovaná síť negativně ovlivněna co do schopnosti generalizovat. Pro zde použité datasety CT snímků tak bylo navrženo několik jednoduchých transformací. Možnými transformacemi jsou rotace, zrcadlové převracení, různé druhy zvětšení či protažení v určitém směru. Je také možné manipulovat s kontrastem, či přidávat do obrazů náhodný šum. V tomto případě byly s přihlédnutím k reálným možnostem CT skenu zvoleny následující transformace:

- náhodné posunutí v rozsahu $(-30; 30)$ nezávisle ve vodorovné a svislé ose,
- náhodná rotace v rozsahu $(-30; 30)$ kolem středu obrazu,
- náhodné protažení v rozsahu $(0; 50)$ nezávisle ve vodorovné a svislé ose.

Poslední zmíněná transformace zajišťuje tedy jak deformaci v určitém směru, tak celkové zvětšení obrazu. Změny kontrastů a podobné manipulace s číselnými hodnotami v samotných snímcích jsou diskutabilní vzhledem ke standardizovanému kódování dat v Hounsfieldově jednotce, proto tyto transformace zatím nebyly zavedeny. Jejich zavedení ale není vyloučeno vzhledem k možným rozdílům v absorpčním koeficientu kostí jednotlivých pacientů. Rozsahy transformací byly zvoleny intuitivně s přihlédnutím k možným polohám pacienta ve snímacím zařízení a k realistickým rozměrům pacientova těla. Deformace jsou prováděny na trénovacím datasetu dynamicky a náhodně přímo před předáním snímků neuronové síti.

[1] http://image-net.org/

Kapitola 5

Realizace

Praktická část této práce, provedená podle výše navrhovaných řešení, byla realizována pomocí jazyka Python 2.7.6 za použití modulu pycaffe, který je součástí frameworku BVLC Caffe pro hluboké učení [4]. Za stěžejní skript je považován trénovací skript, pomocí kterého byly realizovány veškeré experimenty. Sítě typu deploy, zbavené vrstev použitých pouze pro trénování, pak umožňuje použít k tomu určený skript, který však s hlavním skriptem sdílí spoustu funkcí. Kromě těchto hlavních dvou částí byla vytvořena sada pomocných skriptů v jazycích Python a GNU bash 4.3.11, sloužících k úpravám datové sady.

Trénování probíhalo pod operačním systémem Ubuntu 14.04, který je obecně známý dobrou kompatibilitou s Caffe frameworkem, za použití GPU MSI GTX 1060 s pamětí o velikosti 6GB.

5.1 Caffe framework a pycaffe

Pro trénování neuronových sítí již dnes existuje několik v praxi používaných frameworků. Nejznámějšími jsou Torch[1], Theano[2] nebo třeba TensorFlow[3]. Pro tuto práci byl zvolen framework BVLC Caffe. Tento open source framework nabízí možnost trénování na GPU s podporu CUDA (platforma pro paralelní výpočty) a CuDNN (Deep Neural Network library), zahrnuje implementaci běžně používaných vrstev neuronových sítí (včetně vrstev pro pomocné operace, například ořezaní či element-wise operace), optimalizační algoritmy a v neposlední řadě také několik referenčních modelů. Samozřejmostí je možnost použít framework jak k trénování, tak k fine-tunningu již existujících vlastních sítí, či volně dostupných sítí vyvíjených v rámci různých výzkumů. Je běžně využíván jak studenty, tak odbornými týmy z oblasti výzkumu umělé inteligence [6].

Reprezentace dat Data jsou v Caffe reprezentována tzv. bloby o rozměrech $N \times K \times H \times W$. N je počet vzorků (například obrázků) vstupujících do sítě, jinak označovaný také jako batch size. K je počet kanálů, například $K = 3$ pro jednotlivé barvy RGB obrazu, jak už bylo uvedeno výše, případně tři následující vrstvy CT skenu. Počet kanálů

[1] http://torch.ch/
[2] http://deeplearning.net/software/theano/
[3] https://www.tensorflow.org/

Obrázek 5.1: Dva vyskytující se případy trénování pomocí SGD.

také udává jeden z rozměrů konvolučních jader vrstvy, do které blob vstupuje. Naopak počet výstupů vrstvy pak udává počet kanálů blobu z ní vystupujícícho. $H \times W$ je výška a šířka jednotlivých vzorků.

Soubory K definici sítě Caffe používá formát Protocol Buffer (resp. protobuf[4]) , což je formát pro serializaci struktur vyvinutý společností Google. Definice možné podoby takto uložených dat je dán souborem `caffe.proto` a na základě takto definovaného formátu může být vytvořen vlastní soubor běžně označován příponou `.prototxt` obsahující definici sítě.

Druhým konfiguračním souborem je tzv. solver. Ten obsahuje nastavení hyperparametrů trénovaní, jakými jsou například cesta k definici sítě, použitý optimalizátor, jeho nastavení, vypnutí či zapnutí debugovacích informací a další. Tento soubor je běžně označen stejnou příponou jako definice sítě. V příloze F naleznete vzorovou definici vrstvy caffe frameworku ve formátu protobuf.

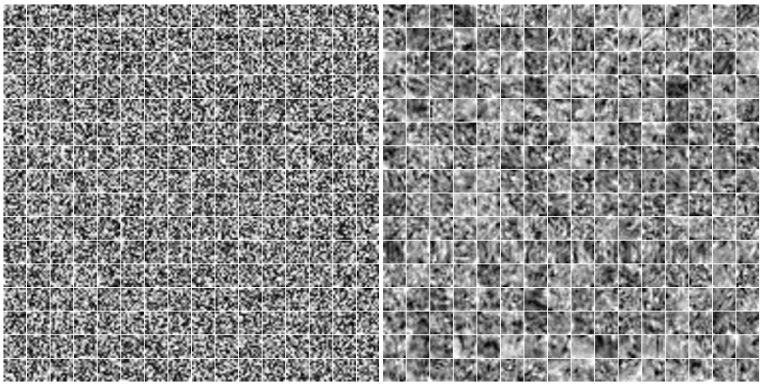
Mezi další významné soubory patří soubor s příponou `.caffemodel`, který obsahuje uložený (natrénovaný) model v podobě jeho parametrů a několika dalších popisných informací, a soubor s příponou `.solverstate` obsahující informace o stavu trénování. Ukládání těchto souborů lze specifikovat v rámci solveru.

5.2 Použité hyperparamtry trénování

První experimentální sítě byly trénovány pomocí stochastic gradient descent, což se ukázalo být nevhodným. Zhruba v polovině případů (otestováno spuštěním experimentu 100x se stejným nastavením) dochází už v několika prvních stech iterací k uváznutí modelu ve stavu, kdy už nedochází k žádným změnám a kdy je výstup sítě zcela očividně nesprávný (obrázek 5.1). Z tohoto důvodu je u všech dalších experimentů používána AdaDelta, u které už se tento problém nevyskytuje.

Na začátku trénování jsou váhy sítě náhodně inicializovány metodou xavier (obrázek 5.2). Samotné trénování probíhá 20 000 iterací s velikostí batche 32 vrstev CT skenu, při-

[4] https://developers.google.com/protocol-buffers/



Obrázek 5.2: Několik filtrů druhé vrstvy architektury Net5_BigFilters (zleva) těsně po inicializaci metodou xavier a na konci trénování.

čemž při každé pětisté iteraci dojde k vyhodnocení na testovacím datasetu. Počet testovacích iterací je pro každý dataset upraven tak, aby v každé testovací fázi prošla sítí všechna testovací data. V testovací i trénovací fázi je ukládán obrazový záznam zachycující na třech grafech (PRC, vývoj loss a vývoj AUC) stav trénování. Součástí každého záznamu jsou také tři vrstvy CT skenu, pro které je zobrazen originál vstupující do sítě, snímek s anotací a výstup sítě. V případě trénovacích dat jsou tyto tři ukázkové vrstvy vybrány náhodně, protože v každém batchi se vyskytuje jiné rozložení snímků. V testovací fázi jsou ukázkové vrstvy vždy stejné. Ukládán je záznam každé padesáté trénovací iterace a celkový záznam pro každou testovací fázi zahrnující všechny testovací iterace.

Fázové trénování U většiny experimentů bylo použito jednoduché trénování o jedné fázi. Experimenty s různými způsoby trénování byly provedeny pouze u CRF architektur, a to ve třech variantrách:

- jedna fáze s jednou loss vrstvou,
- jedna fáze s dvěma loss vrstvami,
- trojfázové trénování.

U varianty s dvěma loss vrstvami se jedna tato vrstva nachází na konci jednoduché konvoluční části a druhá na konci CRF části. Tato varianta trénování se ukázala být neefektivní. V trojfázovém trénování je nejprve samostatně natrénována první část sítě bez CRF. Váhy předtrénované části jsou zafixovány a v druhé fázi je připojena a trénována CRF část. Konečně v třetí fázi jsou trénovány obě samostatně předtrénované části společně. Trojfázový postup se ukázal být pro tento typ architektury nejlepším.

5.3 Pomocné skripty

Pro přípravu dat byla vytvořena sada několika jednoduchých skriptů v jazycích GNU bash a Python. Pomocí těchto nástrojů byly z DICOM souborů vyextrahovány jednotlivé vrstvy CT skenů, které jsou odsud dál uchovávány ve formátu černobílých PNG obrázků. Rozměry

jednotlivých použitých skenů se liší, proto byly hned na začátku ořezány na pevný rozměr 200x200 pixelů. U skenů, které mají jeden nebo oba rozměry menší než 200 pixelů, byla přidaná oblast zaplněna hodnotou 0, tedy hodnotou která odpovídá oblastem mimo tělo pacienta. Protože se v datasetu Tibia nacházel sken, kde byly na několika vrstvách viditelné obě nohy pacienta, zatímco anotovaná byla pouze jedna, byla tato část skenu odstraněna a rovněž nahrazena nulami, aby síti nebyly předkládány zavádějící informace, které by mohly vést ke zhoršení výsledků. K hromadným operacím s obrazovými daty byl použit mimo jiné nástroj ImageMagick[5] a k načítání DICOM formátu byla použita upravená verze skriptu `loadDicom.py`, jež byl vytvořen v rámci bakalářské práce Davida Hlavoně [3].

Další skripty slouží k přejmenování vrstev CT skenů tak, aby číslo vrstvy obsahovalo vždy stejný počet znaků, což je praktičtější pro řazení souborů operačním systémem, a k vytvoření textových seznamů obsahujících názvy skenů určených k trénování a testování neuronových sítí.

5.4 Skripty pro manipulaci se sítěmi

Pro trénování všech variant byl použit trénovací skript `train.py` zahrnující trénování sítě, průběžné testování na testovacích datech, zmíněné náhodné transformace a ukládání obrazových záznamů dokumentujících vývoj trénování. U některých architektur probíhalo trénování v několika fázích. V tomto případě lze opakovaně použít stejný trénovací skript, pouze s nutností specifikovat cestu k předtrénovanému modelu. Nevýhodou skriptu je, že v každé iteraci nejprve načítá požadovaná data a upravuje je, zatímco propagace dat sítí je pozastavena. Tímto je celková doba trénování výrazně prodlužována. Pro budoucí experimenty by určitě bylo vhodné data připravovat paralelně s trénováním a případně využít k uložení dat databázi s rychlým přístupem.

Druhý skript pro manipulaci se sítí slouží k aplikaci natrénovaných sítí. Skript postupně načítá data specifikovaná umístěním adresáře a seznamem souborů. Každý takto načtený soubor (vrstva CT skenu) je předložen na vstup sítě a výstupní obraz je uložen do zadané cesty. U obou skriptů jsou pro manipulaci s obrazovými daty jsou použity moduly nupmy, cv2 a modul sklearn pro vyhodnocování metrik precision, recall a AUC.

[5] https://www.imagemagick.org/script/index.php

Kapitola 6

Experimenty a vyhodnocení

Celkem bylo provedeno zhruba sedmdesát experimentů podle výše uvedeného návrhu. Ty nejvýznamější z nich jsou popsány v této kapitole včetně vyplynuvších závěrů. Experimety, jakými je třeba zkoušení různých počtů filtrů či použití dropoutu, neměly zásadní vliv na výsledek analýzy a nebudou tedy blíže rozebírány.

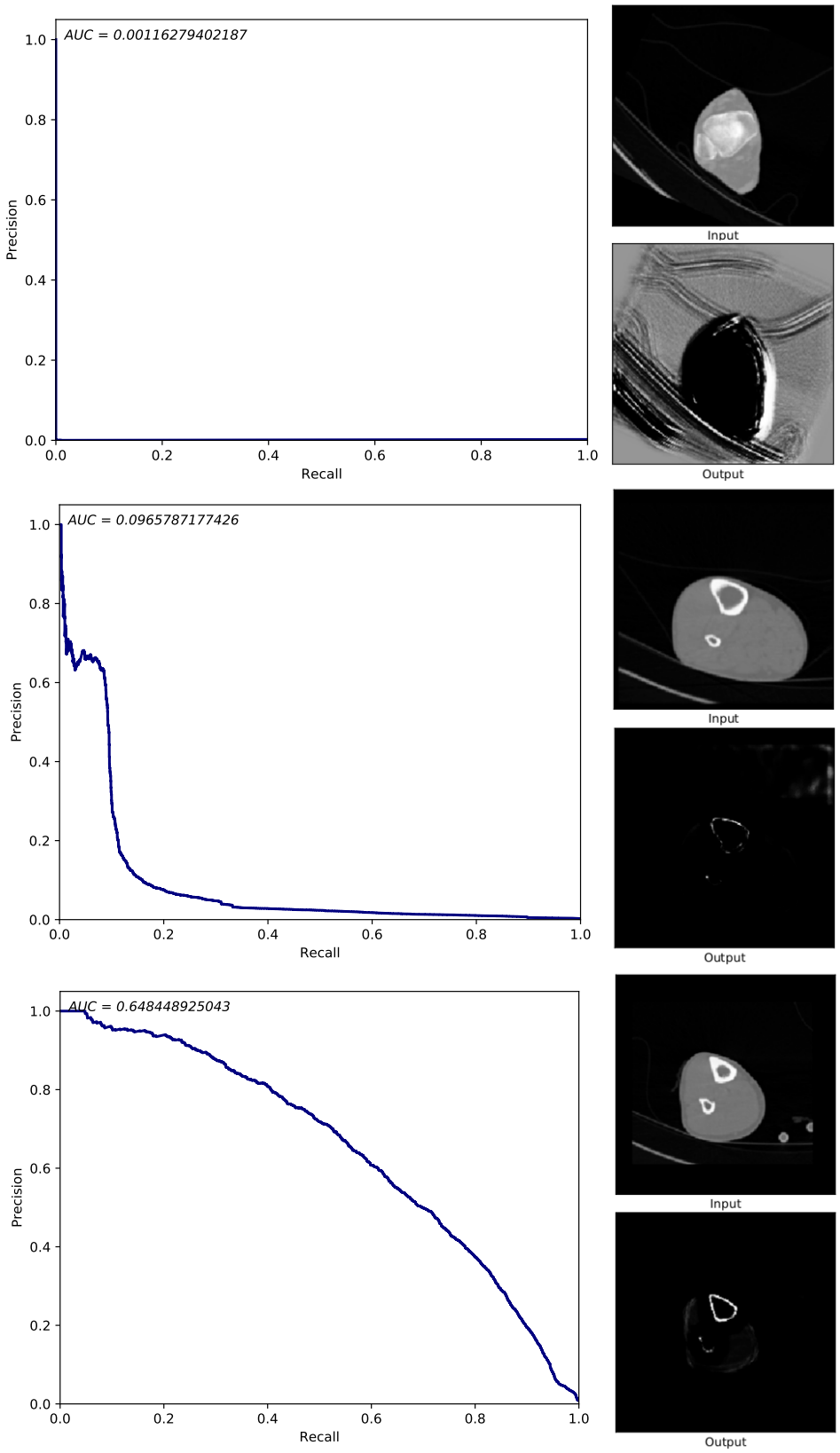
6.1 Metriky pro vyhodnocení úspěšnosti

Metrikou, která je díky podstatě hlubokého učení přítomná vždy, je loss funkce. Z loss funkcí implementovaných v Caffe frameworku lze k segmentaci použít například SoftmaxWithLoss, jejímž vstupem jsou odhady pravděpodobností příslušnosti k třídě pro každý pixel. Pro binární problémy, jakým je i klasifikace tříd kontura/pozadí, je možné použití SigmoidCrossEntropyLoss. Protože ale loss funkce není dostatečně vypovídající metrikou pro hodnocení přesnosti segmentace, byla použita ještě jedna metrika, a tou je plocha pod precision-recall křivkou (AUC).

Metrika AUC je měřena po celou dobu trénování a její vývoj společně s vývojem hodnoty loss funkce je zaznamenáván do grafů, kreté jsou průběžně ukládány. Každé trénování (případně fáze trénování u složitějších architektur trénovaných po částech) probíhá 20 000 iterací a i když už v několika posledních iteracích není patrné zlepšování AUC, tato metrika stále hodně kolísá a je tedy potřeba zvážit, jakým způsobem srovnávat naměřené metriky různých experimentů. Nakonec byly zvoleny dvě varianty původní metriky. První variantou je **b-AUC**, což je nevyšší hodnota AUC naměřená na testovacích datech v průběhu celého trénování. Tato metrika je ale jen velmi orientační, protože testovací fáze nenásleduje po každé iteraci a nejlepší nastavení paramentrů tedy nemusí být vůbec zachyceno. Statisticky větší váhu má **a-AUC**, což je průměr hodnot AUC zaznamenaných v 15 000. až 20 000. iteraci. Na základě této metriky jsou mezi sebou srovnávány jednotlivé experimety.

6.2 Datasety rozšířené transformacemi

U malého datasetu Tibia došlo použitím náhodných transformací ke zlepšení a-AUC přibližně o 5 %. Oproti tomu u datasetu Humerus, který je mnohem větší, nebylo pozorováno žádné zlepšení, v některých případech došlo ke zhoršení a-AUC o jednotky procent. Důvo-



Obrázek 6.1: Precision-recall křivky trénovacích dat zlepšující se v průběhu trénování. Iterace 0, 100 a 500.

Vícekanálový vstup a transformace				
	Tibia		Humerus	
Architektura	a-AUC	b-AUC	a-AUC	b-AUC
Baseline	63 %	65 %	74 %	75 %
3ch	62 %	67 %	73 %	74 %
5ch	64 %	68 %	76 %	77 %
Baseline trans	69 %	70 %	73 %	74 %
3ch trans	68 %	70 %	71 %	73 %
5ch trans	69 %	72 %	72 %	73 %

Tabulka 6.1: Tabulka architektur a jejich výsledků.

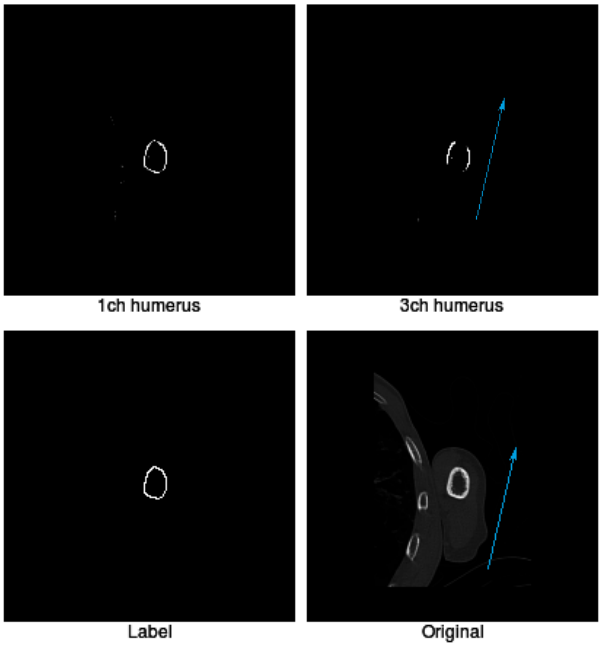
dem k tomu může být, že nepřirozené zvýšení variability dat úplně neodpovídá variabilitě přirozené a naopak vnáší do dat pro síť matoucí informace. Při transformacích typu rotace může dojít k mírnému znekvalitnění anotace. Label musí být samozřejmě orotován stejným způsobem, jako vstupní obraz. To znamená použití interpolace při výpočtu nové anotace kontury, čímž dochází k její drobné deformaci a výsledkem může být rovněž negativní vliv na proces učení. Transformace humeru byly testovány na relativně malých architekturách (Baseline, Baseline 3ch, Baseline 5ch, 5lrs), proto dalším důvodem k naměřeným výsledkům může být neschopnost takových architektur pojmout složitější problém. Do budoucna by bylo zajímavé vyzkoušet transformace Humeru na složitějších sítích.

6.3 Více vstupních kanálů

V rámci experimentů byl srovnáván efekt použití několika vrstev CT skenu coby vícekanálového vstupu konvoluční sítě. Ukázalo se, že použití více kanálů je uspěšnější v případech, kdy dochází k dostatečnému překryvu hledané kontury mezi po sobě následujícími vrstvami. Naopak v případech, kdy kost není umístěna vodorovně podél těla pacienta a dochází tak k posunům kontury na sousedících vrstvách, se vytrácejí odezvy směřující kolmo ke směru posuvu (obrázek 6.2). Různé počty kanálů mají co do metriky podobné výsledky, avšak při vizuálním hodnocení je patrné vytrácení kontur u snímků s větším posunem mezi vrstvami. Do budoucna navrhovaným zlepšením je experimentovat s použitím interpolačních technik pro vytvoření pomocných snímků mezi vrstvami s velkým posunem.

6.4 Rozšíření zorného pole

Z výsledků uvedených v tabulce 6.2 je zjevné, že větší zorné pole (resp. větší síť a tedy více parametrů k natrénování) má smysl, pokud je dataset dostatečně velký k jeho natrénování. U největších architektur BigKernels a Net5_BigKernels dosahuje konečná AUC na trénovacích datech Tibia datasetu až 99 %, zatímco a-AUC na testovacích datech zůstává kolem 60 %. Transformacemi se podařilo dosáhnout přibližně 70% přesnosti, dataset je ale stále příliš malý. Oproti tomu experimety s Net5_BigKernels na datasetu Humerus dosahují téměř 90 % a-AUC a jsou jen nepatrně horší než při použití stejné architektury s CRF3 navíc.



Obrázek 6.2: Při velkém posuvu kosti mezi následujícími vrstvami dochází u sítí s vícekanálovými vstupy (vpravo nahoře) k vytrácení kontury kolmo na směr posuvu. Směr posuvu je vyznačen šipkou.

Různá zorná pole					
		Tibia		Humerus	
Architektura	Zorné pole	a-AUC	b-AUC	a-AUC	b-AUC
Baseline	11x11	63 %	65 %	74 %	75 %
Net5	15x15	60 %	68 %	79 %	81 %
BigKernels	17x17	61 %	65 %	78 %	80 %
Net5 BigKernels	31x31	62 %	65 %	88 %	89 %

Tabulka 6.2: Tabulka výsledků různých zorných polí.

6.5 Experimenty s CRF

Jistého zlepšení oproti plně dopředným architekturám dosahuje i doplnění sítě o rekurentní část (CRF3) částečně inspirovanou conditional random fields. Bylo vyzkoušeno několik přístupů k trénování této architektury: CRF1 s jednofázovým trénováním, CRF2 s jednofázovým trénováním a dvěma loss vrstvami a CRF3 s trojfázovým trénováním. Poslední variantou je CRF4, která se liší od CRF3 pouze větším počtem CRF vrstev. Nejúspěšnější z těchto architektur je CRF3, jak je patrné z tabulky 6.3.

Do budoucna by mohlo být zajímavé vyzkoušet přesnější implementaci conditional random fields tak, aby byla dodržena sémantika jejich původního optimalizačního algoritmu a porovnat, zda oproti triviální implementaci dojde k většímu vylepšení výsledků. Conditional random fields jsou původně určená spíše pro segmentaci, kdy je souvislá segmentovaná plocha větší. Proto jejich použití pro detekci kontur, kdy je segmentovaná oblast velmi tenká, může být více limitované, než použití pro segmentaci větších ploch. Z tohoto důvodu by

CRF				
	Tibia		Humerus	
Architektura	a-AUC	b-AUC	a-AUC	b-AUC
CRF1	66 %	67 %	74 %	75 %
CRF2	62 %	65 %	70 %	71 %
CRF3	63 %	66 %	80 %	80 %

Tabulka 6.3: Tabulka CRF architektur a jejich výsledků.

Nejlepší výsledky				
	Tibia		Humerus	
Architektura	a-AUC	b-AUC	a-AUC	b-AUC
Net5 BigKernels CRF3	59 %	63 %	88 %	89 %
Net5 BigKernels CRF3 trans	73 %	75 %	x	x

Tabulka 6.4: Tabulka nejlepších výsledků

mohlo být zajímavé zapojit vícekanálový vstup i u architektur obohacených o CRF, čímž by při dostatečné návaznosti sousedících vrstev mohlo dojít ke zvětšení souvislé segmentované plochy a účinek CRF by tak mohl být více patrný.

6.6 Shrnutí výsledků

Nejlepšího výsledku experimentů na datasetu Humerus (87 % a-AUC a 89 % b-AUC) bylo dosaženo za použití nastavení BigKernels 5lrs s následovným fázovým trénováním CRF3. U datasetu Tibia bez použití transformací tako architektura dosahuje pouze 59 % a-AUC a 62 % b-AUC. Při použití transformací se výsledky této architektury u datasetu Tibia zlepší na 72 % a-AUC a 75 % b-AUC.

U obou datasetů je patrný značný rozdíl mezi výsledky konvolučních neuronových sítí a výsledky prahování s následnou aplikací sobelova operátoru. Naivní metoda sice dokáže nalézt kontury kostí, ale má vysoké odezvy i v jiných oblastech, kde se vyskytují hrany. Oproti tomu neuronové sítě mohou mít slabší odezvy v koncových oblastech kostí, zato úspěšně potlačují odezvy způsobené jinými hranami, dokonce i hranami sousedících menších kostí.

Kromě výše zmíněných experimetů bylo experimentováno ještě s encoder-decoder architekturou zmíněnou v kapitole 4. Výsledky této architektury nejsou příliš slibné. Architektura dosahuje úspěšnosti 66,32 % a-AUC a 70,73 % b-AUC pro dataset Humerus, což jsou hodnoty horší než pro základní architekturu Baseline. Ač zaměřením této práce nebylo vytvořit encoder-decoder architekturu, do budoucna by mohlo být zajímavé pokusit se lépe implementovat architekturu tohoto typu a vyzkoušet její trénování na větším nebo lépe augmentovaném datasetu.

Kapitola 7

Závěr

Hlavním cílem práce bylo experimetovat s nastavením neuronových sítí a nalézt tímto způsobem co nejlepší řešení problému detekce kontur dlouhých lidských kostí tibie a humeru. Výsledky experimentů byly porovnávány jak objektivně (pomocí zvolené metriky jíž je plocha pod Precision-Recall křivkou) tak subjektivně vizuálním hodnocením kvality segmentace. Dalšími důležitými součástmi bylo získání přehledu o současném použití konvolučních neuronových sítí (obzvlášť těch zaměřujících se na problém detekce resp. segmentace), seznámení se s Caffe frameworkem pro trénování neuronových sítí a jeho následné využití pro implementaci navrženého řešení problému a obecná práce související s přípravou trénovacích dat včetně vyzkoušení metod navyšování omezené velikosti medicínské datové sady.

Podařilo se implementovat několik architektur zaměřující se na různé aspekty řešeného problému s přihlédnutím k charakteru použitých datasetů. Bylo vypozorováno, že použití několika vstupních kanálů, pro nalezení kontury v prostředním z nich má smysl v případě, kdy dochází k dostatečným překryvům hledané kosti v sousedících vrstvách. Použito bylo několik architektur s různě velkým zorným polem, přičemž nejúspěšjěnší variantou byla architektura Net5_BigKernels se současným použitím CRF3. Tato architektura dosahuje na datasetu Humerus téměř 90% úspěšnosti. Na výrazně menším datasetu Tibia dosahuje tato architektura úspěšnosti přobližně 59 % bez transformací a kolem 73 % s použitím tří náhodných transformací.

Do budoucna by bylo vhodné hledat další možnosti rozšíření datasetu a regularizace datasetu, například pomocí batch normalization, kterou se vzhledem k neznámé chybě nepodařilo vyzkoušet. Větší dataset by umožnil experimenty s většími sítěmi typu encoder-decoder. Vhodné by bylo zejména vyzkoušet postup používaný autory SegNetu [1], protože (jak můžeme vidět na obrázku 3.6) dosahuje poměrně přesné segmentace tenkých struktur, například tyčí lamp pouličního osvětlení, což může být předpoklad pro aplikovatelnost na problém detence kontur. Vzhledem k nerovnoměrnému výskytu tříd kontura a pozadí by také mohlo být vhodné použít mechanismus class balancingu. V průběhu vzniku této práce vyšela nová verze Caffe frameworku (Caffe2), bylo by proto pro následující práci vhodné přejít na novou verzi, která slibuje větší modularitu a na první pohled se zdá být lépe dokumentovanou.

Literatura

[1] Badrinarayanan, V.; Kendall, A.; Cipolla, R.: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *CoRR*, ročník abs/1511.00561, 2015.

[2] Dosovitskiy, A.; Springenberg, J. T.; Brox, T.: Learning to Generate Chairs with Convolutional Neural Networks. *CoRR*, ročník abs/1411.5928, 2014.

[3] Hlavoň, D.: Hluboké neuronové sítě pro analýzu 3D obrazových dat. 2016, bakalářská práce, Brno, FIT VUT v Brně.

[4] Jia, Y.; Shelhamer, E.; Donahue, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. *CoRR*, ročník abs/1408.5093, 2014.

[5] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, Red Hook, N.Y: Curran Associates, Inc., 2012, s. 1097–1105.

[6] Long, J.; Shelhamer, E.; Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. *CoRR*, ročník abs/1411.4038, 2014.

[7] McCulloch, W. S.; Pitts, W.: A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, ročník 5, č. 4, 1943.

[8] Mnih, V.; Kavukcuoglu, K.; Silver, D.; aj.: Playing Atari with Deep Reinforcement Learning. *CoRR*, ročník abs/1312.5602, 2013.

[9] Noh, H.; Hong, S.; Han, B.: Learning Deconvolution Network for Semantic Segmentation. *CoRR*, ročník abs/1505.04366, 2015.

[10] Pfister, T.; Charles, J.; Zisserman, A.: Flowing ConvNets for Human Pose Estimation in Videos. *CoRR*, ročník abs/1506.02897, 2015.

[11] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, ročník abs/1409.1556, 2014.

[12] van der Walt, S.; Schönberger, J. L.; Nunez-Iglesias, J.; aj.: scikit-image: image processing in Python. *PeerJ*, 6 2014.

[13] Yang, J.; Price, B. L.; Cohen, S.; aj.: Object Contour Detection with a Fully Convolutional Encoder-Decoder Network. *CoRR*, ročník abs/1603.04530, 2016.

[14] Zeiler, M. D.: ADADELTA: An Adaptive Learning Rate Method. *CoRR*, ročník abs/1212.5701, 2012.

[15] Zheng, S.; Jayasumana, S.; Romera-Paredes, B.; aj.: Conditional Random Fields as Recurrent Neural Networks. *CoRR*, ročník abs/1502.03240, 2015.

[16] Šrámek, J.; Ráček, O.; Sedlář, M.; aj.: Získávání a analýza obrazové informace. [online], 2011, učební text LF MU.
URL http://www.med.muni.cz/biofyz/Image/ucebnice.pdf

Přílohy

Příloha A

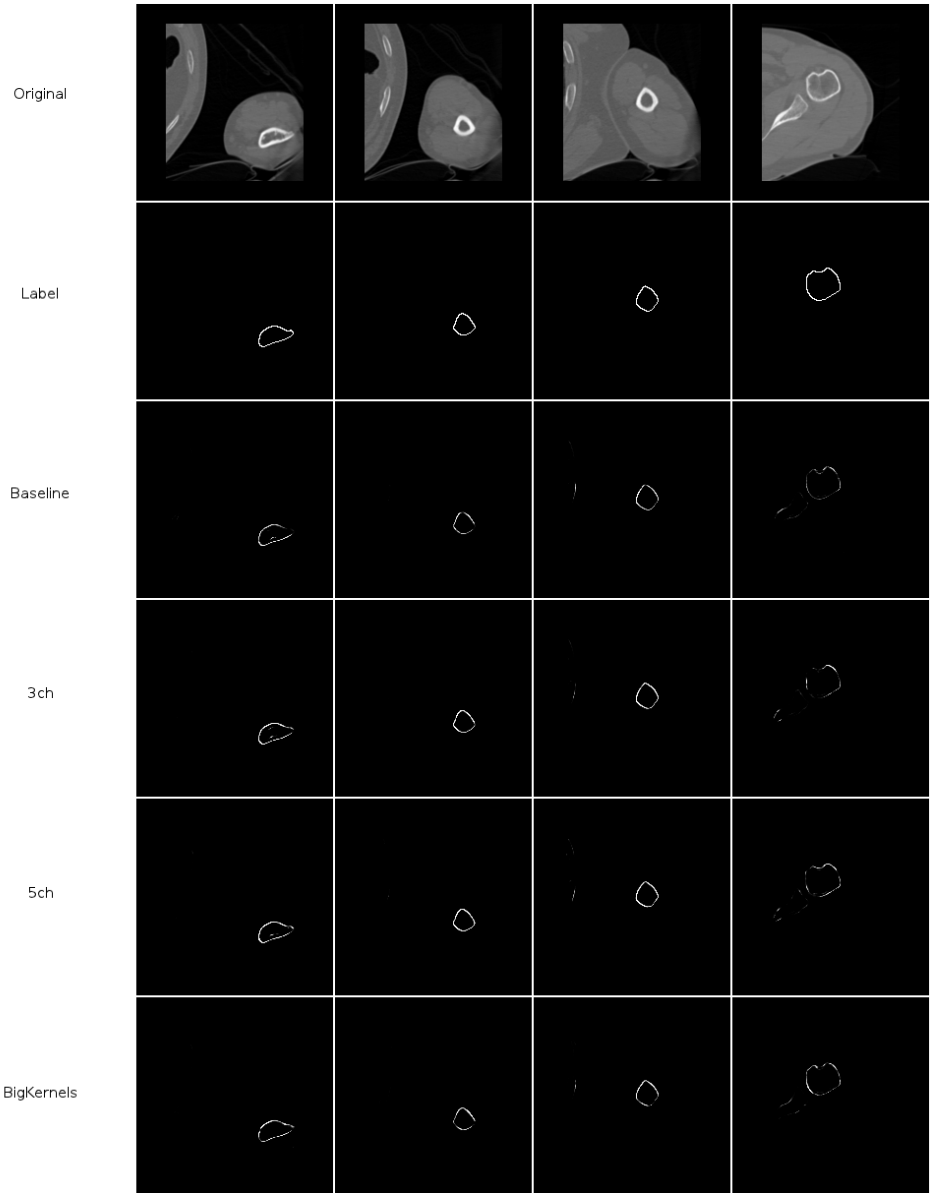
Tabulka architektur

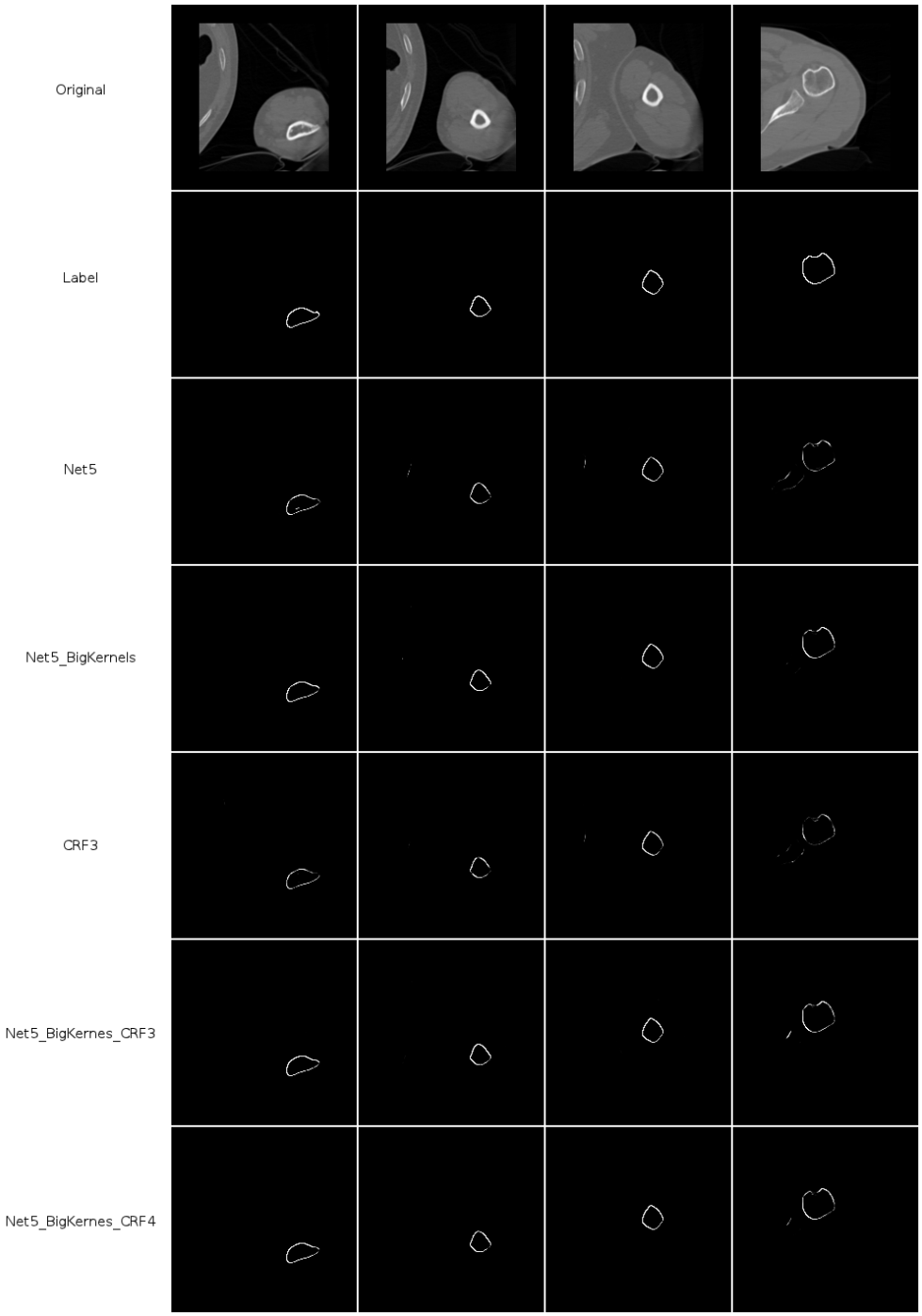
Přehled konvolučních architektur							
Tibia		**Humerus**		**Tibia**		**Humerus**	
a-AUC	b-AUC	a-AUC	b-AUC	a-AUC	b-AUC	a-AUC	b-AUC
Baseline				**3ch**			
data 32x1x200x200 conv1 5x5; 32 + relu conv2 5x5; 32 + relu conv3 3x3; 1 + sigmoid				data 32x**3**x200x200 conv1 5x5; 32 + relu conv2 5x5; 32 + relu conv3 3x3; 1 + sigmoid			
63.26 %	65.24 %	73.58 %	74.96 %	62.45 %	66.83 %	72.83 %	73.75 %
5ch				**BigKernels**			
data 32x**5**x200x200 conv1 5x5; 32 + relu conv2 5x5; 32 + relu conv3 3x3; 1 + sigmoid				data 32x1x200x200 conv1 **11x11**; 32 + relu conv2 5x5; 32 + relu conv3 3x3; 1 + sigmoid			
64.27 %	67.74 %	76.03 %	77.12 %	61.20 %	65.13 %	78.48 %	79.61 %
Net5				**Net5_BigKernels**			
data 32x1x200x200 conv1 5x5; 32 + relu conv2 5x5; 32 + relu conv3 3x3; 32 + relu conv4 3x3; 32 + relu conv5 3x3; 1 + sigmoid				data 32x1x200x200 conv1 **11x11**; 32 + relu conv1 **11x11**; 32 + relu conv2 **5x5**; 32 + relu conv2 **5x5**; 32 + relu conv1 3x3; 1 + sigmoid			
60.00 %	68.43 %	79.12 %	80.60 %	61.56 %	64.85 %	87.77 %	88.88 %

Přehled CRF architektur			
CRF3			
		data 32x1x200x200	
data →		conv1 5x5; 32 + relu	
conv1 →		conv2 5x5; 32 + relu	
conv2 →		conv3 3x3; 1 + softmax	
softmax + conv3 + data →		crf1 5x5; 1	
crf1 + conv3 + data →		crf2 5x5; 1	
crf2 + conv3 + data →		crf3 5x5; 1 + Sigmoid	
Tibia		**Humerus**	
63.11 % a-AUC	65.59 % b-AUC	79.55 % a-AUC	80.26 % b-AUC
Net5_BigKernels_CRF3			
		data 32x1x200x200	
data →		conv1 11x11; 32 + relu	
conv1 →		conv2 11x11; 32 + relu	
conv2 →		conv3 5x5; 32 + relu	
conv3 →		conv4 5x5; 32 + relu	
conv4 →		conv5 3x3; 1 + softmax	
softmax + conv5 + data →		crf1 5x5; 1	
crf1 + conv5 + data →		crf2 5x5; 1	
crf2 + conv5 + data →		crf3 5x5; 1 + Sigmoid	
Tibia		**Humerus**	
58.79 % a-AUC	62.77 % b-AUC	87.78 % a-AUC	89.35 % b-AUC
Net5_BigKernels_CRF4			
		data 32x1x200x200	
data →		conv1 11x11; 32 + relu	
conv1 →		conv2 11x11; 32 + relu	
conv2 →		conv3 5x5; 32 + relu	
conv3 →		conv4 5x5; 32 + relu	
conv4 →		conv5 3x3; 1 + softmax	
softmax + conv5 + data →		crf1 5x5; 1	
crf1 + conv5 + data →		crf2 5x5; 1	
crf2 + conv5 + data →		crf3 5x5; 1	
crf3 + conv5 + data →		crf4 5x5; 1	
crf4 + conv5 + data →		crf5 5x5; 1 + Sigmoid	
Tibia		**Humerus**	
x	x	87.64 % a-AUC	89.07 % b-AUC

Příloha B

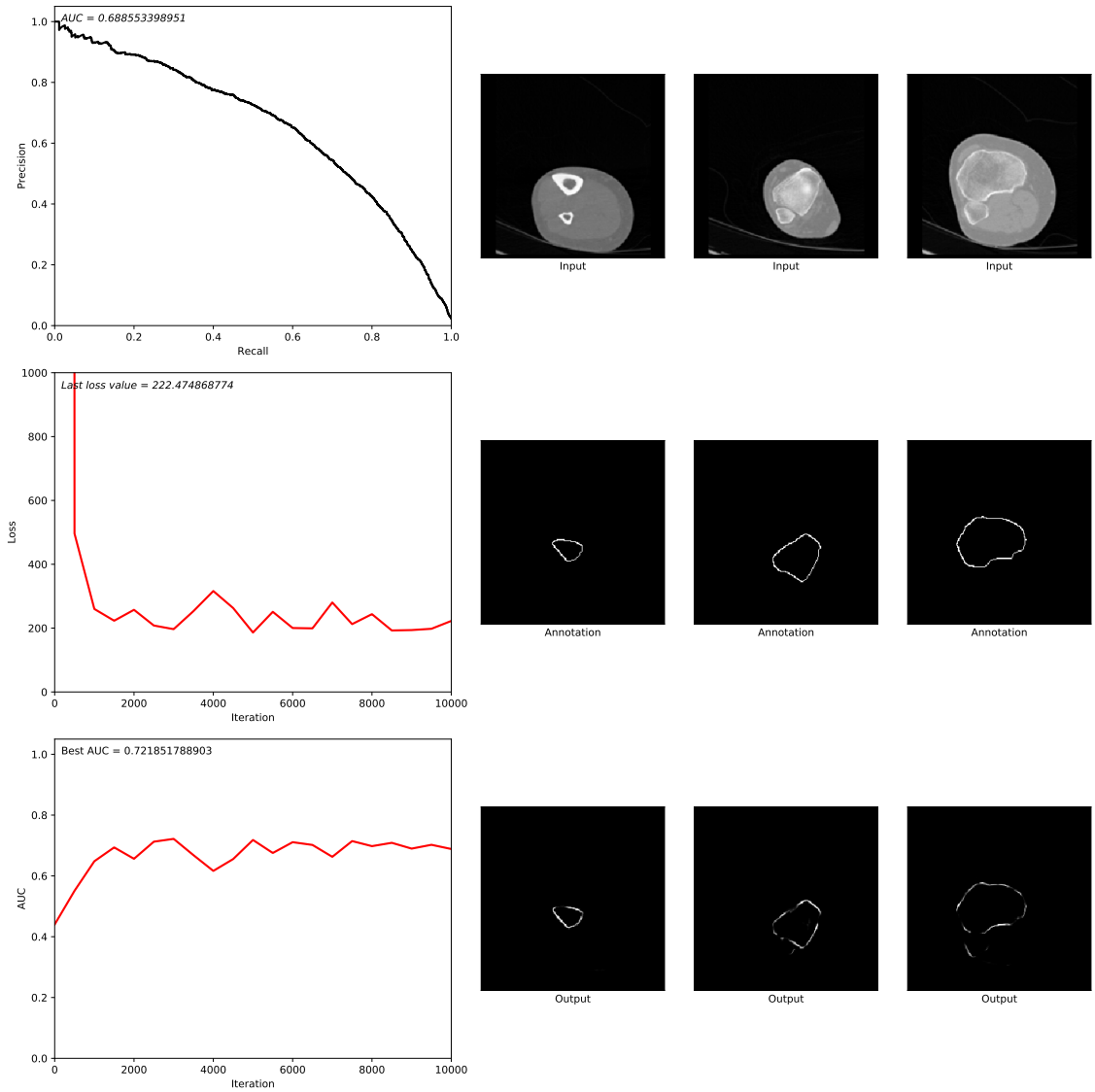
Výstupy různých architektur




Original
Label
Net5
Net5_BigKernels
CRF3
Net5_BigKernes_CRF3
Net5_BigKernes_CRF4

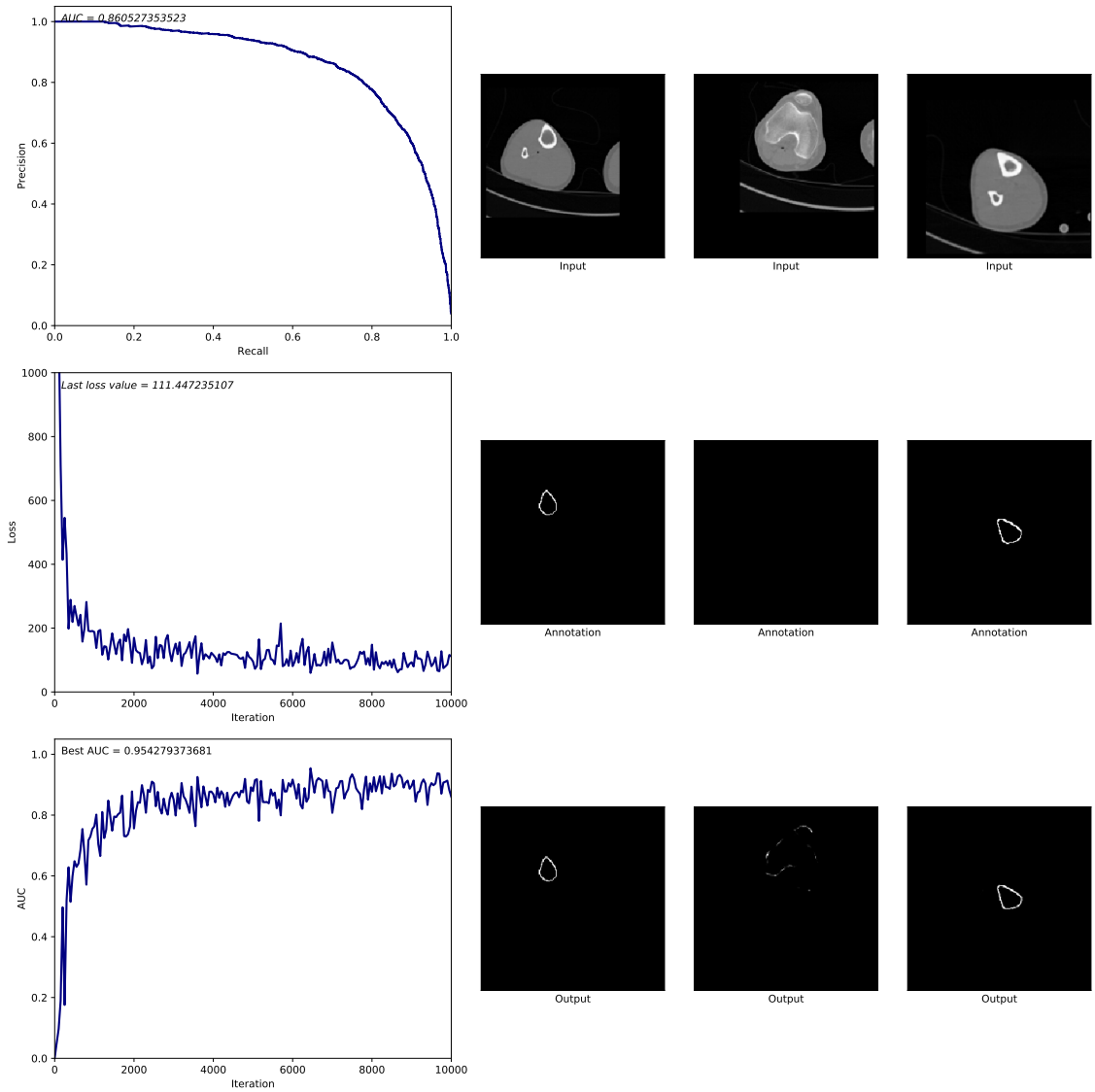
Příloha C

Záznam testování



Příloha D

Záznam trénování



Příloha E

Pseudokód trénovacího skriptu

```
LoadSolver()
LoadPretrainedModel()          # optional
LoadDataLists()

for(step in train_iter)
{
    if(step % 500 == 0)
    {
        # testing #
        for(test_step in test_iter)
        {
            LoadData()
            Testnet.Forward()
        }
        SaveRecord()
    }

    # training #
    LoadData()
    RandomTransform()          # optional
    Solver.Step(1)
    if(step % 50 == 0)
    {
        SaveRecord()
    }
}
PrintMetrics()
SaveModel()
```

Příloha F

Ukázková definice vrstvy pro Caffe

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
 param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 32
    kernel_size: 5
    stride: 1
    pad: 2
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

Příloha G

Plakát