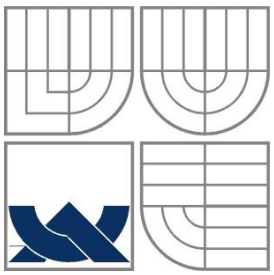
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

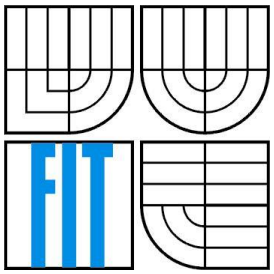
Fakulta informačních technologií
Faculty of Information Technology

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

Brno, 2016 David Hlavoň



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HLUBOKÉ NEURONOVÉ SÍTĚ PRO ANALÝZU 3D OBRAZOVÝCH DAT

DEEP LEARNING FOR 3D IMAGE ANALYSIS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

David Hlavoň

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Michal Španěl, Ph.D.

BRNO 2016

Abstrakt

Práce pojednává o využití plně konvolučních neuronových sítí pro segmentaci kostí z CT snímků. Typickým problémem při trénování na medicinských datech bývá omezená velikost trénovací sady. Experimenty ukázaly, že trénování na podoobrazech při omezeném počtu trénovacích dat dává lepší výsledky. Při trénování na podobrazech bylo dosaženo přesnosti segmentace 95,1%, což je o 30% více než při trénování na celých obrazech. Pro měření úspěšnosti segmentace byla zvolena metrika F-measure. Pro práci s konvolučními neuronovými sítěmi byl použit BVLC Caffe Framework.

Abstract

This work deals with usage of fully convolutional neural network for segmentation of bones in CT scans. Typical issue is limited size of dataset while training on medical images. Experiments show that training on patches gives score of segmentation 95,1%. Training on whole images gives score 30% less than training on patches. As metric F-measure was used. BVLC Caffe Framework was used for training neural network.

Klíčová slova

plně konvoluční neuronová síť, segmentace, podobrazy, omezený dataset

Keywords

fully convolution neural network, segmentation, patches, limited dataset

Citace

HLAVOŇ David: Hluboké neuronové sítě pro analýzu 3D obrazových dat, bakalářská práce, Brno, FIT VUT v Brně, 2016

Hluboké neuronové sítě pro analýzu 3D obrazových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením
Ing. Michala Španěla, Ph.D.
Další informace mi poskytl Ing. Michal Hradiš, Ph.D.
Data pro trénování mi poskytla společnost 3Dim Laboratory, s.r.o.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
David Hlavoň
15. 5. 2016

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Ing. Michalovi Španělovi, Ph.D. za kvalitní vedení během tvorby mé bakalářské práce a pomoc při tvorbě příspěvku na konferenci Excel@FIT pojednávající o tématu mé bakalářské práce. Dále bych chtěl poděkovat Ing. Michalovi Hradišovi, Ph.D. za konzultace při řešení problémů. Poděkování také patří společnosti 3Dim Laboratory, s.r.o., která mi poskytla data pro trénování.

Obsah

Seznam obrázků

1 Úvod

Využití neuronových sítí je v současné době na vzestupu díky rostoucímu výkonu počítačů. Tyto sítě lze využívat k různorodým činnostem, které napodobují činnost lidského mozku. Jedná se o zpracování obrazových dat, komprimace dat, rozpoznávání vzorů v audio datech a jiné. Z mého pohledu jsou neuronové sítě velice zajímavé téma a skrývají velký potenciál, avšak zatím stále brzděný výkonností počítačů.

Tato práce řeší problém segmentace 3D obrazových dat. Řešení toho problému naivním přístupem s využitím běžných programovacích technik je téměř nereálné. Neuronové sítě se pro tento typ problému naopak hodí, jelikož se dokáží z předložených vzorů naučit, jak objekt v obraze najít. Neuronové sítě se pro segmentaci obrazu používají běžně a relativně dlouhou dobu. V posledních letech se dostávají do povědomí metody založené na konvolučních neuronových sítích, které dosahují při segmentaci obrazových dat výborné výsledky, proto byla zvolena i v této práci konvoluční neuronová síť pro segmentaci 3D obrazových dat.

Jako zástupce 3D obrazových dat byly zvoleny CT snímky, kde úkolem konvoluční sítě je nalézt a označit holenní kost. Segmentace CT snímků prováděná lékaři v praxi je využití prahu Hounsfieldových jednotek [30]. Zařízení produkující CT skeny automaticky ukládá data v Hounsfieldových jednotkách. Problém je, že různé CT skenerery produkují snímky rozdílné kvality s různým šumem a také rozdílnou interpretací Housfieldových jednotek. Není tedy přesně definován práh, kdy proběhne segmentace kostí. Konvoluční neuronová síť se dokáže naučit segmentovat kosti v CT skenech s rozdílnou kvalitou, šumem a rozdílnou interpretací Hounsfieldových jednotek.

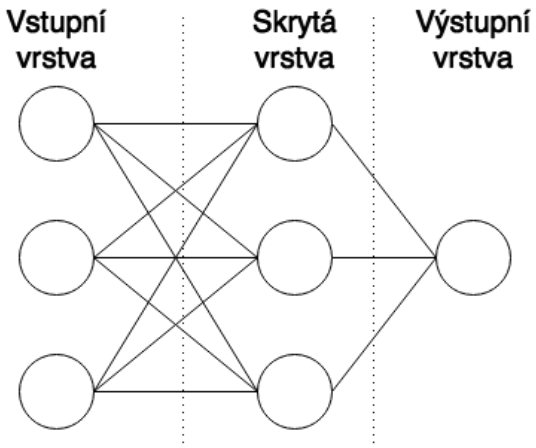
CT snímky patří do kategorie medicinských dat a s tím vyvstává problém s dostupností těchto snímků. Je komplikované získat dostatečné množství obrazů od různých pacientů z právních důvodů, jelikož CT snímky jakožto medicinská data jsou považována za důvěrné informace a dostupné jsou pouze CT snímky pacientů, kteří dali svolení pro jejich zveřejnění. Tímto vzniká problém omezené trénovací sady.

Cílem práce je segmentovat holenní kost v CT snímku s využitím plně konvoluční neuronové sítě a experimentovat s trénováním na podobrazech, čímž dojde ke zvětšení trénovací sady, a na celých obrazech. Pro práci s neuronovou sítí byl použit BVLC Caffe Framework.

Jelikož práce není encyklopedickým přehledem, tak je v práci shrnuta pouze teorie neuronových sítí, která je potřebná pro pochopení segmentace obrazu. Dále jsou uvedeny různé přístupy ostatních řešitelů při segmentaci obrazu. Déle je uveden návrh řešení a popis datové sady. K závěru práce je uvedena implementace řešení a na konec jsou uvedeny experimenty a dosažené výsledky. Tyto výsledky ukazují, že trénování na podobrazech při omezené datové sadě je efektivnější a dává lepší výsledky o 30% oproti trénování na celých obrazech a o 50% lepší než při aplikování prahu Hounsfieldových jednotek běžně používanou metodou lékaři v praxi.

2 Neuronové sítě

Cílem neuronových sítí je napodobení funkce lidského mozku. Neuronové sítě se skládají z jednotlivých umělých neuronů stejně jako biologické neuronové sítě v mozku. Model umělého neuronu je ve značné podobnosti s biologickým neuronem, viz Obrázek 2, kdy každý neuron má vstupy s danou váhou, aktivační funkci, která obecně aktivuje neuron, pokud váhový součet vstupů dosáhne určitého prahu, a výstup neuronu, který slouží jako vstup do dalšího neuronu. Jednotlivé neurony u umělých neuronových sítí jsou organizovány do vrstev, kde první vrstva se nazývá vstupní a poslední výstupní. Vrstvy mezi vstupní a výstupní se nazývají skryté. Schéma vrstvového modelu neuronové sítě ukazuje Obrázek 1.



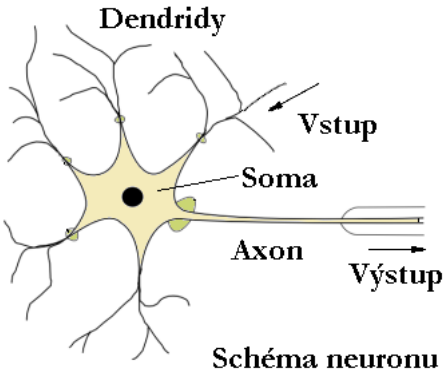
Obrázek 1 – vrstvy neuronové sítě.

Obecně lze říci, že existují dva základní typy neuronových sítí z hlediska použitých vrstev, plně propojené sítě, kde výstup každého neuronu slouží jako vstup každého neuronu v další vrstvě, viz Obrázek 1 a konvoluční sítě, které jsou blíže rozebrány v kapitole 2.3.

Lze říci, že úkolem neuronové sítě je aproximace obecně neznámé funkce *f(N)* o *N* vstupech a *M* výstupech, kde $N > M$. Neuronové sítě se nehodí na úlohy početního charakteru, avšak vynikají v úlohách typu klasifikace, rozpoznávání vzorů, segmentace, interpolace, komprese a jiné. Na různé typy úloh lze použít jiný typ neuronové sítě. Nejlepší neuronové sítě dosahují lepších výsledků než člověk, například úlohy klasifikace ručně psaných číslic [1] či rozpoznání osoby podle chůze [2]. Neuronové sítě jsou velice náročné na výpočetní výkon a paměť. Výhodou je, že výpočty pro neuronové sítě lze vysoce paralelizovat a proto se v dnešní době využívají především grafické karty pro jejich trénování. I přes vysokou výkonnost dnešních (2016) grafických karet, je problém v rychlosti a především v paměťové náročnosti při trénování rozsáhlých neuronových sítí.

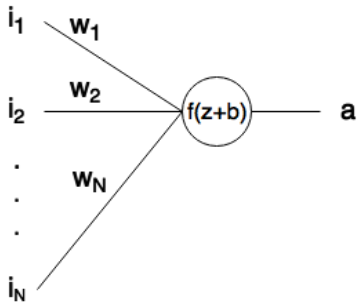
2.1 Umělý neuron

Umělý neuron je inspirován biologickým neuronem, který má vstupy dendrity násobené váhou, tělo neuronu neboli soma a axon, což je výstup neuronu.



Obrázek 2 – schéma biologického neuronu, převzato z [7].

Umělý neuron je složen ze vstupů, ke kterým jsou vázány váhy, dále z aktivační funkce a výstupu.



Obrázek 3 – schéma umělého neuronu.

Obrázek 3 zobrazuje schéma umělého neuronu, kde vstupy jsou označeny písmenem *i*. Obecně lze předpokládat *N* vstupů, kde *N* je přirozené číslo. Váhy jsou označeny písmenem *w* a jedná se o reálná čísla. Aktivační funkce je označena jako *f(z),* kde

$$z = \sum_{j=0}^{N} w_j * i_j + b \qquad (1)$$

a *b* je označován jako bias a definuje, jak snadno lze aktivovat neuron. Výstup neuronu je označen jako *a*, který je roven hodnotě aktivační funkce.

Váhy

Každý vstup do neuronu je opatřen váhou, která násobí vstupní hodnotu. Váhy [3] tedy přikládají určitý význam vstupům, čím vyšší váha tím je více významný vstup pro neuron. U konvolučních sítí jsou váhy sdíleny pro celou vrstvu a nazývají se jádro. Tyto koeficienty se mění při trénování sítě, tak aby síť lépe aproximovala obecně neznámou funkci.

Aktivační funkce

Tato funkce na základě váhového součtu vstupů a biasu aktivuje neuron. Aktivační funkce [3] může být jednoduchá skoková funkce, kdy se neuron aktivuje, pokud váhový součet dosáhne určitého prahu. Takovou aktivační funkcí lze dosáhnout pouze lineární aproximace, avšak při použití nelineární aktivační funkce a minimálně dvou vrstev neuronů lze dosáhnout univerzální aproximace [10]. Nelineární aktivační funkce mohou normalizovat výstup neuronu na intervalu <0;1>, což může být například sigmoida.

$$y = \frac{1}{e^{-x} + 1} \qquad (2)$$

Další využívanou nelineární aktivační funkcí je hyperbolický tangens, který normalizuje výstup neuronu na interval <-1;1>.

Pro segmentace obrazových dat se často používá aktivační funkce ReLU

$$y = max(0, z); z = \sum_{j=0}^{N} w_j * i_j + b \qquad (3)$$

kde *z* je váhový součet vstupů. ReLU dává pouze kladné nebo nulové aktivace neuronu.

Bias

Bias [3] je číslo, která se přičítá k váhovému součtu vstupů. Lze říci, že bias definuje jak snadné je překonat práh pro aktivaci neuronu. Úkolem tohoto koeficientu je také posunout průběh aktivační funkce po ose x a tím zlepšit přesnost sítě při aproximaci obecně neznámé funkce. Bias se společně s váhami mění při trénování. Každý neuron má vlastní bias, ale například u konvolučních sítí má celá vrstva stejný bias.

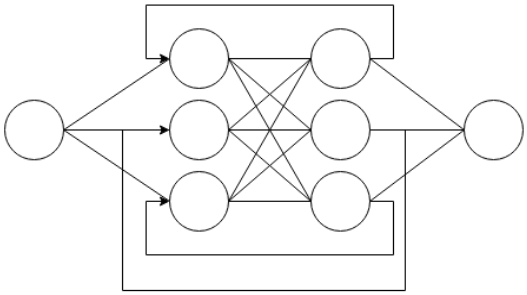
2.2 Architektury neuronových sítí

Existuje celá řada [4] architektur neuronových sítí, z nichž nejpoužívanější jsou dopředné neuronové sítě, které byly jako první navrženy a jsou také nejjednodušší. Další význačným typem jsou rekurentní neuronové sítě. Obě zmíněné architektury se znázorňují jako vrstvy neuronů.

Schéma dopředných neuronových sítí je tvořeno vrstvami, kde výstup neuronu z dané vrstvy může být spojen pouze se vstupem neuronů v následujících vrstvách, tedy v konstrukci sítě nevznikají

smyčky. Vyhodnocení dopředné neuronové sítě se provádí postupným výpočtem výstupů neuronů v rámci jednotlivých vrstev. Tedy v případě plně propojených vrstev každý neuron spočítá váhový součet vstupů (výstupů všech neuronů z předchozí vrstvy) a na výstup předá vyhodnocení aktivační funkce. Výsledkem je výstup z poslední výstupní vrstvy sítě. Tyto typy sítí se trénují s využitím algoritmu back propagation a gradientních metod jako gradient descendant, více v kapitole 2.4. Zástupcem dopředných sítí jsou i konvoluční neuronové sítě. Ukázkou dopředné neuronové sítě s plně propojenými vrstvami zobrazuje Obrázek 1.

Schéma rekurentní neuronové sítě je stejně jako schéma dopředné neuronové sítě tvořeno vrstvami neuronů, avšak výstupy neuronů z dané vrstvy mohou sloužit jako vstup neuronů v předchozích vrstvách, tedy v konstrukci sítě mohou vzniknout smyčky. Vyhodnocení rekurentních neuronových sítí se provádí s opožděným vyhodnocením neuronů [9]. Trénování rekurentních sítí se provádí s využitím upraveného algoritmu back propagation. Tato architektura lze využít pro zpracování lidské řeči [8]. Schéma rekurentní neuronové sítě zobrazuje Obrázek 4.



Obrázek 4 – schéma rekurentní neuronové sítě.

2.3 Typy neuronových sítí

Z hlediska použitých vrstev lze neuronové sítě rozdělit na plně propojené neuronové sítě a konvoluční neuronové sítě. Speciálním případem konvolučních neuronových sítí jsou plně konvoluční neuronové sítě, které nemají ve své konfiguraci žádnou plně propojenou vrstvu.

Plně propojené neuronové sítě

Tyto sítě fungují na principech popsaných výše. Při segmentaci obrazu se plně propojená síť učí přesně dané pozice v obrazech. Tyto sítě lze použít pro segmentaci obrazů, kde rozložení obrazu je vždy podobné. Jako příklad lze uvést rozpoznávání tváří. Nevýhodou těchto sítí je vysoký počet váhových parametrů a s tím spojená vysoká paměťová náročnost. Dále při vysokém počtu váhových parametrů dochází ke zvýšení složitosti procesu trénování sítě. Další nevýhodou těchto sítí je nutnost upravovat obrazy v trénovací sadě a také obrazy, které vstupují do natrénované sítě, tak aby hledaný objekt v obraze byl vždy na stejném místě, například ve středu obrazu.

Konvoluční neuronové sítě

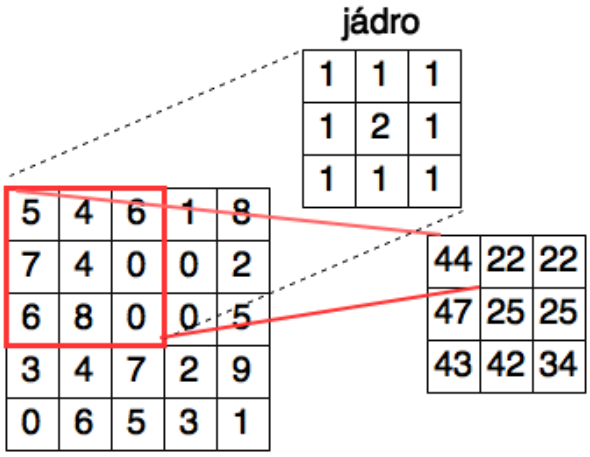
Tyto sítě se běžně využívají pro zpracování obrazových dat a fungují na principu konvoluce a sdílení váhových koeficientů v rámci matic označovaných jako konvoluční jádra. Celá konvoluční vrstva je označována jako konvoluční operátor a definují se u něho vlastnosti:

- velikost kroku konvoluce
- velikost jádra
- velikost rámce nul
- počet výstupů.

Konvoluce používaná konvolučním operátorem je definována jako váhový součet přes část obrazu odpovídajícímu velikosti konvolučního jádra, což je oblast, kterou daný neuron zpracovává. Velikost konvolučního jádra je doporučeno volit jako liché číslo, aby měla síť přesně daný středový bod konvolučního jádra. Krok určuje posun konvolučního jádra po obraze. Rámec nul vytvoří kolem obrazu vstupujícího do konvolučního operátoru okraj s nulovými hodnotami, což může sloužit pro nastavení výstupní velikosti konvolučního operátoru. Počet výstupů definuje počet konvolučních jader, kdy každé jádro může zkoumat jiný typ informace v obraze, například jedno jádro zvýrazní hrany a druhé pouze vodorovné hrany. Výstupem konvolučního operátoru je pod vzorkovaný obraz o rozměrech HxW, kde

$$H = (H_i - K + 2P)/S + 1\,, W = (W_i - K + 2P)/S + 1 \quad (4)$$

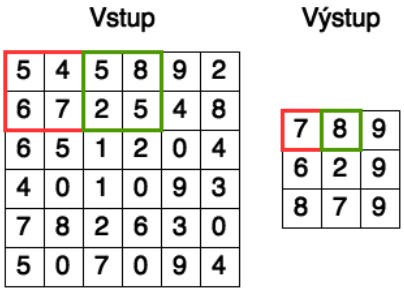
H_i respektive W_i jsou výška a šířka vstupního obrazu, K je velikost konvolučního jádra, P je velikost rámce nul a S je velikost kroku konvoluce. V konvoluční neuronové síti je výstup konvolučního operátoru společně se sdíleným biasem vstupem aktivační funkce stejně jako v případě plně propojených vrstev. V rámci konvolučního operátoru je definovaný jeden sdílený bias.



Obrázek 5 – konvoluce s velikostí jádra 3x3, velikostí kroku 1 a velikosti rámce nul roven 0.

Konvoluční operátor zajišťuje učení sítě nezávisle na pozici a transformaci objektu v obraze. Dále díky sdíleným vahám v rámci jader je možné používat větší rozměry obrazů než v případě plně propojených sítí.

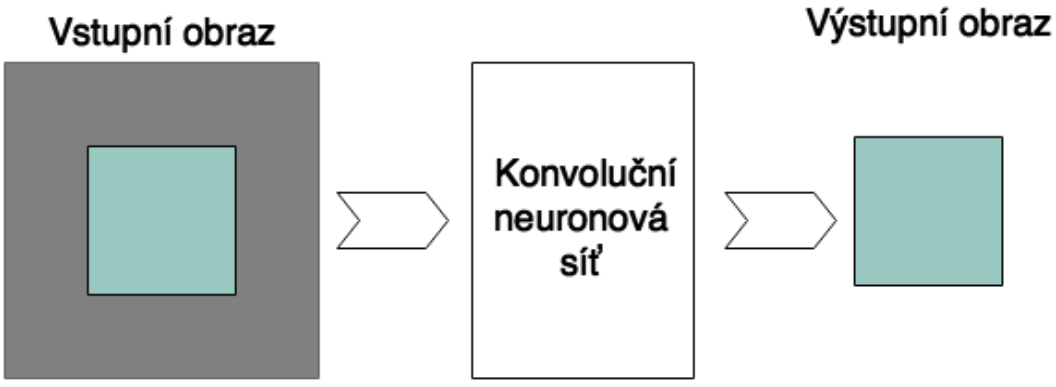
Další využívaný princip u konvolučních sítí je pooling vrstva, která zajišťují nelinearitu procesu. Základním principem pooling vrstvy je pod vzorkování obrazu pomocí nepřekrývajících se jader. Nejpoužívanější metoda pro tuto vrstvu je maximum, kdy v rámci prostoru vytyčeného rozměry jádra se na výstup propaguje maximum z hodnot. Lze použít i funkci, která vrací průměr z hodnot vytyčených velikostí jádra. Nejběžnější velikost jádra je 2 s krokem posunu 2, což zajistí zmenšení obrazu na polovinu.



Obrázek 6 – aplikace maximum pooling vrstvy s rozměry jádra 2x2 a krokem 2.

Celá konvoluční síť je tvořena několika konvolučními vrstvami a několika pooling vrstvami. Výstupy konvolučních vrstev jsou vždy vstupem do aktivační funkce. Pooling vrstva následuje vždy až za konvoluční vrstvou, avšak pooling vrstva může být vynechána a může následovat další konvoluční vrstva. Za těmito vrstvami následuje plně propojená vrstva, která má význam klasifikátoru. V případě plně konvolučních sítí se plně propojená vrstva již neuplatňuje a výstupem neuronové sítě je pod vzorkovaný vstupní obraz. Obecně se velikosti jader konvolučních vrstev s rostoucí hloubkou neuronové sítě zmenšují a počet výstupů se zvětšuje.

V případě konvolučních sítí je oblast vstupního obrazu, kterou neuronová sít klasifikuje rovna velikosti výstupu neuronové sítě, viz Obrázek 7, tedy vše co je mimo tuto oblast, neuronová síť nebere v potaz.



Obrázek 7 – demonstrace pracovní oblasti, kterou neuronová sít bere v potaz. Šedá oblast definuje oblast, která není zahrnuta do vyhodnocení konvoluční neuronovou sítí. Zelená oblast definuje oblast, která je zpracována konvoluční neuronovou sítí.

2.4 Učení neuronových sítí

Učení neboli trénování je nedílnou součástí neuronových sítí. Existují tři základní metody učení:

- učení s učitelem,
- učení bez učitele [11],
- posilované učení [12].

Obecně pro proces učení neuronových sítí je nutné nalézt určité hyper parametry jako:

- koeficient učení,
- počty skrytých vrstev,
- počty neuronů ve vrstvách

a jiné, které zajistí, že síť bude dosahovat dobrých výsledků. Důležitým hyper parametrem je i počet epoch učení, což znamená, kolikrát sítí při procesu učení projde celá trénovací sada. Počet epoch ovlivňuje úroveň natrénování sítě, ale pokud je tento parametr zvolen příliš vysoký, může dojít k přetrénování neuronové sítě a tím ztratí síť schopnost generalizace. Pro snížení nebezpečí přetrénování sítě existují různé optimalizační techniky, více viz kapitola 2.5.

Učení s učitelem

Tato metoda učení spočívá ve vytvoření trénovací sady, kdy každý prvek v trénovací i testovací sadě tvoří dvojici data a klasifikace. Testovací sada se využívá pro vyhodnocení úspešnosti sítě. Tato sada je disjunktní od trénovací, která se využívá pro učení sítě a obsahuje mnohem více vzorků. Data se vkládají na vstup neuronové sítě a klasifikace představuje cílený výstup neuronové sítě pro daný vstup. Klasifikace může být v podobě skaláru, kdy se jedná o klasifikaci do 1 z n tříd, nebo v podobě vektoru, kdy jde buď o klasifikaci do více tříd, nebo jde o segmentaci. V případě segmentace obrazu obsahuje klasifikace třídu každého pixelu, do které má být pixel zařazen. Trénovací sada se připravuje na základě znalosti vstupních dat a velikosti výstupu neuronové sítě. V rámci celé trénovací sady by mělo být rozložení všech tříd klasifikace dle normálního rozdělení pravděpodobnosti, aby se proces učení zefektivnil.

Cenová funkce

Během učení se počítá chyba, která je dána cenovou funkcí. Různé cenové funkce lze použít pro odlišné cíle jako segmentace, klasifikace a jiné. Mezi nejběžněji používané cenové funkce patří:

- Softmax [13], která se využívá pro klasifikace do 1 z n tříd nebo pro segmentaci obrazu
- Cross Entropy a Kvadratická cenová funkce [14], kdy obě tyto funkce se využívají při segmentaci.

Na základě výstupu neuronové sítě se počítá hodnota cenové funkce. Učení neuronových sítí spočívá v nalezení minima této cenové úpravou vah a biasů neuronové sítě.

Nejpoužívanější metodou pro hledání minima cenové funkce je Gradient descendant, který počítá gradientní vektor složený z parciálních derivací cenové funkce podle jednotlivých vah a další gradientní vektor pro biasy. Lze tedy říct, že gradientní vektor udává jakoby směr, kde cenová funkce nejvíce klesá, při degradaci problému do 3D prostoru. Tento gradient se počítá pro každý trénovací vzorek a výsledný gradientní vektor je aritmetickým průměrem gradientních vektorů jednotlivých trénovacích vzorků. Na základě výsledného gradientu se vypočítají nové hodnoty vah a biasů. Ve vtahu pro výpočet nové hodnoty váhy a biasu vystupuje konstanta nazvaná jako koeficient učení, který udává jak velký krok udělat ve směru, který definuje gradientní vektor. Pokud se zvolí koeficient učení příliš velký, řešení hledání minima bude oscilovat kolem minima, ale nikdy ho nedosáhne. Pokud se zvolí koeficient učení příliš malý, tak proces učení bude velice pomalý nebo při strojovém učení se může stát, že se síť přestane učit z důvodů omezení výpočtů v plovoucí desetinné čárce. Koeficient učení se musí volit také z hlediska hloubky neuronové sítě. Neuronové sítě se nejčastěji trénují na velikém počtu vzorků, což by v důsledku použití gradient descendant vedlo ke zpomalení a také k opravdu velké paměťové náročnosti procesu učení. Proto se uplatňuje heuristická metoda nazvaná stochastický gradient descendant, více v kapitole 2.5. Gradient descant není optimální a může uváznout v lokálním minimu. Mimo Gradient descendant existují i další algoritmy pro hledání minima [15]:

- AdaGrad,
- AdaDelta,
- Adam
- Nesterov

Většina z nich jsou gradientní metody.

Jarmo Ilonen [5] zkoušel pro učení neuronových sítí použít algoritmus diferenciální evoluce [6], avšak konstatoval, že je výhodnější tento algoritmus pouze k ověření nalezeného řešení při použití gradientních metod než přímo pro trénování neuronové sítě.

Back propagation

Výše zmíněný princip učení neuronových sítí lze uplatnit pouze v případě jedné vrstvy, která slouží jako vstupní a současně výstupní vrstva. Neuronové sítě zvláště konvoluční jsou však většinou složeny z více vrstev. Pak lze tyto neuronové sítě nazývat hlubokými neuronovými sítěmi. Pro učení takových sítí se využívá algoritmus back propagation.

Princip [16] tohoto algoritmu je založen na výpočtu chyby z výstupu neuronové sítě s využitím výše zmíněných metod pro hledání minima cenové funkce a tuto chybu zpětně propagovat sítí až ke vstupní vrstvě při současné opravě vah a biasů, aby cenová funkce klesala. Pro každou vrstvu se počítá nová chyba na základě chyby předchozí vrstvy.

Specifika učení hlubokých neuronových sítí.

Hluboké neuronové sítě obsahují více než jednu skrytou vrstvu. Sítě s více skrytými vrstvami se dokáží učit komplexnějším problémům a lépe abstrahovat. Na příklad první vrstva dokáže rozpoznávat hrany, další vrstva základní tvary, další vrstva složitější tvary a tak dále. V důsledku je neuronová síť schopna se naučit rozpoznávat v obraze například povrchy, klasifikovat plemena psů a jiné komplexní úkony. Teoretické výsledky ukazují, že hluboké neuronové sítě dávají pro daný problém mnohem lepší výsledky než mělké neuronové sítě [17].

Při učení hlubokých neuronových sítí přichází problém s rozdílnou rychlostí učení vah v jednotlivých vrstvách. Zatímco váhy poslední vrstvy se učí rychle, tak váhy v předních vrstvách se učí velice pomalu. Tento problém způsobí, že neuronová síť se není schopna učit, případně se učí velice pomalu. Problém je znám po názvem mizející gradient [18]. Obecně lze říci, že gradient v hlubokých neuronových sítích je nestabilní a tíhne u předních vrstev sítě buď k mizení, nebo k explozi, tedy k neúměrnému růstu gradientu, což má za následek, že neuronová síť je neschopna se učit.

Pro velice jednoduchý model sítě, kde každá vrstva obsahuje pouze jeden neuron se gradient v dané vrstvě vypočítá jako součin produktů předchozích vrstev, kdy každý produkt je tvořen součinem derivace cenové funkce pro danou vrstvu a vah vstupující do dané vrstvy. Pro získání gradientu se tento součin vynásobí parciální derivací cenové funkce podle výstupu nerunové sítě. Problém mizení gradientu nastává, jestliže součin derivace cenové funkce a váhy je menší než 1, pak se gradienty exponenciálně zmenšují s každou vrstvou bližší vstupu do neuronové sítě. Stejný princip platí pro problém exploze gradientu, kdy součin derivace cenové funkce a vah v daných vrstvách je větší než 1.

Obecně problém trénování hlubokých neuronových sítí je v nestabilitě gradientu ve vrstvách bližších ke vstupu sítě při použití gradientních metod učení a tím zapříčinění velice rozdílných rychlostí učení vrstev. V důsledku se může stát, že poslední vrstvy sítě budou již přetrénované a bližší vrstvy nenatrénované, což by pro testovací data znamenalo, že síť není dobře natrénovaná.

Na příklad zvolíme cenovou funkci jako sigmoidu a počáteční inicializaci vah neuronové sítě jako normální rozdělení se střední hodnotou 0 a rozptylem 1. Pak maximální hodnota derivace cenové funkce je 0,25 a hodnoty vah mají maximální hodnotu menší než 1. Při uplatnění principu zmíněného výše, že vrstvy blíže ke vstupu mají maximální hodnotu gradientu rovnu součinu součinů 0,25 x W, kde $W < 1$, což značí exponenciální pokles gradientu a tím exponenciální pokles změny vah a biasů.

Z poznatků uvedených výše, lze konstatovat, že trénování hlubokých neuronových sítí je velice náročné. Existuje mnoho faktorů, které ovlivňují proces učení jako:

- volba aktivační funkce
- počáteční inicializace vah
- použitá metoda minimalizace cenové funkce
- koeficient učení
- architektura sítě a jiné hyper parametry.

2.5 Optimalizační metody

Většina optimalizačních metod má za úkol potlačit problém přetrénování neuronové sítě. Některé metody se však soustředí i na snížení paměťové náročnosti a zvýšení rychlosti procesu trénování neuronové sítě.

Stochastický gradient descendant

Gradient descendant počítá gradientní vektory přes všechny vzorky z trénovací sady, což je časově náročné na výpočet a také náročné na paměť, jelikož při strojovém učení se musejí všechny vzorky uchovávat v paměti. Jako standartní řešení se využívá heuristická metoda stochastický gradient descendant, který pracuje pouze s podmnožinou trénovacích vzorků, zde nazvanou jako skupina. Při dostatečně velké skupině vzorků lze říci, že stochastický gradient descendat dává výsledky blížící se výsledkům obecnému gradient descendant. Konkrétně při učení s učitelem je důležité, aby v této skupině vzorků byly rovnoměrně zastoupeny třídy klasifikace nebo segmentace, aby nedocházelo k velkým výkyvům cenové funkce a tím výkyvům při změnách vah a biasů. Rovnoměrné rozložení tříd ve skupině lze zajistit náhodným výběrem vzorků z trénovací sady během učení nebo přichystáním trénovací sady ve vlastní režii, tak aby třídy klasifikace či segmentace v postupně načítaných skupinách z trénovací sady, byly rovnoměrně rozloženy.

Strojové navyšování trénovací sady

Zvětšení velikosti trénovací sady má za následek lepší natrénování neuronové sítě a také snížení výskytu problému přetrénování. Princip strojového navýšení trénovací sady spočívá v transformacích existujících vzorků, například rotací nebo zkosením o určitý úhel. Tím lze vytvořit pro síť další vzorky, na kterých se bude učit.

Učení na podobrazech

Tato metoda slouží k navýšení počtu trénovacích vzorků rozřezáním původních obrazů na menší celky, které jsou považovány neuronovou sítí za plnohodnotné trénovací vzorky. Tím se dosáhne lepšího natrénování neuronové sítě a také snížení výskytu problému přetrénování. Dalším významem je zmenšení paměťové náročnosti a zrychlení procesu trénování sítě, jelikož síť pracuje pouze s menším vstupním obrazem, než byl původně.

Metoda dropout

Tato metoda se využívá především u plně propojených vrstev a jejím cílem je snížit výskyt problému přetrénování. Princip metody spočívá v úpravě počtu aktivních neuronů v dané vrstvě. Pro každou iteraci, tedy každou skupinu vzorků při použití stochastického gradient descendant se deaktivuje určitý počet neuronů, tento počet je dán koeficient v intervalu (0;1). Běžně je použit koeficient 0,5,

což značí, že každou iteraci se deaktivuje polovina neuronů v dané vrstvě. Na metodu lze pohlížet jako na trénování různých sítí, jejichž výsledky se v závěru průměrují.

Regularizace

Existují 2 běžně používané regularizace [19] L1 a L2 a obě se snaží o snížení výskytu problému přetrénování. L2 i L1 regularizace přidává k cenové funkci regularizační výraz. Na základě koeficientu regularizačního výrazu se neuronová síť učí více malé váhy nebo se naopak přiklání spíše k vyšším vahám.

Inicializace neuronové sítě

Velice důležitá je počáteční inicializace sítě, která může síť přiblížit ke globálnímu minimu cenové funkce a tím snížit nebezpečí uváznutí v lokálním minimu. Pro počáteční inicializaci lze využít již natrénované neuronové sítě pro danou úlohu, nebo při trénování nové sítě lze využít metod jako inicializace vah s normální rozdělením s přesně daným rozptylem [20] nebo inicializaci vah zvanou jako Xavier [21].

2.6 Nástroje pro práci s neuronovými sítěmi

Práce s neuronovými sítěmi vyžaduje optimalizované matematické algoritmy z důvodů velké výpočetní složitosti algoritmů, které vytváří funkčnost neurnových sítí. Proto je vhodné využít specializovaných frameworků, například:

- BVLC Caffe [27],
- Theano [32],
- Torch [33],
- Veles [34].

Současně nejpoužívanější Framework je BVLC Caffe.

BVLC Caffe Framework je open source nástroj pro deep learning dostupný na GitHub vyvíjený univerzitou v Berkeley. Umožňuje provádět výpočty na procesoru nebo na grafické kartě. Při využívání procesoru se uplatňují moduly psané v C++ a při práci na grafické kartě moduly psané pro CUDA. Tento Framework umožňuje volit počet jader procesoru, na která se rozdělí početní úlohy. U grafických karet lze také volit několik jader. Celý framework se snaží o rychlé vyhodnocování s využitím podpůrných knihoven jako BLAS, ATLAS, cuDNN a jiné. Pro návrh celé struktury neuronové sítě využívá Caffe formátu Protobuffer od Google, jedná se o serializační textový formát.

Ukázka konvoluční vrstvy v Protobufferu je zobrazena v následujícím úseku definice neuronové sítě:

```
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: 1024
    kernel_size: 13
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

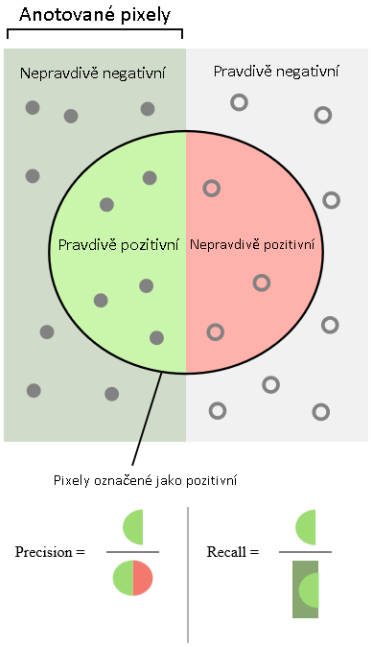
Caffe disponuje velkým množstvím typů vrstev [29]. Caffe také poskytuje možnost vytvořit si vlastní vrstvu s definovaných chování popsaným v Python nebo C++. Vnitřní reprezentace data v Caffe je jako tzv. Blob, což je čtyř rozměrná struktura obsahující čtveřici [počet vzorků, kanál, výška, šířka], kdy kanál představuje buď příslušný kanál z RGB modelu.

2.7 Výpočet F-measure

V této práci je využita metrika F-measure pro vyhodnocení úspešnosti segmentaci s využitím plně konvoluční neuronové sítě. Metrika je počítána pomocí presicion a recall. Pro výpočet těchto dvou hodnot je nutné rozdělit celý segmentovaný obraz tříd do čtyř skupin

- pravdivě pozitivní,
- nepravdivě pozitivní,
- pravdivě negativní,
- nepravdivě negativní

zařazení pixelu do dané třídy segmentace. Presicion definuje procento pravdivě pozitivních zařazení do dané třídy segmentace vůči všem pozitivním zařazením (pravdivě/negativně) do dané třídy segmentace. Recall definuje procento pravdivě pozitivních zařazení do dané třídy segmentace vůči všem anotovaným pozitivní zařazením do dané třídy. Pro názornost výpočet a význam precision a recall zobrazuje Obrázek 8.



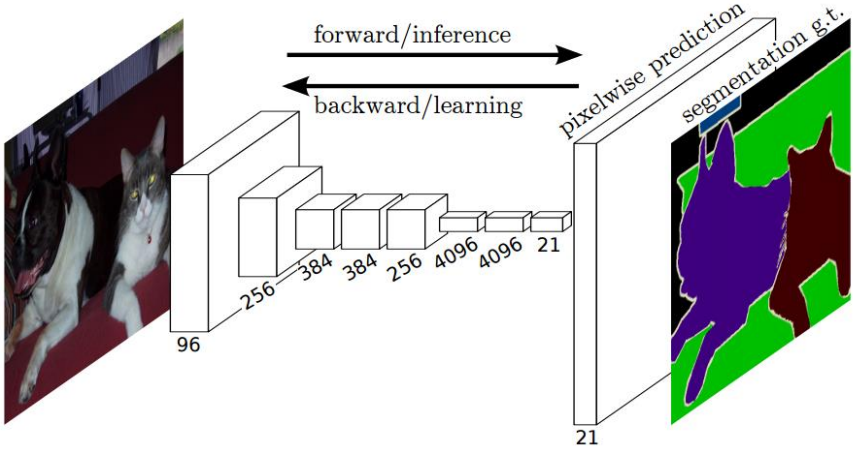
Obrázek 8 – znázornění významu a výpočtu precision a recall. Převzato z [35].

Pro výpočet F-measure byl zvolen vztah, který bere v potah precision a recall se stejnou váhou. Tento typ F-measure je označován jako F_1, viz vzorec (5).

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (5)$$

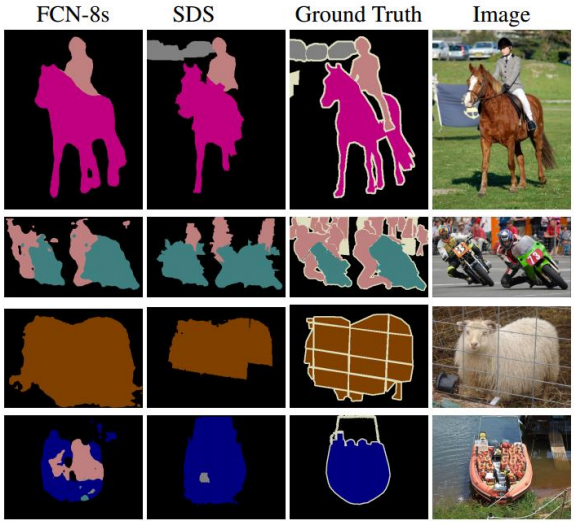
3 Použití konvolučních neuronových sítí

Konvoluční neuronové sítě se využívají zejména pro zpracování obrazových dat. Jonathan Long [22] využívá plně konvoluční neuronovou síť pro segmentaci 2D obrazů. Na místo plně propojené vrstvy je použita jako poslední vrstva konvoluční vrstva s velikostí jádra voleného tak, aby výstup z vrstvy dával rozměr 1x1 pixel a N kanálů. S využitím dekonvoluční vrstvy je znovu vytvořen obraz o původní velikosti vstupu. Architekturu sítě zobrazuje Obrázek 9.



Obrázek 9 – architektura neuronové sítě pro segmentaci 2D obrazu od Jonathana Longa. Převzato z [22].

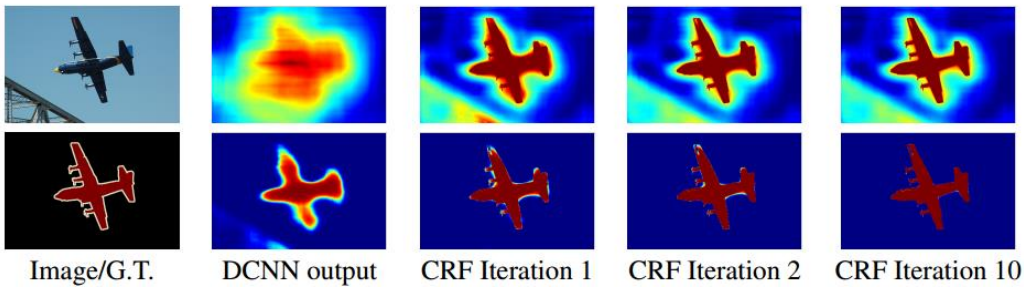
Long svým řešením dosahuje přesnosti na datové sadě PASCAL VOC 2011 a 2012 62,7%. Oproti state-of-the-art metodě dosáhl o 20% lepších výsledků. Výsledky prezentované v práci Longa ukazuje Obrázek 10.



Obrázek 10 – výsledky segmentace plně konvoluční sítě navrhnuté Jonathanem Longem v porovnání se state-of-the-art metodou SDS. Převzato z [22].

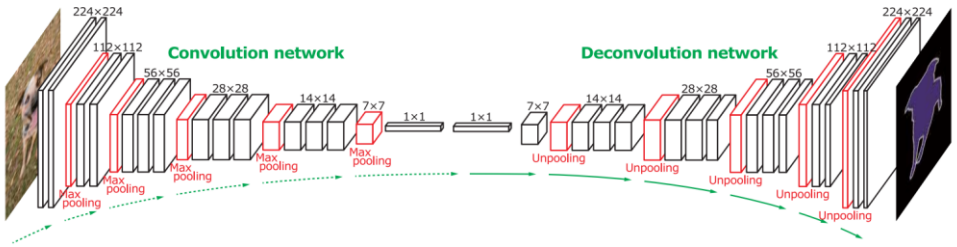
Další použití konvoluční neuronové sítě, viz [23], pro segmentaci 2D obrazů využívá dvou komponent. První komponentou je plně konvoluční síť a druhým komponentem je Condional

Random Field (CRF) [31]. Výstup plně propojené neuronové sítě je zvětšen bilineární interpolací a dále je zpracován pomocí CRF a tím je dosaženo zlepšení segmentace hran objektu. Tímto přístupem bylo dosaženo přesnosti segmentace na 71,6% s použitím datové sady PASCAL VOC 2012. Segmentaci letadla s využitím konvoluční neuronové sítě a CRF zobrazuje Obrázek 11.



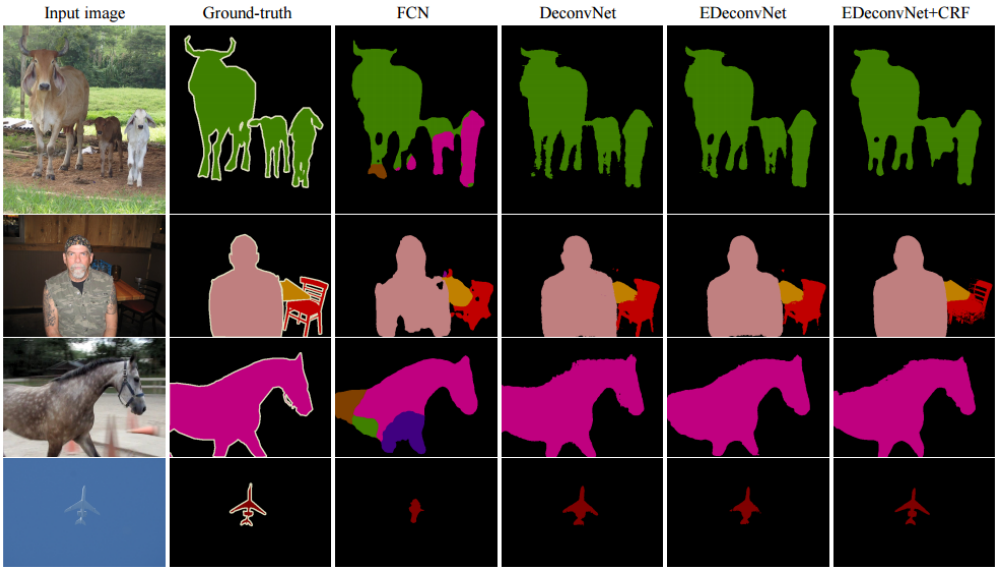
Obrázek 11 – výsledky segmentace s využitím CRF. První řádek zobrazuje výstup z neuronové sítě bez aplikace Softmax funkce. Zleva: Vstupní obraz, výstup z konvoluční sítě, dále zpracování pomocí CRF. Převzato z [23].

Jako další přístup, viz [24], pro zpracování 2D obrazu, konkrétně segmentaci, byla použita dekonvoluční síť[1], kterou lze považovat za plně konvoluční neuronovou síť. Tato síť je rozdělena na 2 části. V první části je vstupní obraz zpracován plně konvoluční neuronovou sítí, kdy autoři využívají stejného principu nahrazení plně propojené vrstvy vrstvou konvoluční jako Long. Ve druhé části zpracovává obraz dekonvoluční neuronová síť, jejíž struktura je zrcadlenou strukturou konvoluční neuronové sítě použité v první části architektury. Výstup této neuronové sítě je stejné velký jako vstup. Autoři uvádějí více různých konfigurací sítě, z nichž nejlepší výsledky dává dekonvoluční síť v kombinaci s CRF. S tímto řešením autoři dosahují přesnosti segmentace na datové sadě PASCAL VOC 2012 v průměru 72,5%. Nevýhoda je v době trénování, která je dvojnásobná oproti samotné konvoluční neuronové síti z důvodů existence zrcadlené dekonvoluční síti, autoři uvádějí, že trénování trvalo 6 dní. Další nevýhodou je velká paměťová náročnost při použití tohoto řešení, autoři využívali grafickou kartu NVIDIA GTX Titan s 12 GB paměti. Architekturu dekonvoluční neuronové sítě zobrazuje Obrázek 12. Ukázky výsledků segmentace zobrazuje Obrázek 13.



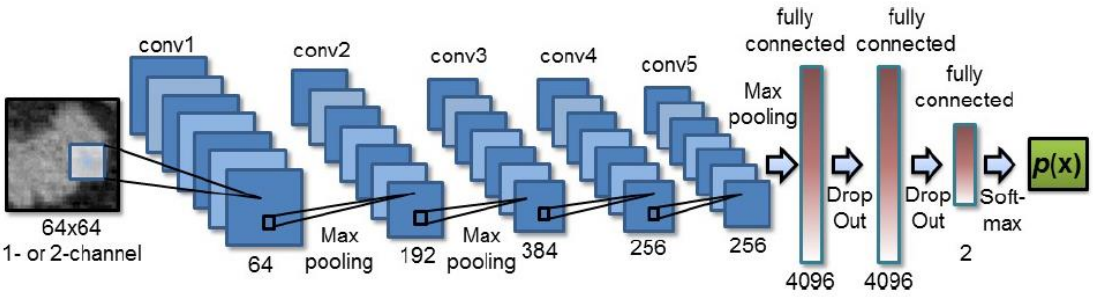
Obrázek 12 – architektura dekonvoluční sítě pro segmentaci 2D obrazu. Převzato z [24].

[1] Upravený BVLC Caffe Framework obsahující dekonvoluční síť lze stáhnout z http://cvlab.postech.ac.kr/research/deconvnet/



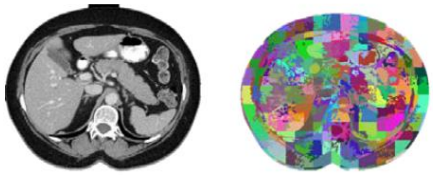
Obrázek 13 – výsledky segmentace s využitím dekonvoluční sítě v porovnání s plně konvoluční sítí. Nejlepší výsledky byly získány s architekturou EDeconvNet+CRF. Převzato z [24].

Z oblasti segmentace 3D obrazu bylo využito komplexního řešení, autory nazývané jako framework, pro segmentaci slinivky břišní z CT snímků [25], kde konvoluční neuronová síť byla použita pouze jako část vytvořeného frameworku. V tomto řešení bylo využito techniky superpixelů s využitím algoritmu SLIC [26], pomocí kterého byly předzpracovány vzorky vstupující do neuronové sítě. Výstup z konvoluční sítě byl zpracován metodou random forest. S využitím těchto přístupů bylo dosaženo úspěšnosti segmentace v průměru 64%. Architekturu konvoluční sítě použité při segmentaci slinivky břišní zobrazuje Obrázek 14.



Obrázek 14 – architektura konvoluční sítě využita pro segmentaci slinivky břišní. Převzato z [25].

Výsledek použití algoritmu SLIC pro vytvoření superpixelů ukazuje Obrázek 15.



Obrázek 15 – výsledek tvorby superpixelů s použitím algoritmu SLIC. Superpixely vstupují do konvoluční neuronové sítě. Převzato z [25].

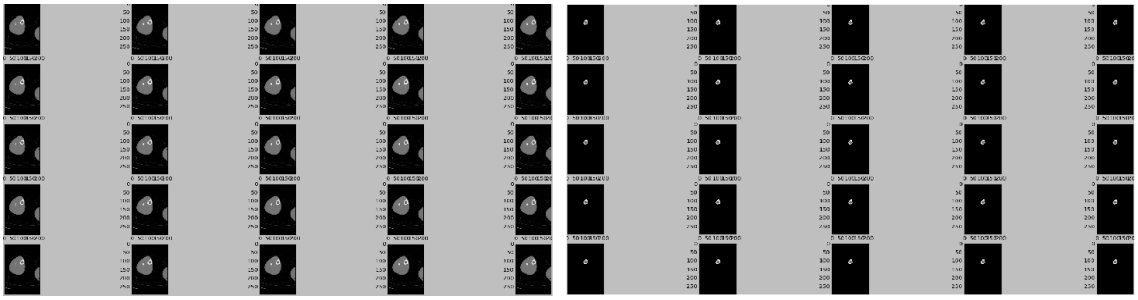
4 Návrh plně konvoluční neuronové sítě pro segmentaci obrazu

Tato kapitola popisuje mnou navrhované řešení a architektury neuronových sítí pro segmentaci holenní kosti z CT snímků, které mi poskytla společnost 3Dim laboratory, s.r.o. Kapitola také popisuje datovou sadu a způsob tvorby trénovací sady pro trénování na podobrazech a celých obrazech. Snaha byla, aby architektura neuronové sítě byla co nejjednodušší, avšak aby síť podávala kvalitní výsledky při segmentaci. Cíl pro úspěšnost segmentace jsem si stanovil na 90% přesnosti. Pro práci s neuronovými sítěmi byl použit BVLC Caffe Framework.

Datová sada

Bylo mi poskytnuto celkem 10 CT skenů naskenované dolní končetiny rozdílných pacientů s anotovanou holenní kostí. Snímky byly rozdílné kvality a rozdílných rozměrů. Anotované snímky obsahují hodnoty 0, kde není holenní kost, a hodnoty stejné jako v neanotovaném CT skenu, tam kde se holenní kost vyskytuje. CT skeny jsou uloženy ve formátu DICOM, což je standart pro zobrazování, distribuci, skladování a tisk medicínských dat pořízených snímacími metodami jako jsou CT, MRI či ultrazvuk. DICOM také definuje strukturu CT skenu jakožto objemového obrazu, který je tvořen vrstvami. Každá tato vrstva je uložena jako samostatný soubor s příponou DCM, kdy každý soubor obsahuje hlavičku s informacemi o celém DICOM svazku. Na každou vrstvu lze pohlížet jako na 2D obrázek. Z pohledu neuronových sítí je každá tato vrstva brána jako celý obrázek, který odpovídá jednomu vzorku. Tedy každý CT sken je složen z několika vzorků a pro segmentaci celého CT skenu je zapotřebí segmentovat každou tuto vrstvu. Musí být zajištěno rovnoměrné rozložení tříd segmentace v rámci celé trénovací sady. Ukázku neanotovaného a anotovaného CT skenu z datové sady ve 3D zobrazuje Obrázek 16 a k němu odpovídající zobrazení v podobě vrstev zobrazuje Obrázek 17.

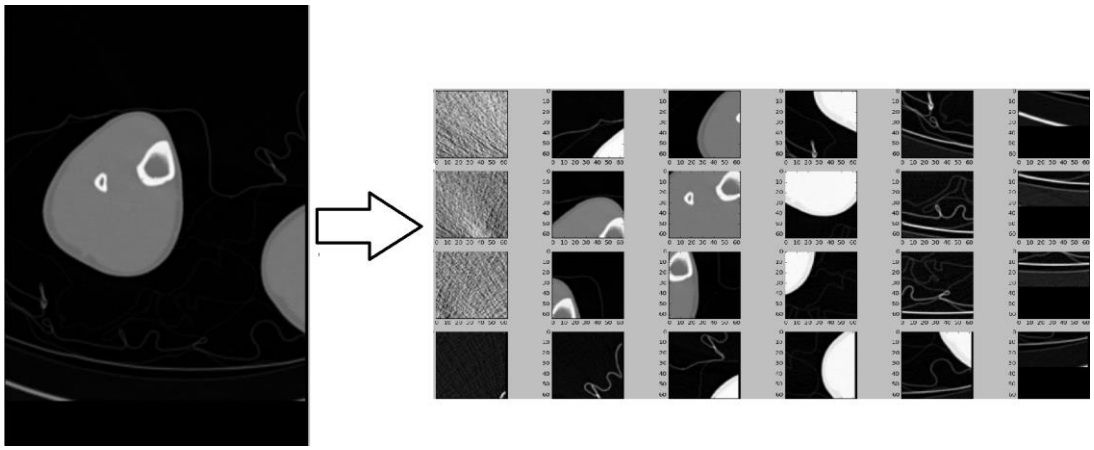
Obrázek 16 – ukázka CT sklenu z datové sady ve 3D. Zleva: neanotovaný CT sken a anotovaný CT sken.



Obrázek 17 – ukázka CT skenu z datové sady v podobě vrstev. Zleva: neanotovaný CT sken a anotovaný CT sken.

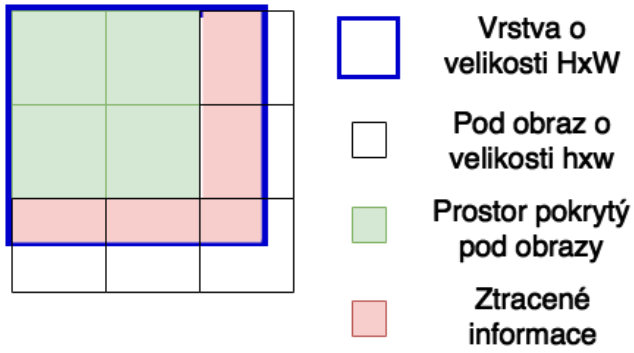
Trénování na podobrazech

Při tvorbě datové sady pro učení na podobrazech bylo nutné brát v úvahu princip konvolučních neuronových sítí a velikost výstupu z neuronové sítě, aby bylo zajištěno dostatečné překrývání jednotlivých podobrazů a tím poskytnutí maximálního množství informací pro konvoluční neuronovou síť. Ukázku podobrazů získaných z jedné vrstvy zobrazuje Obrázek 18.



Obrázek 18 – rozdělení jedné vrstvy CT skenu na podobrazy.

V souvislosti s přístupem trénování na podobrazech bylo nutné upravit velikost každé zpracovávané vrstvy CT skenu z důvodu nedělitelnosti velikosti podobrazu a velikosti dané vrstvy, aby nedocházelo ke ztrátě informací na okrajích vrstev. Tento problém zobrazuje Obrázek 19.



Obrázek 19 – problém ztráty informace na okrajích CT snímku při rozdělování na podobrazy.

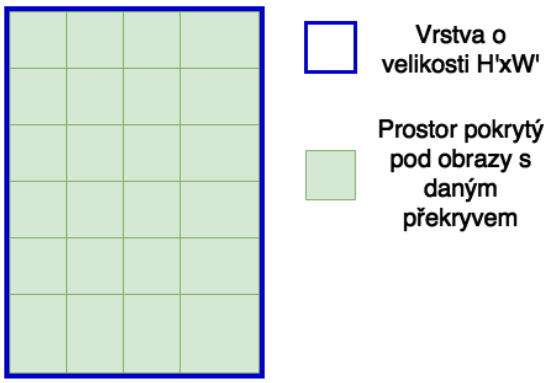
Tento problém byl vyřešen úpravou velikosti CT skenu podle vzorců:

$$W' = \left(\left\lfloor \frac{W_{slice} - (w_{patch} - w_{out})}{w_{out}} \right\rfloor \right) * w_{out} + w_{patch} \quad (6)$$

$$H' = \left(\left\lfloor \frac{H_{slice} - (h_{patch} - h_{out})}{h_{out}} \right\rfloor \right) * h_{out} + h_{patch} \quad (7)$$

Vzorce (5) a (6) jsou použity pro úpravu velikosti CT skenu, aby bylo dosaženo bezeztrátového rozdělení na podobrazy. Kde W‘ a H‘ značí novou šířku a výšku CT skenu , W_{slice} a H_{slice} definují původní rozměr CT skenu, w_{patch} a h_{patch} definují velikost podobrazu, w_{out} a h_{out} definují velikost výstupu z neuronové sítě.

Při aplikaci těchto vzorců dojde k tomu, že při rozdělování na podobrazy s daným překryvem nedojde ke ztrátě informací na okrajích CT skenu a do trénovací sady je zahrnut celý objem CT skenu. Při úpravě velikosti CT skenu dojde v drtivé většině případů k rozšíření velikosti, kde přidaný prostor je vyplněn nulovými hodnotami, tedy jakoby pozadím. Výsledek aplikace výše uvedených vzorců zobrazuje Obrázek 20.



Obrázek 20 – dělení po úpravě velikosti CT snímku na podobrazy s daným překryvem.

Výstupem plně konvoluční neuronové sítě jsou v případě trénování na podobrazech pod vzorkované podobrazy, které je nutno složit do výsledné vrstvy CT snímku. S využitím **Chyba! Nenalezen zdroj odkazů.** pro úpravu velikosti CT skenu, lze bez problémů složit výsledný CT sken z výstupů neuronové sítě.

Při rozšiřování CT skenu se nepřímo vytvoří podobrazy, které obsahují pouze pozadí. Toto řeší metodologie pro výběr podobrazů pro zařazení do trénovací sady.

Výběr podobrazů pro zařazení do trénovací sady

Tato metoda spočívá v nalezení všech podobrazů vytvořených neanotovaných CT skenů obsahujících pouze pozadí. Následně tyto podobrazy nejsou zařazeny do sady pro trénování a totéž platí pro odpovídající podobrazy z anotovaného CT snímku. Tento postup vychází z úvahy, že je zbytečné učit

neuronovou síť na prázdných datech, což by jednak zpomalilo proces učení a navíc by mohlo dojít ke zvýšení složitosti funkce, kterou neuronová síť aproximuje. Jednodušší řešení je takový podobraz nalézt a odstranit. Stejný postup je uplatněn při používání natrénované neuronové sítě, kdy každý takový podobraz je ihned přesměrován na výstup, kde je zmenšen na požadovanou velikost výstupu neuronové sítě. Při tvorbě trénovací sady je takto odstraněno asi 30% všech podobrazů.

Trénování na celých obrazech

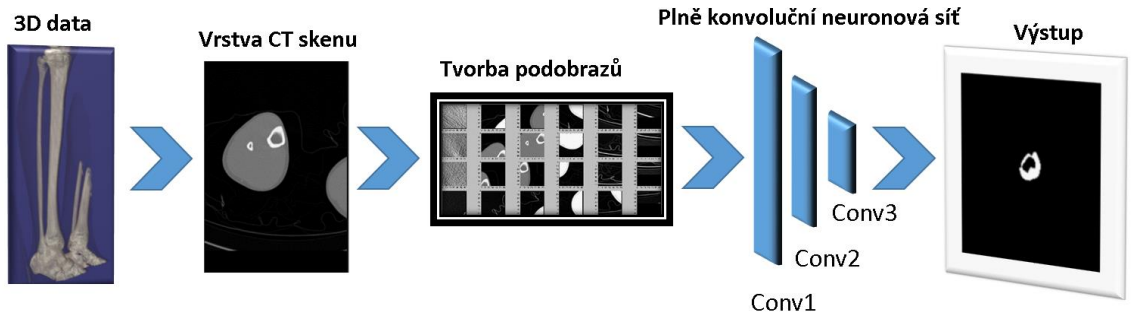
Příprava trénovací sady pro trénování na celých obrazech je mnohem jednodušší. U všech obrazů se upravuje velikost, která je volena podle CT skenu s největšími rozměry z důvodů sjednocení velikostí CT skenů v trénovací sadě. To přináší nevýhodu, jelikož datová sada obsahuje CT skeny rozdílných velikostí a při zvětšení malých CT skenů dochází k nárůstu velikosti až o 50%. Do prostoru, který vznikl zvětšením, byly vloženy hodnoty odpovídající pozadí, tedy hodnota 0.

Navrh konfigurace plně konvoluční neuronové sítě

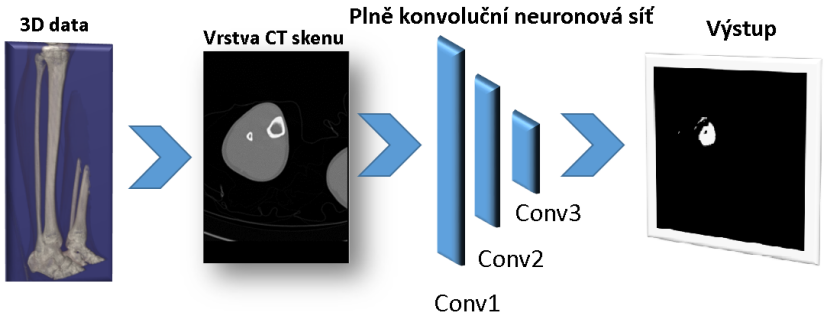
Cílem bylo vytvořit jednoduchou plně konvoluční neuronovou síť. Všechny konfigurace byly trénovány pomocí stochastického gradient descendat se skupinou vzorků o velikost 32, což je z hlediska klasifikace vzorků do 2 tříd a rovnoměrného rozložení zastoupení tříd klasifikace v trénovací sadě dostačující. Jako cenová funkce byla zvolena Cross Entropy s poslední vrstvou aktivovanou sigmoidou pro možnost učení na jednotlivých pixelech trénovacích vzorků. S ohledem na tuto cenovou funkci se musejí upravit anotace trénovací sady na hodnoty 0 pro pozadí a 1 pro segmentovanou holenní kost. Jako inicializační metoda vah u každé konvoluční vrstvy byla použita metoda Xavier, což vyžaduje normalizaci hodnot trénovacích vzorků do intervalu <0;1>.
Během experimentů bude třeba otestovat a odladit vhodné nastavení těchto hyper parametrů:

- Aktivační funkce
- Počet výstupů konvolučních jader
- Počet konvolučních vrstev neuronové sítě
- Koeficient učení

Přístup trénování na podobrazech zobrazuje Obrázek 21 a trénování na celých obrazech zobrazuje Obrázek 22.



Obrázek 21 – proces vyhodnocení při trénování na podobrazech.



Obrázek 22 – proces vyhodnocení při trénování na celých obrazech.

5 Implementace

Celý vývoj probíhal pod operačním systémem CentOS 7. Dalším využívaným nástrojem byla distribuce Python Anaconda, který poskytuje jednak interpret Pythonu a jednak velký počet doplňujících balíčků. Jako vývojové prostředí pro psaní python skriptů byl použit PyCharm Community.

S ohledem na Caffe Framework byl využit formát Lightning Memory-Mapped Database (LMDB) pro tvorbu datových sad. Jedná se o databázi pro časově kritické aplikace uchovávající data jako klíč a k němu hodnota. Celá trénovací sada byla rozdělena na dvě samostatné LMDB databáze, data a k nim odpovídající anotace. Při procesu učení pak Caffe Framework zajistí synchronizaci načítání vzorků z obou databází. Pro tvorbu těchto databází byl využit python 2,7 a PyCaffe API pro komunikaci s Caffe Framework. Před samotným vytvořením LMDB databází pro trénování sítě se musí rovnoměrně rozložit v rámci celé trénovací sady zastoupení tříd klasifikace. Jelikož v tomto případě existují 2 třídy klasifikace, pak stačí zajistit rozložení podobrazů, které obsahují anotovanou kost, vytvořených z anotovaných CT skenů. Při přípravě trénovací a testovací sady bylo nutné každý anotovaný CT snímek upravit tak, aby obsahoval pouze hodnoty 0 pro pozadí a 1 pro kost. Tímto byly definovány třídy segmentace. Tato úprava byla nutná také z důvodu volby cenové v navrhovaném řešení.

Python skripty

Byly vytvořeny 2 hlavní skripty, tvorba trénovací sady a vyhodnocení natrénované neuronové sítě s využitím trénovací sady. Dále bylo vytvořeno několik pomocných skriptů pro čtení data z LMDB, export vytvořených podobrazů a jiné. Všechny skripty jsou dostupné jako příloha v přiloženém CD.

Jako hlavní python balíček byl využit numpy, který zajišťuje podporu pro N-dimensionální výpočty, což je příhodné, jelikož skripty pracují s 3D obrazem a 4D Blob, které vyžaduje Caffe. Pro práci s DICOM soubory využívají skripty balíček Pydicom, který byl upřednostněn před VTK DicomReader z důvodu větší variabilnosti. Jelikož Pydicom načítá DICOM soubory po jednom, musí se zajistit pořadí, aby nedošlo k přeházení vrstev CT skenu. Jelikož konvence pojmenování vrstev bývá dána regulárním výrazem /%s%d.dcm/, pak pořadí, jaké nabídne operační systém, není správné. Z tohoto důvodu se musí využít python jakožto nástroj pro načítání souboru ve správném pořadí s využitím systémového času nebo s využitím znalosti struktury názvu souboru. Ukázka zdrojového kódu zajišťující načtení vrstev CT skenu ve správném pořadí ukazuje Obrázek 23.

```
files = []
lstFilesDCM = []  # create an empty list
for dirName, subdirList, fileList in os.walk(PathDicom):
    for filename in fileList:
        files.append(filename)
sorted_list = sorted(files, key=lambda y: int(y.rsplit('.',2)[0]))

for filename in sorted_list:
    if ".dcm" in filename.lower():  # check whether the file's DICOM
        lstFilesDCM.append(os.path.join(PathDicom,filename))
```

Obrázek 23 – Ukázka zdrojového kódu pro načítání vrstev ve správném pořadí s ohledem na znalost struktury názvů DICOM souborů.

Pro implementaci metriky F-measure je využit balíček numpy a díky tomu se výpočet velice zrychlil. Jelikož výstupní obrazy z neuronové sítě obsahují pouze hodnoty 0 pro negativní nebo 1 pro pozitivní aktivace a stejného rozměru výstupu sítě a anotovaného obrazu, pak pro počet pravdivě pozitivních hodnot lze použít funkci součinu N-dimensionálních polí a následně sumy, které zajišťuje numpy. Výsledek součinu je dán N-dimensionálním polem s hodnotami součinů prvků odpovídajícím stejné pozici v násobených polích. Precision je pak dán jako podíl součtu hodnot v poli daného součinem výstupu neuronové sítě a anotovaného obrazu děleno součtem hodnot z výstupu neuronové sítě. Recall je dán obdobně jako presicion, avšak dělitel je roven součtu hodnot anotovaného obrazu. Implementaci ukazuje Obrázek 24.

```
###########PRECISION and RECALL################
F = numpy.zeros(len(inNet),dtype='float')
k=0
for n in inNet:
    label = parsedLabels[n][:,:,numpy.newaxis]
    sumL = label.sum()
    sumO = normalizedOut[n].sum()
    pom = (normalizedOut[n]*label).sum()
    if sumL == 0 and sumO == 0:
        F[k] = 1
    elif pom == 0:
        F[k] = 0
    else:
        rec = pom/sumL
        prec = pom/sumO
        F[k] = (2*((prec*rec)/(prec+rec)))
    k+=1
Fvolume = F.sum()/F.shape[0]
scores.append(Fvolume)
```

Obrázek 24 – Implementace výpočtu F-measure přes všechny výstupy neuronové sítě

Dalším zajímavým využití balíčku numpy je při normalizaci výstupu neurnové sítě podle daného prahu. Celé pole obsahující hodnoty výstupu se porovná vůči danému prahu a výsledkem je pole obsahující hodnoty *True* pokud hodnota překročila práh nebo *False* pokud hodnota nepřekročila práh.

Následně se využije možnosti numpy a jako index pole výstupů se využije pole s hodnotami *True*/*False*, tam kde je *True* se doplní hodnota 1 a ostatní se nechají původní. Následně se tento postup zopakuje s opačnou podmínkou porovnání prahu a na pozice *True* se doplní hodnota 0. Zdrojový kód implementující tuto techniku zobrazuje Obrázek 25.

```
normalizedOut = []
for n in range(0,len(lstOutDCM)):
    normalizedOut.append(copy.deepcopy(lstOutDCM[n].transpose((1,2,0))))
    idx = normalizedOut[n][:,:,0] >= threshold
    normalizedOut[n][idx,0] = 1
    idx = normalizedOut[n][:,:,0] < threshold
    normalizedOut[n][idx,0] = 0
```

Obrázek 25 – Implementace prahování s využitím balíčku numpy

6 Experimenty a výsledky

Tato kapitola obsahuje konfigurace plně konvolučních neuronových sítí, se kterými byly prováděny experimenty. V druhé části kapitola prezentuje dosažené výsledky experimentů.
Experimenty probíhaly na stroji s následující parametry:

- CPU: Intel® Core™ i7-4790K CPU @ 4.00 GHz
- RAM: 8 GB DDR3 2666 MHz
- GPU: NVIDIA GeForce GTX 970 4GB GDDR5 (využito pro trénování)
- HDD: 320GB SATA 3

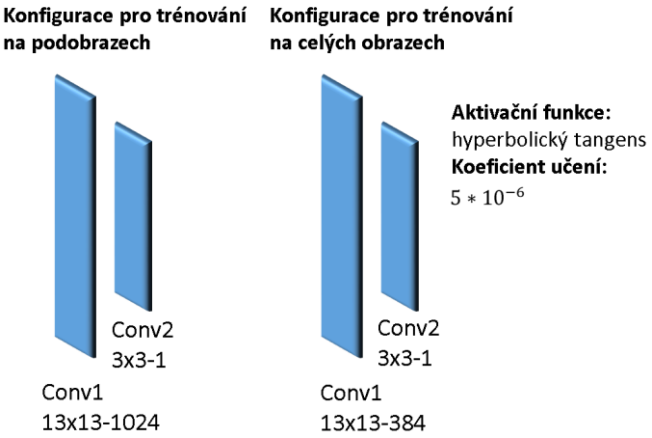
6.1 Experimenty

Různými kombinacemi hyper parametrů:

- Aktivační funkce {Hyperbolický tangens. ReLU}
- Počet výstupů konvolučních jader {Malý počet, maximální možný s ohledem na paměťovou kapacitu}
- Počet konvolučních vrstev neuronové sítě {2,3}
- Koeficient učení {$5*10^{-6}$, 10^{-6}. $5*10^{-7}$, 10^{-7}}

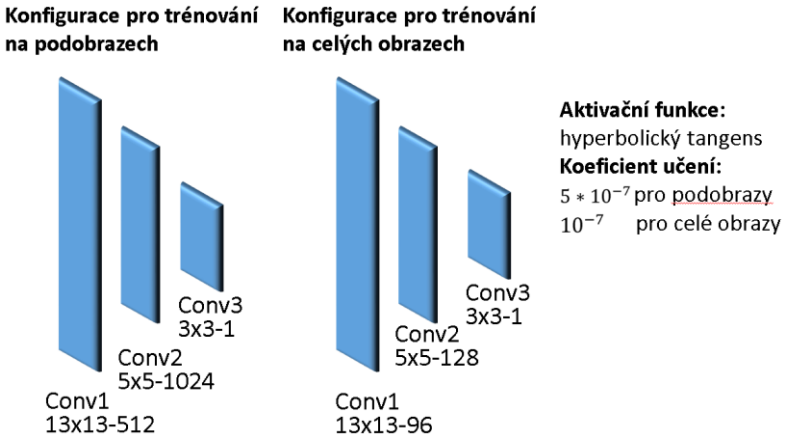
bylo vytvořeno 5 konfigurací plně konvoluční neuronové sítě a každá konfigurace byla trénována na celých obrazech a na podobrazech. Byly zvoleny 2 velikosti podobrazů: 64x64 a 46x46 pixelů.
V uvedených konfiguracích lze vidět metodiku návrhu plně konvolučních neuronových sítí pro segmentaci CT snímků, kdy v každé konfiguraci se mění pouze jeden hyper parametr oproti konfiguraci předchozí, aby bylo možno posoudit jaký vliv má tato změna. Výjimku tvoří konfigurace 1 a 2, kde se mění 2 hyper parametry.

Konfigurace 1 vytváří neuronovou sítě složenou z 2 konvolučních vrstev. Velikosti konvolučních jader byly zvoleny 13x13 a 3x3 s počty výstupů těchto jader jako 1024 a 1 pro trénování na podobrazech. Pro trénování na celých obrazech byly z důvodu nedostatečné paměti zvoleny výstupy konvolučních jader jako 384 a 1. Aktivační funkce byla zvolena jako hyperbolický tangens a koeficient učení $5*10^{-6}$.



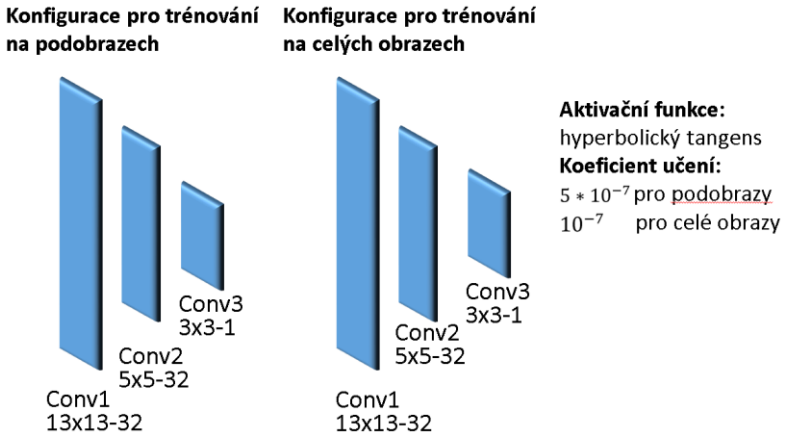
Obrázek 26 – konfigurace 1 plně konvoluční neuronové sítě pro segmentaci CT snímků.

Konfigurace 2 byla rozšířena o další konvoluční vrstvu, která měla rozměr konvolučního jádra 5x5. Výstupy konvolučních operátorů byly postupně od první vrstvy 512, 1024 a 1 trénování na podobrazech. Pro trénování na celých obrazech byly počty výstupů z konvolučních operátorů zvoleny 96, 128 a 1 z důvodů nedostatečné paměti. Koeficient učení byl změněn na $5*10^{-7}$ pro trénování na podobrazech a 10^{-7} pro trénování na celých obrazech. Ostatní hyper parametry zůstaly stejné jako v konfiguraci 1.



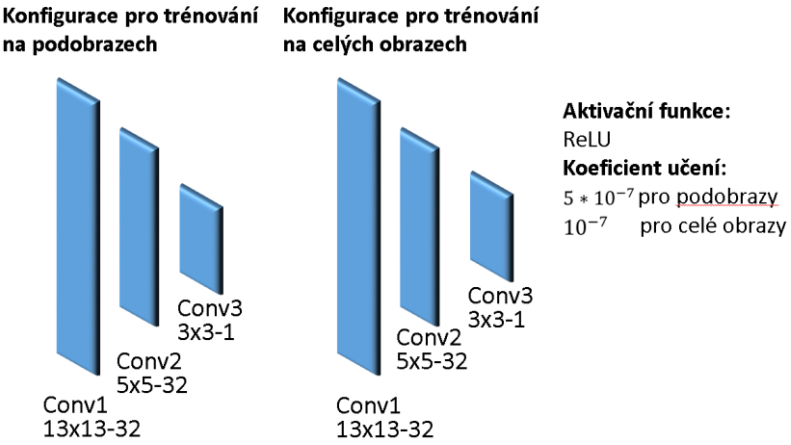
Obrázek 27 – konfigurace 2 plně konvoluční neuronové sítě pro segmentaci CT snímků.

Konfigurace 3 mění oproti konfiguraci 2 pouze počty výstupů konvolučních operátorů a to na 32, 32 a 1 pro oba přístupy trénování.



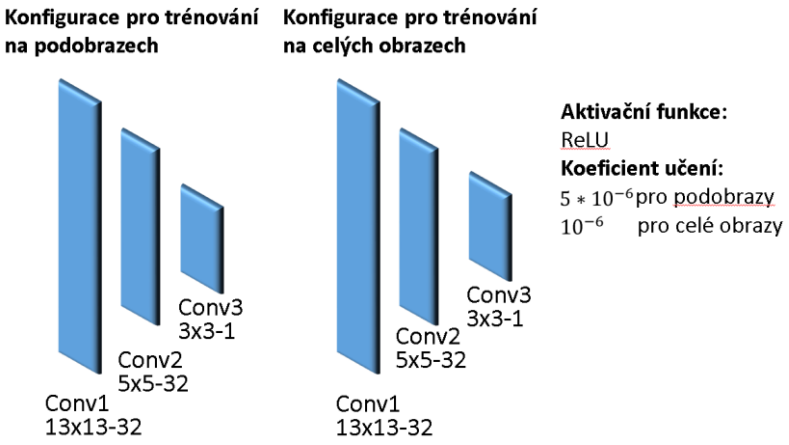
Obrázek 28 – konfigurace 3 plně konvoluční neuronové sítě pro segmentaci CT snímků.

Konfigurace 4 mění aktivační funkce z hyperbolický tangens na ReLU. Ostatní hyper parametry zůstávají stejné jako v konfiguraci 3.



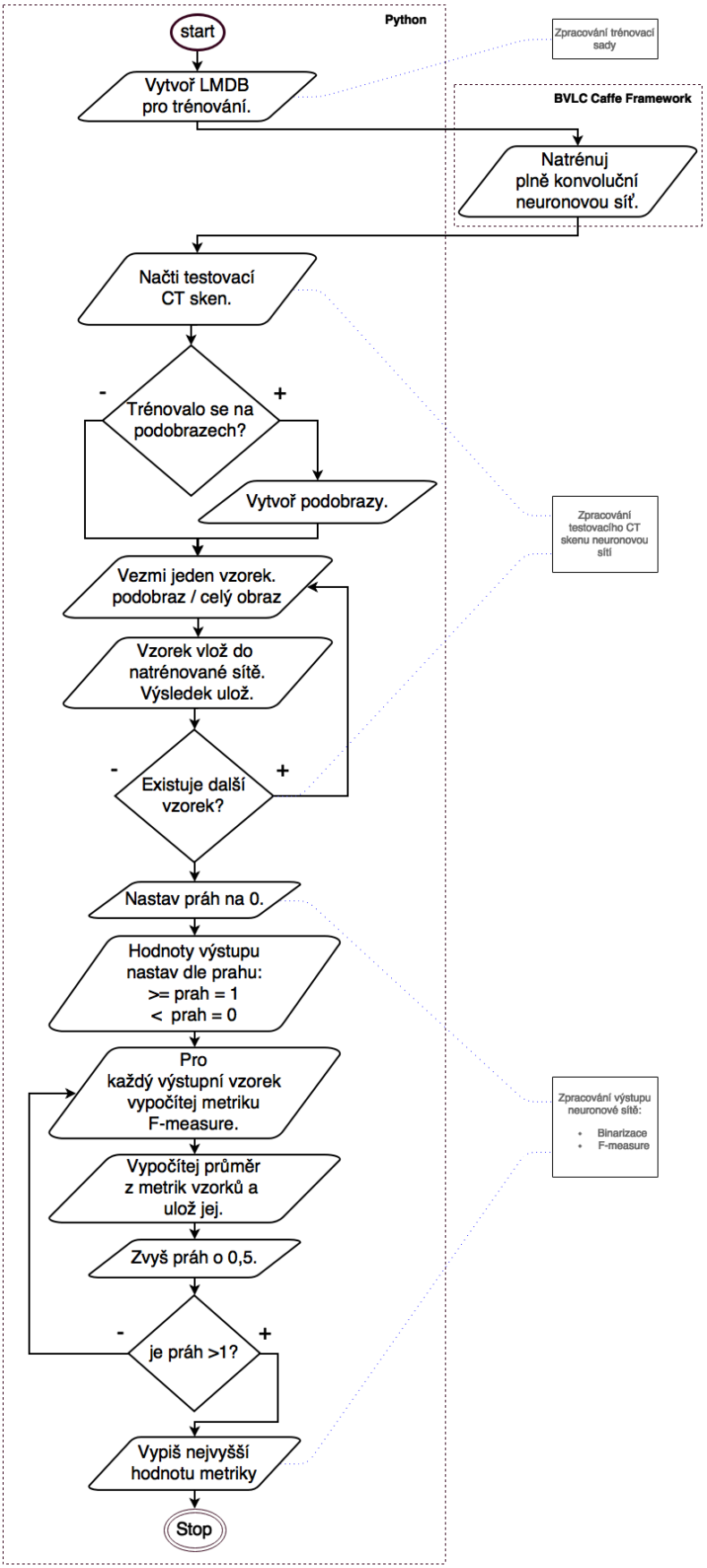
Obrázek 29 – konfigurace 4 plně konvoluční neuronové sítě pro segmentaci CT snímků.

Konfigurace 5 mění velikost koeficientu učení na $5*10^{-6}$ pro trénování na podobrazech a 10^{-6} pro trénování na celých obrazech. Ostatní hyper parametry zůstávají stejné jako v konfiguraci 4.



Obrázek 30 – konfigurace 5 plně konvoluční neuronové sítě pro segmentaci CT snímků.

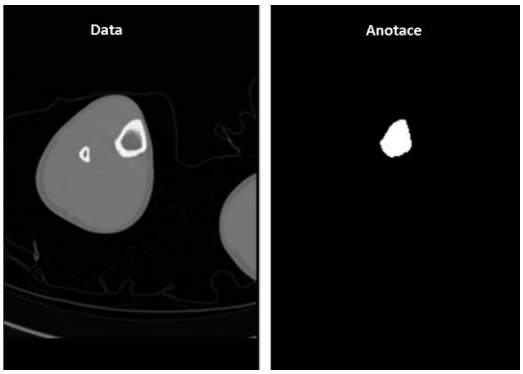
Úspěšnost segmentace holenní kosti z CT snímků výše navrhnutých konfigurací byla měřena metrikou F-measure [28] s využitím precision a recall, Dále je měřena doba trénování po 20 000 iterací při dané velikosti skupiny vzorků pro stochastický gradient descendant. Pro srovnání byl proveden experiment a měření úspěšnosti segmentace kosti pomocí přístupu vyhledání nejlepšího prahu Hounsfieldových jednotek s maximální odchylkou 20 jednotek.
Proces jednoho experimentu s plně konvoluční neuronovou sítí zobrazuje Obrázek 31.



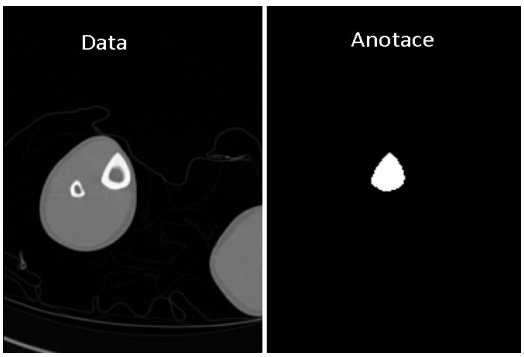
Obrázek 31 – diagram procesu jednoho experimentu s danou konfigurací neuronové sítě.

6.2 Výsledky experimentů

Pro trénování a testování navržených konfigurací plně konvolučních neuronových sítí bylo nutné vytvořit trénovací a testovací sady obsahující vzorky o určité velikosti odpovídající výstupu z neuronové sítě dané konfigurace. Pro trénovací data bylo použito 9 CT skenů a jeden CT sken byl ponechán pro tvorbu testovací sady. Všechny ukázky výstupů plně konvoluční neuronové sítě jsou prezentovány na 650. a 492. vrstvě testovacího CT snímku, který je zobrazuje Obrázek 32 a Obrázek 33. Výsledky pro více vrstev jsou přiloženy jako příloha D. Nutno brát však v potaz, že výsledná přesnost pomocí F-measure byla počítána přes všechny výstupní vzorky testovacího CT skenu. Velikost trénovací sady pro trénování na celých obrazech činila 8804 vzorků, kdy každá vrstva CT skenu odpovídá jednomu vzorku. Nezávisle na konfiguraci neuronové sítě byly zvoleny 2 velikosti podobrazů 64x64 a 46x46, kdy každá vrstva daného CT skenu byla rozdělena na tyto poobrazy. Tím bylo dosaženo navýšení trénovací sady na 105 103 a 225 378 vzorků při největším překryvu podobrazů.



Obrázek 32 – vrstva 650 z testovacího CT skenu. Zleva: data vstupující do neuronové sítě a požadovaný výstup.



Obrázek 33 – vrstva 492 z testovacího CT skenu. Zleva: data vstupující do neuronové sítě a požadovaný výstup.

Celkové výsledky experimentů ukázaly, že trénování na podobrazech při omezené datové sadě je efektivnější a lepší než trénování na celých obrazech. S využitím přístupu podobrazů bylo dosaženo přesnosti segmentace 95.1% při velikosti podobrazů 46x46 a s konfigurací 5, viz Obrázek 30 na straně 32. Použitím techniky nalezení optimálního prahu Hounsfieldových jednotek pro segmentaci kostí bylo dosaženo přesnosti segmentace 42,4%.

Výsledky experimentů navržených konfigurací plně konvolučních neuronových sítí shrnuje Tabulka 1. Kompletní výsledky s definovanými konfiguracemi a časy trénování jsou zobrazeny v příloze A.

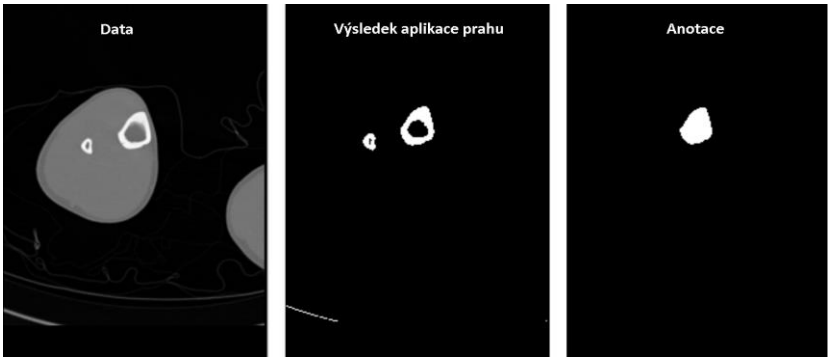
Tabulka 1 – výsledky experimentů pro dané konfigurace neuronové sítě a daný přístup učení, trénování na podobrazech velikosti 64x64 nebo 46x46 a trénování na celých obrazech.

Konfigurace	Podobrazy 64x64	Podobrazy 46x46	Celé obrazy
Konfigurace 1	88,45%	92,8%	52,26%
Konfigurace 2	92,45%	93,96%	50,03%
Konfigurace 3	92,16%	94%	46,27%
Konfigurace 4	92,39%	94,38%	50,88%
Konfigurace 5	93%	95,1%	61,29%

Následující obrázky zobrazují:

- obraz získaný nalezením optimálního prahu Hounsfieldových jednotek pro segmentaci kostí
- výstupy z neuronové sítě pro konfiguraci 5 při trénování
  - na podobrazech
  - na celých obrazech

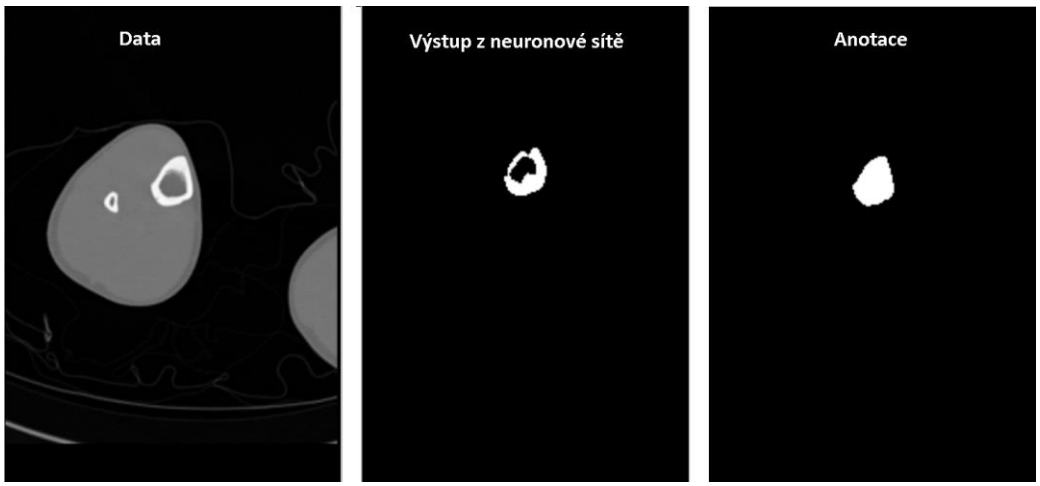
Segmentace pomocí nalezeného optimálního prahu Hounsfieldových jednotek



Obrázek 34 – segmentace získaná aplikací nalezeného optimální prahu Hounsfieldových jednotek pro segmentaci kosti. Dosažená přesnost 42,4%. Zleva: Vstupní data, výstup aplikací prahu, anotace.

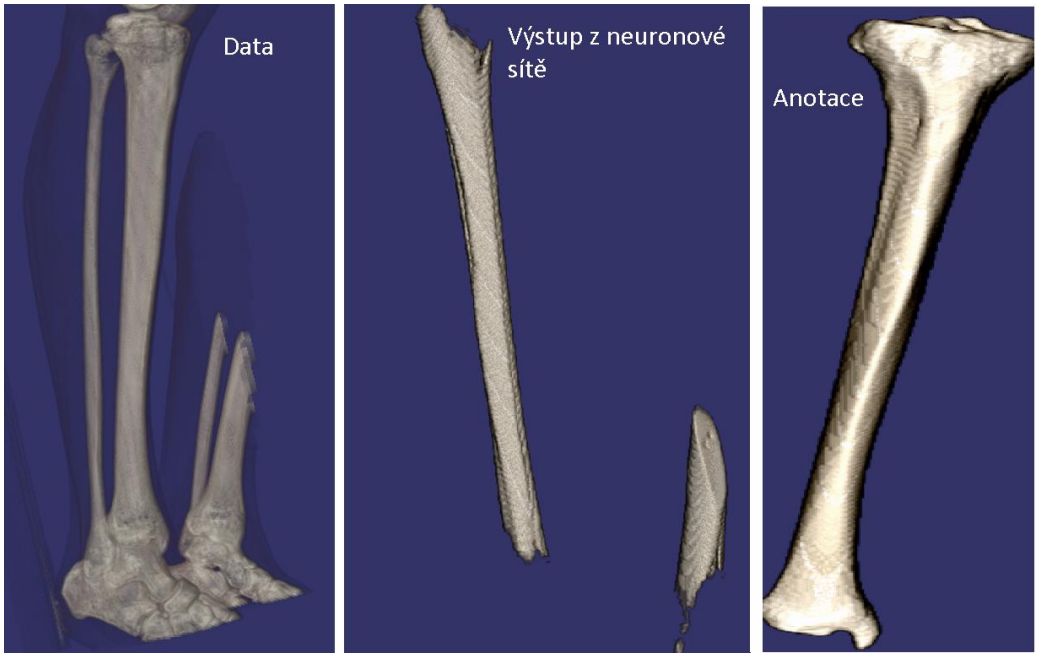
Trénování na podobrazech

Obrázek 35 – podobraz získaný z 650. vrstvy testovacího CT skenu. Dosažená přesnost 95,1%. Zleva: vstup do neuronové sítě, výstup z neuronové sítě a anotace.



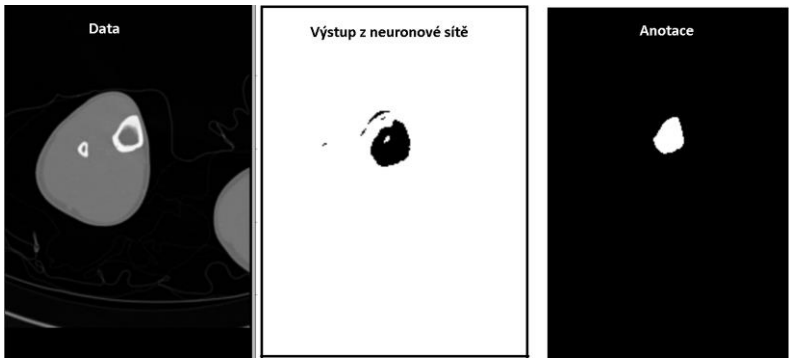
Obrázek 36 – trénování na podobrazech, 650. vrstva testovacího CT skenu. Dosažená přesnot 95,1%. Zleva: vstupní data, výstup z neuronové sítě po složení podobrazů do výsledné vrstvy a požadovaný výstup.

Obrázek 37 – podobraz získaný ze 492. vrstvy testovacího CT skenu. Dosažená přesnost 95,1%. Zleva: vstupní data, výstup z neuronové sítě a anotace.

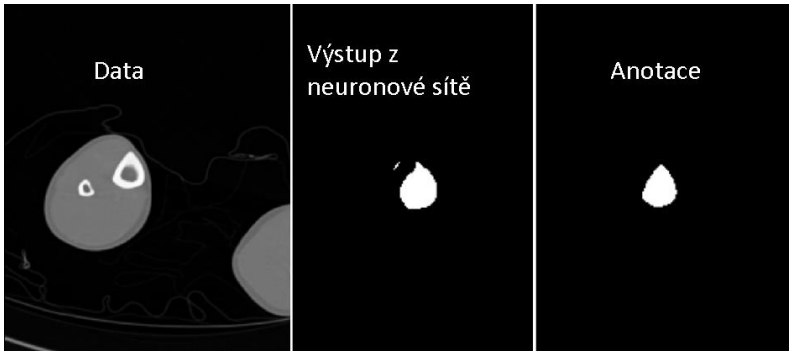


Obrázek 38 – testovací CT sken zobrazený ve 3D. Zleva: vtstupní data, výstup z neuronové sítě a anotace.

Trénování na celých obrazech



Obrázek 39 – trénování na celých obrazech, 650. vrstva testovacího CT skenu. Dosažená přesnost 61,29%. Zleva: vstupní data, výstup z neuronové sítě a anotace.



Obrázek 40 – trénování na celých obrazech, 492. vrstva testovacího CT skenu. Dosažená přesnost 61,29%. Zleva: vstupní data, výstup z neuronové sítě a anotace.

Při porovnání metody prosté aplikace prahu a výstupu neuronové sítě je vidět, že neuronová síť se naučila správně rozpoznávat holenní kost oproti ostatním. Při aplikaci prahu se ve výsledném obraze vyskytuje i neanotovaná kost, jelikož vyhovuje danému prahu, zatímco neuronová síť tuto kost ve výsledném obraze nezobrazuje.

7 Závěr

Cílem bylo vytvořit plně konvoluční neuronovou síť, která má za úkol segmentaci holenní kosti z CT skenů s úspešností převyšující 90%. Dílčím cílem bylo experimentování s trénováním neurnových sítí se dvěma přístupy tvorby trénovací sady, podobrazy a celé obrazy, jestliže jsme omezeni velikostí datové sady, což je hlavní problém při trénování na medicinských datech. Výsledkem je plně konvoluční neuronová síť dosahující přesnosti segmentace holenní kosti z CT snímku na 95,1% s využitím trénování na podobrazy. Oproti nejlepšímu výsledku neuronové sítě trénované na celých obrazech byla konfigurace trénovaná na podobrazech úspěšnější o 32,81%. Trénování na podobrazech se jeví jako lepší z hlediska úspěšnosti segmentace a také z hlediska časové a paměťové náročnosti procesu trénování nerunové sítě.

Návrhu nejlepší konfigurace plně konvoluční neuronové sítě předcházelo studium a pochopení principů neuronových sítí a konvolučních neuronových sítí. Dále jsem zkoumal různé přístupy segmentace obrazu zveřejněných autory v podobě vědeckých článků.

Při samotném návrhu neuronové sítě jsem narážel na problémy způsobující nefunkčnost navržené sítě. Zásadní problém byl ve způsobu tvorby trénovací sady. Chybou byla také tvorba příliš rozsáhlých neurnových sítí již od samotného začátku, což způsobovalo příliš velkou časovou náročnost procesu trénování a tím pádem příliš dlouhou dobu odezvy na provedené změny v dané konfiguraci sítě. Při trénování na celých obrazech jsem narážel na problémy s nedostatečnou video pamětí potřebnou pro nerunovou síť trénovanou na grafické kartě NVIDIA GeForce GTX 970 4 GB GDDR5.

Do budoucna by bylo možné implementovat do BVLC Caffe Framework vrstvu, která bude zajišťovat 3D konvoluci, aby bylo možné experimentovat opravdu s 3D obrazy či 3D podobrazy. Avšak tato skutečnost by vyžadovala výkonnější grafickou kartu osazenou pamětí o vyšší kapacitě. Pro budoucí vývoj bych určitě vyměnil pevný disk, který používám, za SSD disk, jelikož při práci s velkými objemy dat docházelo ke zpomalování celého procesu učení z důvodu rychlosti klasického pevného disku s magnetickou záznamovou vrstvou. Při použití mnou navržené konfigurace není segmentace hran přesná, proto by v budoucnu mohlo být využito metody CRF [31] pro dodatečné zpracování výstupu neuronové sítě pro zlepšení segmentace hran.

Literatura

[1] Deep, Big, Simple Neural Nets for Handwritten Digit Recognition
Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber
Neural Computation, December 2010, Vol. 22, No. 12 , Pages 3207-3220
(doi: 10.1162/NECO_a_00052)

[2] BENABDELKADER, Chiraz, Ross CUTLER, Harsh NANDA a Larry DAVIS. EigenGait: Motion-Based Recognition of People Using Image Self-Similarity. Springer Berlin Heidelberg. 2001, 2001(2091), 284-294. DOI: 10.1007/3-540-45344-X_42. ISSN 0302-9743.

[3] Neuralnetworksanddeeplearning.com: Using neural nets to recognize handwritten digits. Neuralnetworksanddeeplearning.com [online]. Michael Nielsen, 2016 [cit. 2016-04-16]. Dostupné z: http://neuralnetworksanddeeplearning.com/chap1.html

[4] Google [online]. [cit. 2016-04-16]. Dostupné z: https://scholar.google.cz/scholar?q=neural+network+architecture&btnG=&hl=cs&as_sdt=0%2C5&as_vis=1

[5] ILONEN, Jarmo, Joni-Kristian KAMARAINEN a Jouni LAMPINEN. Differential Evolution Training Algorithm for Feed-Forward Neural Networks. Neural proccessing letters. 2003, 2003(17), 93-105. DOI: 10.1023/A:1022995128597. ISSN 1573-773X.

[6] An Introduction to Differential Evolution. Www.maths.uq.edu.au [online]. Kelly Fleetwood [cit. 2016-04-16]. Dostupné z: http://www.maths.uq.edu.au/MASCOS/Multi-Agent04/Fleetwood.pdf

[7] JAGTAP, Virag. Artificial Neural Networks. In: VIRAG [online]. 2014 [cit. 2016-04-16]. Dostupné z: http://ann-machinelearning.blogspot.cz/

[8] Socher, Richard; Lin, Cliff; Ng, Andrew Y.; Manning, Christopher D. "Parsing Natural Scenes and Natural Language with Recursive Neural Networks". The 28th International Conference on Machine Learning (ICML 2011). New York. 2011, 129 – 136.

[9] CAO, Jinde a Jun WANG. Global asymptotic and robust stability of recurrent neural networks with time delays. IEEE Transactions on Circuits. 2005, 2005(53), 417 - 426. DOI: 10.1109/TCSI.2004.841574. ISSN 1549-8328.

[10] Cybenko, George. "Approximation by superpositions of a sigmoidal function." Mathematics of control, signals and systems 2.4 (1989): 303-314.

[11] SANGER, Terence D. Optimal unsupervised learning in a single-layer linear feedforward neural network. Elsevier [online]. Massachusetts Institute of Technology USA, 1989, 2(6), 549-473 [cit. 2016-04-18]. DOI: 10.1016/0893-6080(89)90044-0. Dostupné z: http://www.sciencedirect.com/science/article/pii/0893608089900440

[12] MILLER, W, Richard S SUTTON a Paul J WERBOS. Neural networks for control. První vydání. Cambridge, Mass.: MIT Press, 1990. ISBN 02-621-3261-3.

[13] UFLDL tutorial: Softmax regression. Ufldl.stanford.edu/ [online]. Stanford, http://ufldl.stanford.edu/ [cit. 2016-04-18]. Dostupné z: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

[14] RADY, Hussein Aly Kamel. Shannon Entropy and Mean Square Errors for speeding the convergence of Multilayer Neural Networks: A comparative approach. Egyptian Informatics Journal [online]. 2011, 12(3), 197-209 [cit. 2016-04-18]. DOI: 10.1016/j.eij.2011.09.002. ISSN 11108665. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1110866511000399

[15] Caffe: Solver. Caffe [online]. Berkeley university [cit. 2016-04-18]. Dostupné z: http://caffe.berkeleyvision.org/tutorial/solver.html

[16] How the backpropagation algorithm works: The four fundamental equations behind backpropagation. Neuralnetworksanddeeplearning [online]. Michael Nielsen, 2016 [cit. 2016-04-21]. Dostupné z: http://neuralnetworksanddeeplearning.com/chap2.html

[17] PASCANU, Razvan, Guido MONTUFAR a Yoshua BENGIO. On the number of response regions of deep feed forward networks with piece-wise linear activations [online]. 2014 [cit. 2016-04-21]. Dostupné z: http://arxiv.org/abs/1312.6098

[18] HOCHREITER, Sepp, Yoshua BENGIO, Paolo FRASCONI a Jürgen SCHMIDHUBER. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies [online]. 2001 [cit. 2016-04-21]. DOI: 10.1.1.24.7321. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.7321

[19] GIROSI, Federico, Michael JONES a Tomaso POGGIO. Regularization Theory and Neural Networks Architectures. Neural Computation [online]. 1995, 7(2), 219-269 [cit. 2016-04-21]. DOI: 10.1162/neco.1995.7.2.219. ISSN 0899-7667. Dostupné z: http://www.mitpressjournals.org/doi/abs/10.1162/neco.1995.7.2.219

[20] Improving the way neural networks learn: Weight Initialization. Neuralnetworksanddeeplearning. [online]. Michael Nielsen, 2016 [cit. 2016-04-21]. Dostupné z: http://neuralnetworksanddeeplearning.com/chap3.html#weight_initialization

[21] GLOROT, Xavier a Yoshua BENGIO. Understanding the difficulty of training deep feedforward neural networks [online]. 2010 [cit. 2016-04-21]. Dostupné z: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.207.2059

[22] LONG, Jonathan, Evan SHELHAMER a Trevor DARRELL. Fully Convolutional Networks for Semantic Segmentation [online]. 2014 [cit. 2016-04-23]. Dostupné z: http://arxiv.org/abs/1411.4038

[23] CHEN, Liang-Chieh, George PAPANDREOU, Iasonas KOKKINOS, Kevin MURPHY a Alan L. YUILLE. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs [online]. 2014 [cit. 2016-04-23]. Dostupné z: http://arxiv.org/abs/1412.7062

[24] HYEONWOO, Noh, Hong SEUNGHOON a Han BOHYUNG. Learning Deconvolution Network for Semantic Segmentation [online]. 2015 [cit. 2016-04-23]. Dostupné z: http://arxiv.org/abs/1505.04366

[25] FARAG, Amal, Le LU, Holger R. ROTH, Jiamin LIU, Evrim TURKBEY a Ronald M. SUMMERS. A Bottom-up Approach for Pancreas Segmentation using Cascaded Superpixels and (Deep) Image Patch Labeling [online]. 2016 [cit. 2016-04-23]. Dostupné z: http://arxiv.org/abs/1505.06236

[26] ACHANTA, R., A. SHAJI, K. SMITH, A. LUCCHI, P. FUA a Sabine SÜSSTRUNK. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. IEEE Transactions on Pattern Analysis and Machine Intelligence [online]. 2012, 34(11), 2274-2282 [cit. 2016-04-23]. DOI: 10.1109/TPAMI.2012.120. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6205760

[27] Caffe [online]. Bekeley university [cit. 2016-04-23]. Dostupné z: http://caffe.berkeleyvision.org/

[28] POWERS, David. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation [online]. 2011, 2(1), 37-63 [cit. 2016-04-26]. ISSN : 2229-3981. Dostupné z:

http://dspace2.flinders.edu.au/xmlui/bitstream/handle/2328/27165/Powers%20Evaluation.pdf?sequence=1

[29] Caffe: Layer Catalogue [online]. [cit. 2016-04-26]. Dostupné z: http://caffe.berkeleyvision.org/tutorial/layers.html

[30] RadClass: Hounsfield units - scale of HU, CT numbers. Classifications, online calculators, and tables in radiology Classifications, online calculators, and tables in radiology [online]. 2013 [cit. 2016-04-29]. Dostupné z: http://radclass.mudr.org/content/hounsfield-units-scale-hu-ct-numbers

[31] LAFFERTY, John D., Andrew MCCALLUM a Fernando C. N. PEREIRA. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001) [online]. 2001, , 282-289 [cit. 2016-04-29]. ISSN ISBN:1-55860-778-1. Dostupné z: http://dl.acm.org/citation.cfm?id=655813

[32] Deeplearning: Theano [online]. University of Montreal, 2008 [cit. 2016-04-29]. Dostupné z: http://www.deeplearning.net/software/theano/

[33] Torch [online]. Facebook, Twitter, Google [cit. 2016-04-29]. Dostupné z: http://torch.ch/

[34] Veles [online]. Samsung [cit. 2016-04-29]. Dostupné z: https://velesnet.ml/

[35] Precision and Recall. Wikimedia [online]. uživatel wikipedia Walber, 2014 [cit. 2016-05-02]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/2/26/Precisionrecall.svg

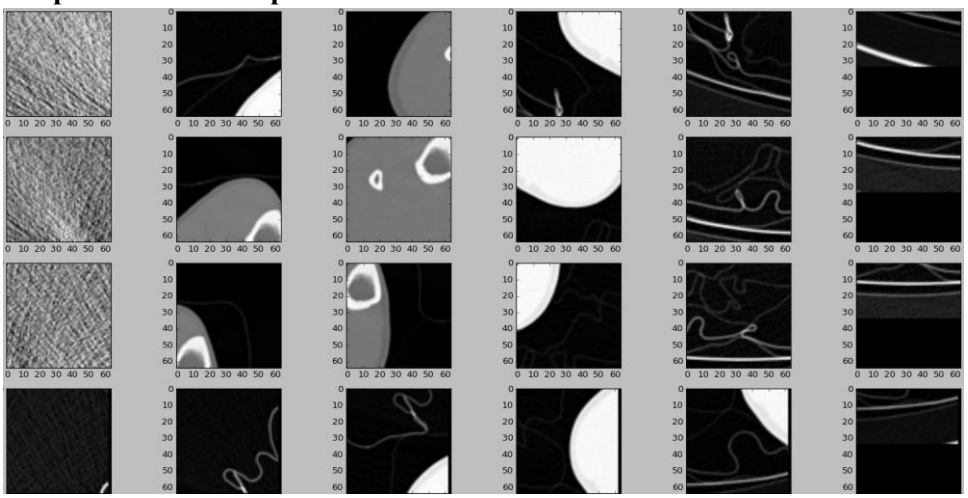
Seznam příloh

Příloha A. Výsledky experimentů

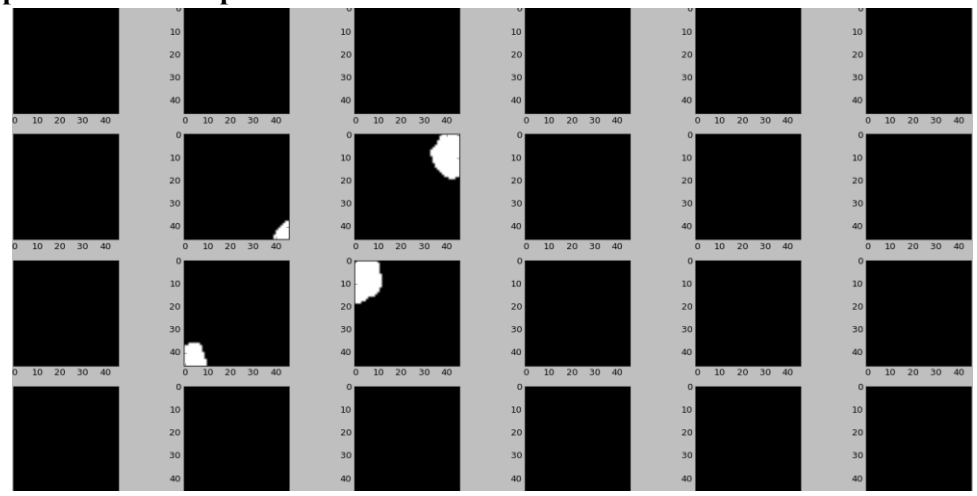
Výsledky experimentů	Podobrazy – 64x64	Podobrazy – 46x46	Celé obrazy
Konfigurace 1			
Neuronová síť	13x13 – 1024	13x13 – 1024	13x13 – 384
	3x3 – 1	3x3 – 1	3x3 – 1
Aktivační funkce	hyperbolický tangens	hyperbolický tangens	hyperbolický tangens
Koeficient učení	$5*10^{-6}$	$5*10^{-6}$	$5*10^{-6}$
F-measure	0,8845	0,9281	0,5226
Doba učení	40 minut	35 minut	3 hodiny
Konfigurace 2			
Neuronová síť	13x13 – 512	13x13 – 512	13x13 – 96
	5x5 – 1024	5x5 – 1024	5x5 – 128
	3x3 – 1	3x3 - 1	3x3 – 1
Aktivační funkce	hyperbolický tangens	hyperbolický tangens	hyperbolický tangens
Koeficient učení	$5*10^{-7}$	$5*10^{-7}$	10^{-7}
F-measure	0,9245	0,9396	0,5037
Doba učení	5 hodin	4 hodiny	15 hodin
Konfigurace 3			
Neuronová síť	13x13 – 32	13x13 – 32	13x13 – 32
	5x5 – 32	5x5 – 32	5x5 – 32
	3x3 – 1	3x3 – 1	3x3 – 1
Aktivační funkce	hyperbolický tangens	hyperbolický tangens	hyperbolický tangens
Koeficient učení	$5*10^{-7}$	$5*10^{-7}$	10^{-7}
F-measure	0,9216	0,94	0,4627
Doba učení	16 minut	11 minut	2 hodiny 45 minut
Konfigurace 4			
Neuronová síť	13x13 – 32	13x13 – 32	13x13 – 32
	5x5 – 32	5x5 – 32	5x5 – 32
	3x3 – 1	3x3 – 1	3x3 – 1
Aktivační funkce	ReLU	ReLU	ReLU
Koeficient učení	$5*10^{-7}$	$5*10^{-7}$	10^{-7}
F-measure	0,9239	0,9238	0,5088
Doba učení	16 minut	11 minut	2 hodiny 45 minut
Konfigurace 5			
Neuronová síť	13x13 – 32	13x13 – 32	13x13 – 32
	5x5 – 32	5x5 – 32	5x5 – 32
	3x3 – 1	3x3 – 1	3x3 – 1
Aktivační funkce	ReLU	ReLU	ReLU
Koeficient učení	$5*10^{-6}$	$5*10^{-6}$	10^{-6}
F-measure	0,93	0,951	0,6129
Doba učení	16 minut	11 minut	2 hodiny 45 minut

Příloha D. Snímky datové sady

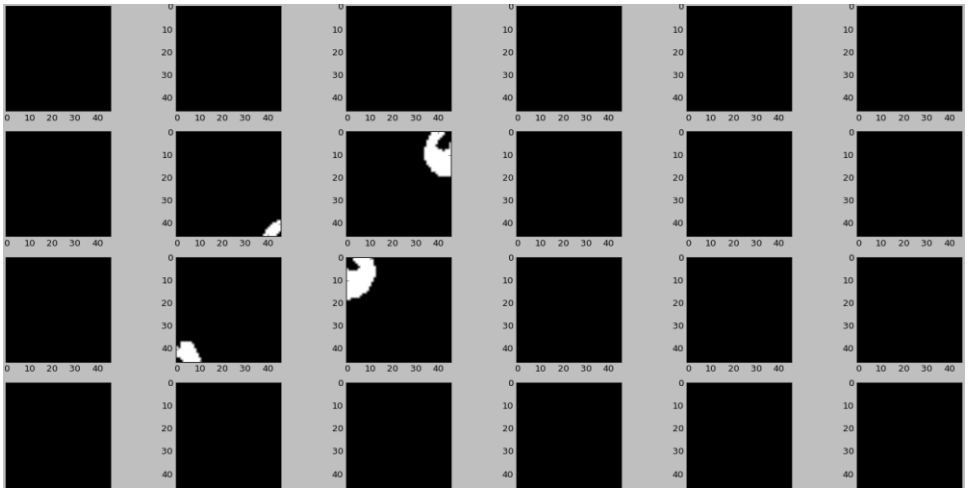
Vstupní data pro trénování na podobrazech – 650. vrstva



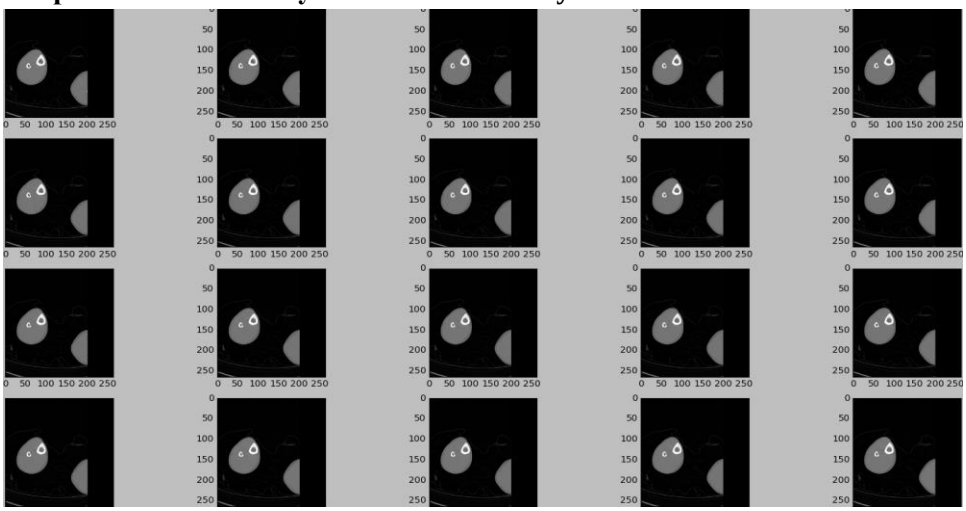
Anotace pro trénování na podobrazech – 650. vrstva



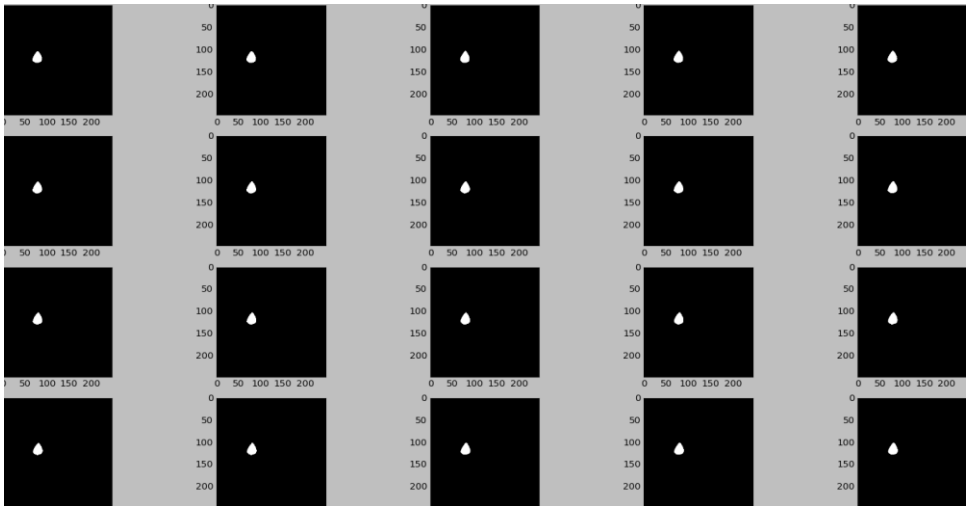
Výstup z neuronové sítě při trénování na podobrazech – 650. vrstva



Vstupní data při trénování na celých obrazech – vrstvy 487 až 497



Anotace pro trénování na celých obrazech – vrstvy 487 až 497



Výstup z neuronové sítě při trénování na celých obrazech – vrstvy 487 až 497