Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Lokalizace jaterních lézí v CT snímcích

Plzeň, 2018 **Miroslav Bulka**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Miroslav BULKA**
Osobní číslo:	**A16N0140P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Název tématu:	**Lokalizace jaterních lézí v CT snímcích**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování:

1. Prostudujte doporučenou literaturu týkající se předzpracování obrazu a jeho segmentace.
2. Zaměřte se především na zpracování obrazových dat v lékařství.
3. Otestujte vhodnost použití klasifikačního přístupu.
4. Otestujte vhodnost přístupu založeného na neuronových sítích.
5. Porovnejte výše zmíněné přístupy a eventuálně navrhněte vlastní.

Rozsah grafických prací: **dle potřeby**

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

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

— **Milan Šonka, Václav Hlaváč, Roger Boyle: Image Processing, Analysis and Machine Vision**

— **Miloš Železný: přednášky k předmětu ZDO**

— **Marek Hrúz: přednášky k předmětu MPV**

— **Milan Šonka, J. Michael Fitzpatrick: Handbook of Medical Imaging**

— **Isaac N. Bankman: Handbook of Medical Image Processing and Analysis**

Vedoucí diplomové práce: **Ing. Tomáš Ryba**
Nové technologie pro informační společnost

Datum zadání diplomové práce: **2. října 2017**

Termín odevzdání diplomové práce: **18. května 2018**

Doc. Dr. Ing. Vlasta Radová
děkanka

Západočeská univerzita v Plzni
L.S.
Fakulta aplikovaných věd

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 2. října 2017

Prohlášení

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

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

V Plzni dne 6. května 2018

.....

vlastnoruční podpis

Poděkování

V první řadě bych chtěl poděkovat Ing. Tomáši Rybovi, Ph.D. z oddělení umělé inteligence na Katedře kybernetiky Fakulty aplikovaných věd Západočeské univerzity v Plzni za skvělé uvedení do dané problematiky a za poskytnutí spousty cenných rad. Zvláštní poděkování patří Ing. Marku Hrúzovi, Ph.D. ze stejného oddělení za obrovskou ochotu seznámit mě s moderními metodami využití neuronových sítí v této úloze.

Tato práce vznikla za podpory projektů CERIT Scientific Cloud (LM2015085) a CESNET (LM2015042) financovaných z programu MŠMT Projekty velkých infrastruktur pro VaVaI.

Anotace

Cílem této práce je využít získaných znalostí z oblasti počítačového vidění a strojového učení k navržení metody pro automatickou lokalizaci jaterních lézí v CT snímcích. V této práci jsou navrženy dva přístupy jako řešení tohoto problému. První je založen na generování ohraničujících obdélníků, v nichž se nacházejí léze, s využitím HoG extraktoru a klasifikátoru SVM. Druhý přístup využívá konvoluční neuronové sítě pro sémantickou segmentaci jaterních lézí. Pro oba přístupy je vybrán nejlepší vyhodnocený model a nakonec jsou oba nejlepší modely mezi sebou porovnány. Na závěr jsou zmíněna dvě možná vylepšení do budoucna a uveden jeden příklad využití v praxi.

Klíčová slova

Histogram orientovaných gradientů, klasifikátor s podpůrnými vektory, konvoluční neuronové sítě, dekonvoluční síť, sémantická segmentace.

Annotation

The aim of this work is to use acquired knowledge of computer vision and machine learning to design a method for automatic localization of liver lesions in CT images. In this work two different approaches are proposed as a solution of this issue. The first one is based on generating of bounding boxes in which the lesions are present using HoG as a feature extractor and SVM as a classifier. The second approach uses convolutional neural networks for semantic segmentation of liver lesions. For both approaches the model with the best evaluated configuration is selected. Finally, both best models are compared with each other. At the end, two possible improvements to the future and one example of use in practice are mentioned.

Keywords

Histogram of oriented gradients, support vector machines, convolutional neural networks, deconvolution network, semantic segmentation.

Obsah

Kapitola 1

Úvod

Jednou z mnoha úloh v oblasti lékařské diagnostiky je hledání lézí v různých tkáních vyskytujících se v lidském těle. Jejich lokalizaci provádějí lékaři například na základě procházení jednotlivých řezů CT snímků, kde hledají útvary, které by lézím mohly odpovídat. Velice přínosné by bylo, kdyby takovýto proces byl automatizován a proveden mnohonásobně rychleji, a to ihned po pořízení snímku. Výsledky automatické detekce by pak totiž mohly usnadnit práci lékařům tím, že by je upozornily na objekty, které by potenciálně lézemi být mohly, a rovněž na malé artefakty, které by mohly být snadno přehlédnuty. Navrhnout takovouto automatickou metodu s využitím počítačového vidění je cílem této práce.

Jednou z nejznámějších úloh počítačového vidění je segmentace obrazu, která má široké využití také v oblasti medicíny. Konkrétní úlohou je například automatická segmentace jater v CT snímcích (viz [1]). Segmentace jater však bývá obvykle jen prostředkem pro další úlohu zpracování daných obrazových dat, jak je tomu i v této práci, kde jsou k dispozici CT snímky a pro ně výsledky segmentace jater. Úkolem je pak detekovat jaterní léze v těchto CT snímcích s využitím informace o rozložení jater v jednotlivých řezech. V práci je nejprve popsána lékařská zobrazovací metoda zvaná výpočetní tomografie (CT) a využití metod počítačového vidění v různých úlohách pracujících s daty pomocí ní získanými. Další kapitoly se již týkají použitých metod počítačového vidění, jejich využití v dané úloze a následných prováděných experimentů.

V této práci jsou zkoumány dva přístupy pro lokalizaci jaterních lézí v CT snímcích. Tím prvním je metoda hledání obdélníků ohraničujících detekované léze. Zde je využíváno sekvence operací, kterými jsou předzpracování obrazu, extrakce vektoru příznaků a rozhodnutí klasifikátoru, zda daný obdélník, pro jehož obsah byl tento vektor extrahován, obsahuje lézi, či nikoliv. Z metod předzpracování obrazu je věnována pozornost mediánové a bilaterální filtraci. Jako extraktor vektorů příznaků je využíván histogram orientovaných gradientů (HoG), jehož parametry jsou jedním z hlavních objektů testování. Jako klasifikátor, který má za úkol rozhodnout, zda daný vektor příznaků byl extrahován z regionu obsahujícího lézi či nikoliv, je v této práci využit lineární klasifikátor s podpůrnými vektory (SVM).

Druhým přístupem je sémantická segmentace obrazu pomocí konvolučních neuronových sítí. Nejprve je představen princip fungování neuronové sítě a jejího učení. Dále jsou

uvedeny různé aktivační funkce, optimalizační metody a ztrátové funkce. Z aktivačních funkcí jsou využívány ReLU a pro výstupní vrstvu sítě softmax. Využívanými optimalizátory jsou SGD, RMSprop a Adam. Následně jsou představeny konvoluční neuronové sítě jako silný nástroj v úlohách počítačového vidění a jsou uvedeny jejich nejdůležitější vrstvy, jako jsou konvoluční nebo max-pooling vrstva. Nakonec je představena architektura sítě kombinující konvoluční a za ní připojenou dekonvoluční síť, využívaná v práci [2] pro sémantickou segmentaci obrazu.

Další kapitola se zabývá použitými daty, kterými jsou řezy CT snímků. Je nutné poznamenat, že medicínská data byla poskytnuta Fakultní nemocnicí v Plzni v rámci smlouvy o spolupráci mezi FN Plzeň a ZČU v Plzni. Veškerá data byla anonymizována, tj. zbavena veškerých osobních a citlivých údajů (jméno, adresa, rodné číslo). Úloha předpokládá dostupnost binární masky jater, která je v obou přístupech využívána jiným způsobem. U přístupu využívajícího HoG a SVM je procházena pouze oblast jater. U konvolučních neuronových sítích je veškerá tkáň mimo játra obarvena na nulovou hodnotu. Pro zvětšení trénovací množiny jsou data navíc augmentována pomocí afinní transformace.

Další části práce již dokumentují všechny prováděné experimenty. Nejprve jsou prováděny experimenty za účelem získání optimální konfigurace modelu využívajícího přístup HoG a SVM, kde jsou testovány různé metody předzpracování obrazu a parametry HoG extraktoru. Pro vybraný nejlepší natrénovaný model se pak provádí ještě FROC analýza pro určení optimálního prahu minimální pravděpodobnosti, která musí být okénku přiřazena, aby došlo k závěru, že se jedná o lézi. Následně je zkoumán vliv metody Hard Negative Mining na úspěšnost klasifikace.

V následující části jsou zdokumentovány experimenty prováděné s využitím konvolučních neuronových sítí, kde je detailně popsán postup vytváření architektury sítě pro tuto úlohu a také návrh metriky, která je pro výběr nejlepšího modelu nejvhodnější. Testovány jsou různé architektury a pro ně rovněž různé optimalizační metody.

Dále jsou nejlepší modely obou přístupů porovnány podle stejné hodnotící metody, která byla použita při výběru optimální konfigurace přístupu HoG a SVM. Je zde především vykreslena křivka závislosti míry recall na hodnotě prahu minimálního nutného zastoupení léze v obdélníku pro oba modely.

Na závěr je uvedeno několik možných vylepšení do budoucna coby dalších směrů, kterými by se mohla rozšíření práce ubírat. Také je uveden jeden příklad praktické aplikace automatické detekce lézí, a to křivka procentuálního zastoupení lézí v jednotlivých řezech, která by lékařům mohla sloužit jako orientační náhled ještě před prozkoumáním jednotlivých řezů.

Kapitola 2

Výpočetní tomografie

Výpočetní tomografie neboli CT (z anglického Computed Tomography) je lékařská zobrazovací metoda založená na konstrukci anatomických vrstvových obrazů zvaných řezy, získaných z informací o absorpci záření v mnoha průmětech po obvodu kruhu, jak definují Ferda aj. v jejich práci [3], odkud jsou čerpány informace v celé této kapitole. Tato metoda způsobila zásadní přelom v lékařském zobrazování.

2.1 Princip CT

Konstrukci CT tvoří soustava rentgenky a proti ní ležícího oblouku, který je složen z několika stovek detektorů. Tato soustava, která je součástí tzv. gantry, se otáčí se kolem těla pacienta ležícího na stole, který zajíždí do otvoru ve středu gantry. Doba, za kterou je dokončena jedna rotace soustavy, se pohybuje v řádech desetin sekundy, přičemž záleží na typu přístroje a druhu vyšetření. Během této rotace stihne přístroj provést několik set expozic z různých úhlů, z nichž je následně zrekonstruován CT obraz.

2.2 Hounsfieldova stupnice

K vyjádření míry absorpce rentgenového záření slouží Hounsfieldova stupnice, ve které je míra absorpce udávána v tzv. Hounsfieldových jednotkách HU (z anglického Hounsfield Unit). Tato stupnice obsahuje zhruba 4000 hodnot od -1000 do $+3000$ HU, přičemž nulová hodnota odpovídá míře absorpce vody. Zde platí pravidlo, že čím vyšší je hodnota na stupnici, tím vyšší absorpci má prostředí. Jednotlivé hodnoty pak lze kódovat do stupňů šedi, kde světlejší body znamenají vyšší absorpci. Každý řez lze tedy vykreslit jako 2D šedotónový obrázek. Zobrazení, kde nabývají body v obrázku 4000 jasových hodnot, však není pro člověka příliš přínosné, jelikož lidské oko dokáže identifikovat jen asi 16 stupňů šedi. Místo toho se zobrazuje jen část denzitní škály neboli CT okno s definovanou šířkou a středem. Hodnoty, které se nacházejí pod dolní hranicí okna, se zobrazují jako černé a ty, které se vyskytují nad ní, jsou zobrazeny jako bílé. Použití CT okna pak umožňuje posoudit i malé rozdíly hodnot absorpce záření

2.3 Využití počítačového vidění ve výpočetní tomografii

Počítačové vidění má v oblasti medicíny velice široké uplatnění. Běžnými úlohami jsou například automatická segmentace jater v CT snímcích (viz [1]), která může mít využití například pro měření objemu v sekvenčních CT snímcích, jak je uváděno práci [4]. Po získání segmentovaných jater pak může být dalším úkolem automatická lokalizace jaterních lézí v nich obsažených. Tato práce má za úkol navrhnout mechanismus pro automatickou lokalizaci jaterních lézí v CT snímcích, přesněji řečeno v jejich 2D řezech, za předpokladu, že je k dispozici binární maska jater. Existuje řada způsobů, jak tuto úlohu řešit, z nichž dva byly v této práci zkoumány. Jedním z nich je pro každý řez generovat množinu obdélníků, které ohraničují jednotlivé jaterní léze v něm obsažené. Druhým způsobem je sémantická segmentace obrazu, tedy klasifikace každého pixelu obrazu, jak je popsáno v práci [2]. Takto segmentovaný obraz pak obsahuje informaci nejen o umístění lézí, ale také o jejich tvaru.

Kapitola 3

Předzpracování obrazu

Předzpracování obrazu bývá často jedním z prvních kroků v úlohách počítačového vidění. Předzpracování obrazu mohou být různého druhu podle typu úlohy a zpracovávaných dat. Těmi jsou v této práci CT snímky, konkrétně z nich vytažené 2D řezy. Pro tuto úlohu se ukázalo jako potřebné především odstranění šumu, čehož lze docílit filtrací obrazu, která bývá realizována konvolucí obrazu a daného konvolučního jádra neboli filtru. Dalším požadavkem v této úloze je zachování hran obrazu i po odstranění šumu. Existují metody, které i tuto podmínku splňují. V této práci byly testovány dvě z nich, a sice mediánový filtr a bilaterální filtr.

3.1 Mediánový filtr

Mediánovým filtrem se zabývá i Šonka ve své práci [5], kde popisuje vyhlazování pomocí mediánového filtru tak, že je jasová hodnota každého pixelu nahrazena hodnotou mediánu jasů všech pixelů v jeho okolí. Tím dochází k redukci nežádoucího vyhlazování hran. Hodnota mediánu jasů v okolí navíc není ovlivněna izolovanými pixely s výrazně vyšší hodnotou jasu. Parametrem této metody jsou rozměry mediánového filtru, tedy velikost okolí, ze kterého je medián počítán. V této práci byly využívány čtvercové filtry, tudíž parametrem metody byl pouze rozměr jedné strany tohoto čtvercového filtru. Výsledek vyhlazení obrazu pomocí mediánového filtru o různých velikostech je k vidění na obrázku 3.1.

(a) (b) (c)

Obrázek 3.1: Zleva: originální snímek, výsledek po aplikaci mediánového filtru o velikosti okna 5 pixelů a dále o velikosti 15 pixelů.

3.2 Bilaterální filtr

Dalším filtrem zachovávajícím hrany je bilaterální filtr uváděný v práci [6], odkud byly čerpány informace v této kapitole. K zachování hran se zde používá nelineární kombinace hodnot pixelů v okolí. Hlavním principem je využití nejen geometrické blízkosti pixelů, ale také jejich fotometrické podobnosti, přičemž jsou v obou těchto oblastech preferovány bližší hodnoty před těmi vzdálenými. Běžné filtrační metody využívají pouze geometrickou blízkost pixelů, a to tak, že jejich hodnoty váží koeficienty, jejichž hodnoty se s rostoucí vzdáleností od centrálního pixelu snižují. Bilaterální filtr využívá navíc filtraci, která průměruje hodnoty jasů s váhami, které se s klesající podobností s hodnotou centrálního pixelu snižují. Tento filtr je díky tomu, že váhy v něm obsažené závisí na intenzitě obrazu, nelineární. Filtrace využívající obou přístupů, tedy geometrické blízkosti a jasové podobnosti, se nazývá bilaterální filtrace. Ta na pozici centrálního pixelu ukládá průměr hodnot podobných a geometricky blízkých pixelů. Výsledná hodnota by neměla být ovlivněna pixely s příliš odlišnými hodnotami jasu oproti centrálnímu pixelu.

Bilaterální filtraci lze popsat jako kombinaci dvou typů filtrace pomocí vztahu:

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi)\, c(\xi, \mathbf{x})\, s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))\, d\xi, \tag{3.1}$$

kde $\mathbf{f}(\mathbf{x})$ je hodnota jasové funkce v daném pixelu, $c(\xi, \mathbf{x})$ je geometrická blízkost mezi centrálním bodem $\mathbf{x}$ a blízkým bodem ξ a $s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))$ značí fotometrickou podobnost mezi centrálním bodem $\mathbf{x}$ a blízkým bodem ξ. Hodnota $k(\mathbf{x})$ je dána vztahem:

$$k(\mathbf{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, \mathbf{x})\, s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))\, d\xi. \tag{3.2}$$

Jednoduchým případem bilaterální filtrace je gaussovská filtrace, kde jsou funkce $c(\xi, \mathbf{x})$ i $s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))$ gaussovskými funkcemi euklidovských vzdáleností svých argumentů:

$$c(\xi, \mathbf{x}) = e^{-\frac{1}{2}\left(\frac{d(\xi, \mathbf{x})}{\sigma_d}\right)^2}, \tag{3.3}$$

$$s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) = e^{-\frac{1}{2}\left(\frac{\delta(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))}{\sigma_r}\right)^2}, \tag{3.4}$$

kde σ_d značí geometrický rozsah a σ_r fotometrický rozsah. Oba tyto rozsahy jsou parametry metody bilaterální filtrace. Funkce d a δ mají následující předpis:

$$d(\xi, \mathbf{x})) = \|\xi - \mathbf{x}\|, \tag{3.5}$$

$$\delta(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) = \|\mathbf{f}(\xi) - \mathbf{f}(\mathbf{x})\|. \tag{3.6}$$

Vyšší σ_d obraz více rozmazává, jelikož kombinuje hodnoty jasů ze vzdálenějších oblastí. Příliš vysoká hodnota σ_r může rozmazat i hrany obrázku, což je nežádoucí efekt. Je důležité ji tedy nastavit tak, aby hrany zůstaly zachovány a zároveň došlo k vyhlazení homogenních oblastí v obrazu zatížených šumem. V implementaci OpenCV [7] lze nastavit také velikost filtru ve smyslu geometrické blízkosti, tedy velikost oblasti, ve které se má výše zmíněné průměrování provádět.

Na obrázku 3.2 je k vidění výsledek aplikace bilaterálního filtru s různě nastavenými parametry na originální snímek.

(a) (b) (c)

Obrázek 3.2: Vlevo je originální snímek, uprostřed výsledek po aplikaci bilaterálního filtru s velikostí okna 15 pixelů a s parametry $\sigma_r = 15,\ \sigma_d = 15$. Vpravo je pak k vidění výsledek bilaterální filtrace se stejnou velikostí filtru a s parametry $\sigma_r = 100,\ \sigma_d = 100$.

Kapitola 4

Klasifikace obrazu

Jak se uvádí v práci [5], odkud jsou čerpány následující informace, je objekt v obrázku obvykle reprezentován jako region v segmentovaném obraze. Jednotlivé objekty mohou být rozděleny do menších podmnožin zvaných třídy, ve kterých se jednotlivé prvky vyznačují podobnými vlastnostmi. Každý objekt je reprezentován svým obrazem neboli vektorem příznaků $\mathbf{x} = \{x_1, ..., x_n\}$, které numericky popisují různé vlastnosti objektu. Zařazování příznakových vektorů objektů do jednotlivých tříd je úkolem tzv. klasifikátoru. Množina všech možných vektorů příznaků se nazývá příznakový prostor, v němž si jsou obrazy, které patří do stejné třídy, navzájem blízké. Po zanesení všech obrazů do příznakového prostoru se vytvářejí shluky odpovídající daným třídám. Tyto shluky obrazů mohou být od sebe odděleny rozdělující hyperplochou, kterou lze popsat tzv. diskriminační funkcí. Podle hodnoty diskriminační funkce po přivedení jednoho z obrazů na její vstup se pak rozhodne o zařazení tohoto pixelu do jedné ze tříd. V případě, že jsou data lineárně separabilní, se hovoří o lineární diskriminační funkci, která popisuje rozdělující nadrovinu. Pokud jsou všechny diskriminační funkce klasifikátoru lineární, jedná se o lineární klasifikátor.

K nastavení optimálních parametrů klasifikátoru se používá proces učení, během kterého jsou klasifikátoru předkládány jednotlivé obrazy a informace od učitele o jejich správném zařazení do některé ze tříd. Klasifikátor se pak snaží nastavovat své parametry tak, aby minimalizoval předem stanovené kritérium optimality, kterým může být například ztráta způsobená jeho špatnými rozhodnutími. Informace o správném zařazení obrazu do třídy se nazývá informace od učitele a pokud je k dispozici, jedná se o proces učení s učitelem. Cílem trénování je nastavit parametry klasifikátoru tak, aby byl schopen rozpoznat i objekty, se kterými se předtím nesetkal. Pokud klasifikátor dosahuje skvělých výsledků na trénovací množině, ale naopak špatných výsledků na dosud neviděných datech, znamená to, že došlo k jeho přetrénování. Tímto problémem a zároveň způsobem, jak jej eliminovat, se dále zabývá kapitola 6.3.3.

4.1 Klasifikátor s podpůrnými vektory

SVM (z anglického Suport Vector Machines) neboli metoda podpůrných vektorů je velmi dobře shrnuta v práci [5]. V úloze klasifikace do dvou tříd se tato metoda snaží docílit optimální klasifikace tím, že je maximalizována šířka pásma necitlivosti čili vzdálenost od rozdělující nadroviny k nejbližším trénovacím obrazům, které jsou označovány jako podpůrné vektory. Podpůrné vektory tedy určují diskriminační funkci.

V dalších odstavcích bude uvažován jednoduchý případ klasifikace do dvou tříd s lineárně separabilními daty, kterými je množina n-dimenzionálních vektorů příznaků $\mathbf{x}$, k nimž jsou přiřazeny identifikátory ω, obsahující informaci od učitele o tom, do jaké třídy patří. Rozdělující nadrovina má tvar:

$$\mathbf{w} \cdot \mathbf{x} + b = 0. \tag{4.1}$$

Pro maximalizaci šířky pásma necitlivosti jsou zavedeny další dvě paralelní nadroviny:

$$\mathbf{w} \cdot \mathbf{x} + b = 1, \qquad \mathbf{w} \cdot \mathbf{x} + b = -1, \tag{4.2}$$

které procházejí podpůrnými vektory a platí, že mezi nimi neleží žádné trénovací obrazy. Pro všechny trénovací obrazy $\mathbf{x}_i$ musí být tedy ještě splněna podmínka:

$$\omega_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1, \tag{4.3}$$

kde se předpokládá, že $\omega \in \{-1, 1\}$. Vzdálenost mezi nadrovinami popsanými v rovnici 4.2 se spočítá následovně:

$$d = \frac{2}{\|\mathbf{w}\|}. \tag{4.4}$$

Aby byla maximalizována šířka pásma necitlivosti, je zapotřebí minimalizovat $\|\mathbf{w}\|$ za dodržení podmínky 4.3. Rozdělující nadrovina pak leží mezi nadrovinami danými vztahem 4.2.

Optimalizační proces je následně zjednodušen na hledání optimální rozdělující nadroviny dané takovými parametry $\mathbf{w}$ a b, při kterých dojde k minimalizaci Lagrangeovy funkce definované vztahem:

$$L\left(\mathbf{w}, b, \alpha\right) = \frac{1}{2}\|\mathbf{w}\| - \sum_{i=1}^{N} \alpha_i \omega_i \left(\mathbf{w}_i \cdot \mathbf{x}_i + b\right) + \sum_{i=1}^{N} \alpha_i, \tag{4.5}$$

kde α_i jsou Lagrangeovy koeficienty, pro které platí omezení, že $\alpha_i \geq 0$. Výpočtem parciálních derivací funkce $L\left(\mathbf{w}, b, \alpha\right)$ vzhledem k $\mathbf{w}$ a b a lze získat vztahy:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \omega_i \mathbf{x}_i, \tag{4.6}$$

$$\sum_{i=1}^{N} \alpha_i \omega_i = 0. \tag{4.7}$$

Po dosazení vztahů 4.6 a 4.7 do původní Lagrangeovy funkce 4.5 vznikne duální formulace:

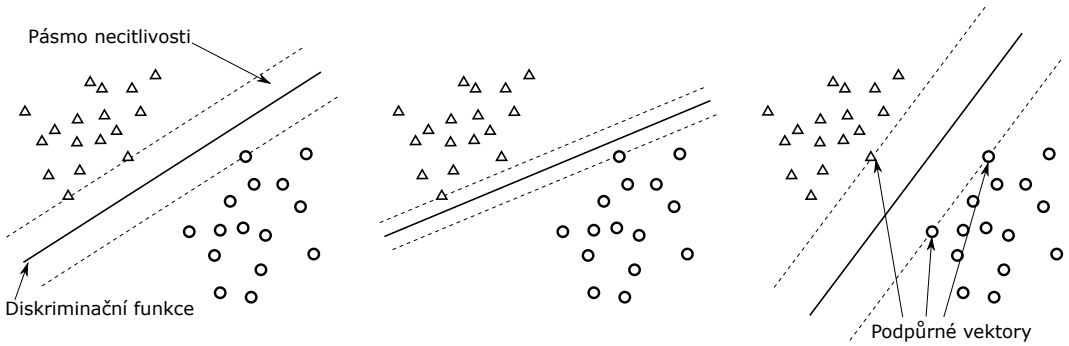
$$L\left(\mathbf{w}, b, \alpha\right) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \omega_i \omega_j \left(\mathbf{x}_i \cdot \mathbf{x}_j\right). \tag{4.8}$$

Optimalizace pak spočívá v hledání takových koeficientů α_i, které maximalizují hodnotu funkce 4.8. Parametr b lze dopočítat, jakmile je dokončena optimalizace. Získané parametry $\mathbf{w}$ a b jsou poté dosazeny do rovnice diskriminační funkce:

$$f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i + b, \tag{4.9}$$

která nabývá nulové hodnoty, pokud obraz $\mathbf{x}_i$ leží přímo na rozdělující nadrovině. Podle toho, zda je hodnota diskriminační funkce pro daný vstupní obraz kladná, nebo záporná, se rozhodne o zařazení obrazu do jedné ze tříd.

Jednoduchá ukázka klasifikace dvoudimenzionálních dat do dvou tříd je k vidění na obrázku 4.1, kde jsou zobrazeny dva případy neoptimální lineární diskriminační funkce a poté optimální řešení maximalizující šířku pásma necitlivosti mezi obrazy dvou tříd.



Obrázek 4.1: Porovnání různých voleb lineární diskriminační funkce pro stejná trénovací data. Vlevo a uprostřed jsou navrženy diskriminační funkce, které nejsou z hlediska maximalizace šířky pásma necitlivosti optimální. Vpravo je pak vykreslena optimální lineární diskriminační funkce, která maximalizuje šířku pásma necitlivosti a je určena vyznačenými podpůrnými vektory. Obrázek byl přejat z práce [5] a upraven.

Jedním z častých problémů SVM je, že trénovací data nejsou lineárně separabilní. Tento problém lze řešit pomocí takzvané kernelové metody, která spočívá v produkci nelineárně transformovaného příznakového prostoru, ve kterém již lineární rozdělující nadrovina být stanovena může.

4.2 Analýza hlavních komponent

Jak dále vyplývá z kapitoly 5, extrahované vektory příznaků, tedy histogramy orientovaných gradientů, mohou nabývat i více než deseti tisíců prvků. Trénovací data mohou tím pádem nabývat značných rozměrů. Lze však předpokládat, že mnoho prvků těchto vektorů nebude pro úlohu klasifikace přínosných například proto, že budou nabývat stejných hodnot pro všechna trénovací data. Nabízí se tak možnost pokusit se výrazně redukovat dimenzi dat, aniž by došlo ke kritické ztrátě informace.

Analýza hlavních komponent neboli PCA (z anglického Principal Component Analysis) je velice často používaná metoda pro redukci dimenzionality úlohy. Bishop ve své práci [8]

definuje PCA jako ortogonální projekci dat do lineárního prostoru nižší dimenze, která maximalizuje varianci výsledných dat.

Je tedy vyvíjena snaha o projekci dat o dimenzionalitě D do prostoru dimenze M, přičemž $M < D$. Dimenze M je parametrem metody. Prvním krokem je spočtení střední hodnoty dat:

$$\overline{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \tag{4.10}$$

a následně jejich kovarianční matice:

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \overline{\mathbf{x}}) (\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}, \tag{4.11}$$

kde $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ jsou jednotlivá data v původním prostoru dimenze D. Dalším krokem je výpočet vlastních čísel a s nimi spjatých vlastních vektorů matice $\mathbf{S}$. Pokud je zadána požadovaná dimenze M, je následně vytaženo M vlastních vektorů odpovídajících největším vlastním číslům matice $\mathbf{S}$. Vlastní vektor spojený s nejvyšším vlastním číslem se nazývá první hlavní komponenta.

Jak se uvádí v práci [9], projekce, kterou PCA hledá, je ortogonální lineární transformací dat, která je dána vztahem:

$$\mathbf{z} = \mathbf{W}^{\mathrm{T}} \mathbf{x}, \tag{4.12}$$

kde $\mathbf{z}$ je získaná projekce původních dat $\mathbf{x}$ do prostoru nižší dimenze M a $\mathbf{W}$ je matice tvořená M vlastními vektory odpovídajícími největším vlastním číslům matice $\mathbf{S}$.

Implementace v knihovně scikit-learn [10] nabízí rovněž možnost místo přímého nastavení požadované dimenze M vybrat jen takový počet komponent, aby nabývala výsledná transformovaná data alespoň určité hodnoty variance. Konkrétně se nastavuje procentuální hodnota, která značí, jaká musí být variance transformovaných dat v porovnání s variancí dat původních. Pro experimenty v této práci byla nastavena jednotná hodnota, a to 90%. Pokud by totiž byla zvolena pevná hodnota požadované dimenze, byla by v experimentech s daty vyšších dimenzí zachovávána nižší variance, čímž by byly tyto experimenty znevýhodněny.

Jak je zmíněno v publikaci [9], velmi důležitou vlastností metody PCA je, že výsledné prvky získané reprezentace navíc nejsou vzájemně korelovány. Stojí však za zmínku, že tato metoda vyžaduje výpočet kovarianční matice o rozměrech $D \times D$, což znamená, že s rostoucím D lze předpokládat také rostoucí paměťové nároky a dobu výpočtu, jak uvádí Bishop ve své práci [8].

Kapitola 5

Extrakce vektorů příznaků

Jak se uvádí v kapitole 4, trénovací data jsou klasifikátoru předávána ve formě vektorů příznaků. V úlohách počítačového vidění jsou však k dispozici obrazová data, ze kterých je potřeba vektory příznaků nejdříve vhodným způsobem extrahovat.

5.1 Histogram orientovaných gradientů

Velice rozšířeným způsobem extrakce vektoru příznaků je metoda histogramu orientovaných gradientů neboli HoG, uváděná v práci [11], kde byla použita pro detekci lidských postav v obraze. Základní myšlenkou této metody je, že tvar objektu může být dobře charakterizován rozdělením lokálních gradientů nebo směrů hran. Postup je takový, že obraz je rozdělen na malé regiony zvané buňky, uvnitř kterých je spočten histogram orientací všech pixelů, které tyto jednotlivé buňky obsahují. Dále je dosaženo nezávislosti na osvětlení pomocí normalizace lokální odezvy. Toho lze dosáhnout akumulací míry energie lokálního histogramu v poněkud větší oblasti, než je velikost jedné buňky, a normalizací všech buněk této větší oblasti, která se nazývá blok. Jak se uvádí v práci [12], kde byla metoda HoG použita v úloze rozpoznávání obličejů, kombinace těchto histogramů pak reprezentuje finální HoG deskriptor.

Metoda extrakce vektorů příznaků pomocí histogramu orientovaných gradientů byla implementována také knihovně OpenCV [7], kde byly během experimentů nastavovány tři parametry této metody:

- **počet orientací**, tedy počet prvků histogramu v jedné buňce,
- **velikost buňky** neboli rozměry buněk udávané v pixelech,
- **počet buněk v bloku** čili rozměry, které značí, přes kolik buněk se jeden blok rozprostírá v obou osách.

Pro jednoduchost byly v této práci jak buňky, tak i bloky nastavovány jako čtvercové. Bylo tedy zapotřebí nastavit pouze velikost jedné z hran buňky, resp. bloku. Velikost buňky bude dále značena jako PPC (z anglického Pixels Per Cell) a pokud její hodnota bude rovna n, bude to znamenat, že buňka nabývá rozměrů $n \times n$ pixelů. Počet buněk

v bloku bude značen dále jako CPB (z anglického Cells Per Block) a pokud jeho hodnota bude rovna m, bude to znamenat, že blok zabírá oblast o velikosti $m \times m$ buněk.

Jak je možné si odvodit, všechny tři zmíněné parametry metody mají zásadní vliv na velikost extrahovaného vektoru příznaků. Ta pak během experimentů výrazně ovlivňovala paměťové nároky. Například vektor příznaků extrahovaný metodou HoG z jednoho obrázku o velikosti 48×48 pixelů při nastavení počtu orientací na 12 a parametrů $\text{PPC} = 4$ a $\text{CPB} = 3$ dosahuje délky $L = 10800$ příznaků.

Kapitola 6

Neuronové sítě

Neuronové sítě, jak popisuje Šonka ve své práci [5], reprezentují silný nástroj v úlohách rozpoznávání obrazu. Většina přístupů je založena na kombinaci elementárních procesorů neboli neuronů, z nichž každý má několik vstupů a generuje jeden výstup, který je funkcí váženého součtu všech vstupů neuronu, přičemž každý vstup má vlastní přiřazenou váhu. Jednoduchý McCullochův-Pittsův model neuronu byl popsán v práci [13]. Do neuronu jsou přivedeny vstupy v_i s váhami w_i. Výstup neuronu je pak dán vztahem:

$$x = \sum_{i=1}^{n} v_i w_i - \theta, \tag{6.1}$$

kde θ je práh spjatý s daným neuronem a n je počet vstupů neuronu. K neuronu je rovněž přiřazena aktivační funkce, která určuje hodnotu na výstupu. Některými používanými aktivační funkcemi se dále zabývá kapitola 6.2.

Hlavní myšlenkou neuronových sítí je vzájemné propojení jejích neuronů, což znamená, že výstup jednoho neuronu je jedním ze vstupů jiných neuronů. V úloze klasifikace mohou být neuronové sítě využity tak, že výstupní vektor sítě udává informaci o zařazení vstupního obrazu do jedné ze tříd.

Perceptron

Jak je uvedeno v práci [14], perceptron, kterému je věnována pozornost v publikaci [15], je jednovrstvá neuronová síť s aktivační funkcí, za kterou je nejčastěji volena funkce:

$$g(a) = \begin{cases} -1 & \text{pro } a < 0 \\ +1 & \text{pro } a \geq 0. \end{cases} \tag{6.2}$$

Výstup perceptronu s M vstupy x_i, M váhami w_i, kde $i = 1, ..., M$, prahem w_0 a aktivační funkcí $g(\cdot)$ je pak dán vztahem, který byl pro účely této práce upraven následovně:

$$y = g\left(\sum_{i=1}^{M} w_i x_i + w_0\right). \tag{6.3}$$

6.1 Dopředné sítě

Jak uvádí Bishop ve své práci [8], velice úspěšným modelem pro rozpoznávání obrazu je dopředná neuronová síť známá jako vícevrstvý perceptron. Základní model neuronové sítě může být popsán sérií funkcionálních transformací. Výstup j-tého neuronu se spočte podle vztahu:

$$z_j = h\left(a_j\right), \tag{6.4}$$

kde $j = 1, ..., M$, $h\left(\cdot\right)$ je aktivační funkce a a_j je vážený součet vstupů j-tého neuronu zvaný aktivace, který je dán vztahem:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \tag{6.5}$$

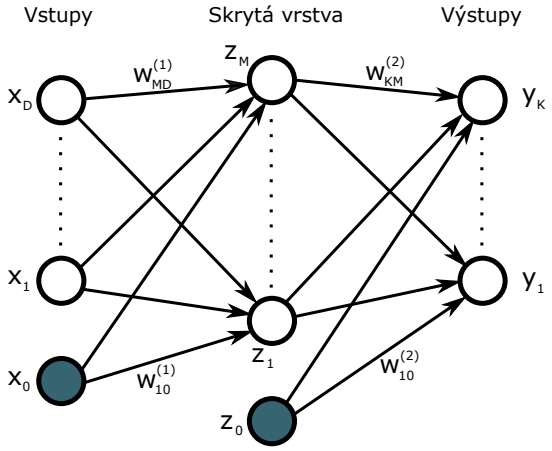
kde $w_{ji}^{(1)}$ jsou váhy, $w_{j0}^{(1)}$ značí práh (anglicky bias) a D je počet vstupů x_i daného neuronu. Horní index (1) znamená, že se jedná o první vrstvu v síti. Parametry prahu mohou být absorbovány do množiny váhových parametrů, což vede k upravenému vztahu:

$$a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i, \tag{6.6}$$

kde je třeba definovat přídavnou složku $x_0 = 1$. Pokud je přidána druhá vrstva sítě, jsou výstupy první vrstvy z_j jejími vstupy. Pak pro výstup k-tého neuronu druhé vrstvy platí:

$$y_k\left(\mathbf{x}, \mathbf{w}\right) = h^{(2)}\left(\sum_{j=0}^{M} w_{kj}^{(2)} h^{(1)}\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right), \tag{6.7}$$

kde $h^{(1)}\left(\cdot\right)$, resp. $h^{(2)}\left(\cdot\right)$, je aktivační funkce první, resp. druhé, vrstvy, $w_{ji}^{(1)}$, resp. $w_{kj}^{(2)}$, jsou váhy spjaté s první, resp. druhou, vrstvou, D je počet vstupů první vrstvy a M je počet jejích výstupů, tedy vstupů druhé vrstvy. Takováto neuronová síť se nazývá vícevrstvý perceptron a je funkcí řízenou vektorem parametrů $\mathbf{w}$, jejímž vstupem je $\mathbf{x}$ a výstupem $\mathbf{y}$. Její schéma by pak mohlo být ilustrováno tak, jak je k vidění na obrázku 6.1.



Obrázek 6.1: Schéma dvouvrstvé neuronové sítě. Převzato z [8] a upraveno.

Cílem trénování neuronové sítě je nastavení optimálního váhového vektoru $\mathbf{w}$ tak, aby byla minimalizována předem definovaná ztrátová funkce. Jak učení neuronových sítí probíhá, je dále popsáno v kapitole 6.3.

6.2 Aktivační funkce

Jak popisuje Bishop ve své práci [14], aktivační funkce může být interpretována jako nelineární funkce, která transformuje lineární kombinaci vstupů neuronu, přičemž koeficienty této lineární kombinace jsou parametry modelu. Aktivační funkce bývá obecně navrhována jako monotónní. Jeden z nejjednodušších typů aktivační funkce ve tvaru 6.2 byl již představen výše. Jak se uvádí v práci [16], standardní aktivační funkcí neuronu je:

$$f(x) = \tanh(x), \tag{6.8}$$

nebo například:

$$f(x) = \left(1 + e^{-x}\right)^{-1}. \tag{6.9}$$

Aktivační funkcí ve tvaru 6.9, která se nazývá logistická sigmoidální aktivační funkce a mapuje vstupy z intervalu $(-\infty, \infty)$ na výstupy nabývající hodnot z intervalu $(0, 1)$, se zabývá i Bishop ve své práci [14].

ReLU

Aktivační funkci ReLU (z anglického Rectifier Linear Unit [17]) je věnována pozornost například v práci [18], kde je rovněž uveden její předpis:

$$f(x) = \max(0, x). \tag{6.10}$$

Tvar derivace funkce ReLU lze jednoduše odvodit:

$$f(x) = \max(0, x) = \begin{cases} x & \text{pro } x < 0 \\ 0 & \text{pro } x \geq 0 \end{cases} \Rightarrow f'(x) = \begin{cases} 1 & \text{pro } x < 0 \\ 0 & \text{pro } x \geq 0 \end{cases}. \tag{6.11}$$

Jak je uvedeno v práci [16], využití ReLU v hlubokých neuronových sítích vede k sedmkrát rychlejšímu učení než v případě aktivačních funkcí 6.8 a 6.9.

Softmax

Jak je uvedeno v publikaci [9], funkce softmax se hodí zejména v případech, kde je snaha o reprezentaci pravděpodobnostního rozdělení diskrétní proměnné nabývající n možných hodnot. Tohoto přístupu je využíváno zejména na výstupu klasifikátorů k reprezentaci rozložení pravděpodobnosti pro n odlišných tříd. Je tedy snahou získat vektor $\widehat{\mathbf{y}}$, jehož prvky $\widehat{y}_i = P(y = i|\mathbf{x})$ by představovaly hodnoty pravděpodobností, že pokud je na vstupu obraz $\mathbf{x}$, bude klasifikován do třídy i. Hodnoty těchto pravděpodobností tvořících vektor $\widehat{\mathbf{y}}$ lze získat pomocí softmax funkce, která je dána vztahem:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}, \tag{6.12}$$

kde $\mathbf{z}$ je vektor výstupů n neuronů dané vrstvy před aplikováním aktivační funkce.

6.3 Učení neuronové sítě

K učení neuronových sítí se používá nejčastěji algoritmus zpětného šíření [19], který byl stručně shrnut v práci [5] a detailně popsán v práci [14]. Algoritmus spočívá v porovnávání skutečného výstupu, kterým síť reaguje na vstupní data, s očekávaným, který představuje informaci od učitele. Na základě rozdílu mezi těmito výstupy se spočte hodnota ztrátové funkce, podle které dojde k následné aktualizaci vah neuronů v síti podle vzorce:

$$w_{ij}\left(k+1\right) = w_{ij}\left(k\right) - \epsilon\frac{\partial E}{\partial w_{ij}}, \tag{6.13}$$

kde k značí iterační krok, ϵ je konstanta učení, případně označovaná jako míra učení, a E je ztrátová funkce.

6.3.1 Stochastic Gradient Descent

Algoritmu Stochastic Gradient Descent neboli SGD a jeho konvergenci je věnována pozornost v práci [20], odkud byly čerpány následující informace.

Předpokládejme, že $\widehat{y}$ je výstupem modelu, jestliže je na jeho vstup přiveden obraz x. Pak $\ell\left(\widehat{y}, y\right)$, kde y je požadovaný výstup pro obraz x, značí hodnotu ztrátové funkce. Cílem je najít takovou funkci modelu $f_w\left(x\right)$, parametrizovanou váhovým vektorem w, aby byla minimalizována ztráta:

$$Q\left(z, w\right) = \ell\left(f_w\left(x\right), y\right), \tag{6.14}$$

která je zprůměrována přes všechny vzorky a kde $z = (x, y)$ je označení pro dvojici tvořenou vstupem a příslušným požadovaným výstupem. Cílem algoritmu GD (z anglického Gradient Descent) je minimalizovat hodnotu empirického rizika, daného vztahem:

$$E_n\left(f\right) = \frac{1}{n}\sum_{i=1}^{n}\ell\left(f\left(x_i\right), y_i\right), \tag{6.15}$$

kde n je počet dat v dávce. Aktualizace parametrů podle algoritmu Gradient Descent probíhá podle předpisu:

$$w_{t+1} = w_t - \epsilon\frac{1}{n}\sum_{i=1}^{n}\nabla_w Q\left(z_i, w_t\right), \tag{6.16}$$

kde ∇_w značí parciální derivaci Q podle w, dolní index t značí daný krok a z_i je označení pro i-tou dvojici (x_i, y_i).

Algoritmus SGD (z anglického Stochastic Gradient Descent) je oproti GD, uvedenému v předešlém odstavci, značně zjednodušen. Místo exaktního výpočtu gradientu je v každé iteraci jeho hodnota odhadována na základě jednoho náhodně vybraného vzorku z_t a aktualizace parametrů se řídí vztahem:

$$w_{t+1} = w_t - \epsilon\nabla_w Q\left(z_t, w_t\right), \tag{6.17}$$

kde není potřeba si pamatovat, které vzorky byly procházeny v předešlých iteracích.

Algoritmus SGD je uváděn také v práci [16], kde se používá navíc ještě momentová metoda. Předpis pro aktualizaci vah byl z této práce přejat a upraven tak, aby jednotlivá značení byla konzistentní se vzorci ve zbytku této kapitoly. Každá váha w je pak aktualizována podle následujícího předpisu:

$$v_{t+1} = \alpha \cdot v_t - \beta \cdot \epsilon \cdot w_t - \epsilon \cdot \left\langle \frac{\partial E}{\partial w} \middle| w_t \right\rangle_{D_t}, \tag{6.18}$$

$$w_{t+1} = w_t + v_{t+1}, \tag{6.19}$$

kde t je iterační krok, ϵ je konstanta učení, v je momentová proměnná, α je momentová konstanta, β je hodnota úpadku vah, E je ztrátová funkce a $\left\langle \frac{\partial E}{\partial w} \middle| w_t \right\rangle_{D_t}$ je průměr její derivace vzhledem k w přes t-tou dávku D_t, vyhodnocený pro w_t.

Dávka D_t obvykle v optimalizačních algoritmech strojového učení nezahrnuje celou trénovací množinu, ale jen její malou podmnožinu. Proto se také tyto algoritmy někdy označují jako algoritmy s malou dávkou.

V práci [9] je zmiňována také jiná varianta momentové metody, která se nazývá Nesterovův moment, kde probíhá aktualizace váhy w následovně:

$$\widetilde{w}_t = w_t + \alpha \cdot v_t, \tag{6.20}$$

$$v_{t+1} = \alpha \cdot v_t - \beta \cdot \epsilon \cdot \widetilde{w}_t - \epsilon \cdot \left\langle \frac{\partial E}{\partial \widetilde{w}} \middle| \widetilde{w}_t \right\rangle_{D_t}, \tag{6.21}$$

$$w_{t+1} = w_t + v_{t+1}, \tag{6.22}$$

kde jednotlivé proměnné mají stejný význam jako v rovnicích 6.18 a 6.19 s tím rozdílem, že místo váhy w je použita proměnná $\widetilde{w}$. Tato metoda byla inspirována zrychlenou gradientní metodou, kterou uvádí Nesterov ve své práci [21].

6.3.2 Algoritmy s adaptivní konstantou učení

Konstanta učení, jak se uvádí v práci [9], je parametrem, který významně ovlivňuje práci modelu, který je často velice citlivý na změnu některých parametrů a naopak na změnu jiných parametrů nikoliv. Jako řešení se nabízí použít samostatnou konstantu učení pro aktualizaci každého parametru a tuto konstantu navíc v průběhu trénování také upravovat. V následujících odstavcích jsou uvedeny některé základní optimalizační metody s adaptivní konstantou učení, přičemž jsou využívány informace čerpané z práce [9].

Adagrad

Jedním z algoritmů s adaptivní konstantou učení je Adagrad, který upravuje konstantu učení pro každý parametr modelu zvlášť.

Pro přehlednost si zaveďme označení pro hodnotu gradientu v kroku t:

$$g_t = \left\langle \frac{\partial E}{\partial \widetilde{w}} \middle| \widetilde{w}_t \right\rangle_{D_t}. \tag{6.23}$$

Poté probíhá aktualizace váhy w následovně:

$$r_{t+1} = r_t + g_t^2, \tag{6.24}$$

$$\Delta w_{t+1} = -\frac{\epsilon}{\xi + \sqrt{r_{t+1}}} \cdot g_t, \tag{6.25}$$

$$w_{t+1} = w_t + \Delta w_{t+1}, \tag{6.26}$$

kde ξ je velmi malá konstanta zavedená kvůli numerické stabilitě, ϵ je konstanta učení a $r_0 = 0$. Pro proměnnou r platí, že se během trénování neustále zvyšuje.

RMSprop

Další optimalizační metodou, která je uváděna v práci [9], je RMSprop. Oproti algoritmu Adagrad, který postupně potlačuje vliv konstanty učení kvůli akumulaci kvadrátů gradientů, mění RMSprop tuto akumulaci pomocí exponenciálně váženého klouzavého průměru. Tím RMSprop výrazně potlačuje vliv vzdálené historie. Váha w je v tomto algoritmu upravována následujícím způsobem:

$$r_{t+1} = \rho \cdot r_t + (1 - \rho)\, g_t^2, \tag{6.27}$$

$$\Delta w_{t+1} = -\frac{\epsilon}{\xi + \sqrt{r_{t+1}}} \cdot g_t, \tag{6.28}$$

$$w_{t+1} = w_t + \Delta w_{t+1}, \tag{6.29}$$

kde ρ je míra úpadku a g_t je dáno vztahem 6.23. Všechny ostatní proměnné zde mají stejný význam jako u algoritmu Adagrad.

Adam

Adam [22] je metodou pro efektivní stochastickou optimalizaci, která počítá adaptivní konstantu učení pro každý parametr zvlášť z odhadů prvního a druhého momentu gradientu. Algoritmus je popsán i v literatuře [9]. Nejprve je nutné nastavit počáteční hodnoty: $s_0 = 0$, $r_0 = 0$, $t = 0$. Optimalizace váhy w se pak řídí pomocí následujícího iterativního procesu:

$$s_{t+1} = \rho_1 \cdot s_t + (1 - \rho_1)\, g_t, \tag{6.30}$$

$$r_{t+1} = \rho_2 \cdot r_t + (1 - \rho_2)\, g_t^2, \tag{6.31}$$

$$\widehat{s} = \frac{s_{t+1}}{1 - \rho_1^{t+1}}, \tag{6.32}$$

$$\widehat{r} = \frac{r_{t+1}}{1 - \rho_2^{t+1}}, \tag{6.33}$$

$$\Delta w_{t+1} = -\epsilon \frac{\widehat{s}}{\xi + \sqrt{\widehat{r}}}, \tag{6.34}$$

$$w_{t+1} = w_t + \Delta w_{t+1}, \tag{6.35}$$

$$t \leftarrow t + 1, \tag{6.36}$$

kde konstanty ρ_1 a ρ_2 jsou vybírány z intervalu $\langle 0, 1)$, ξ je malá konstanta pro numerickou stabilitu a g_t je opět hodnota gradientu spočtená podle vzorce 6.23.

6.3.3 Doprovodné jevy trénování

Přetrénování

Jedním z hlavních problémů, se kterými se lze během trénování klasifikátoru setkat, je přetrénování. To nastává, pokud se model během trénování skvěle naučí klasifikovat trénovací data, ale při přivedení dat, se kterými se dosud nesetkal, na jeho vstup dosahuje špatných výsledků. Zřejmě nejsilnější a zároveň také nejjednodušší metodou pro redukci přetrénování je augmentace dat, jak je uváděno v práci [16]. Během augmentace dat dochází k umělému zvětšení datové množiny až na několikanásobek původní velikosti tak, že se na každý trénovací obraz aplikuje několik různých transformací zachovávajících původní zařazení obrazu. Jinak řečeno, transformace vytvoří nové obrazy odlišné od původního, ne však natolik odlišné, aby je expert, který prováděl anotaci výchozího obrazu, zařadil do jiné třídy než původní obrázek, jehož transformací byly získány. V úloze počítačového vidění se nabízí využít geometrickou transformaci obrazu, a to například afinní transformaci, popsanou v práci [23], která zahrnuje translaci, rotaci, zkosení i změnu měřítka. Její předpis vypadá následovně:

$$\mathbf{x}' = A\mathbf{x} + \mathbf{t}, \tag{6.37}$$

kde $\mathbf{t}$ je vektor translačních parametrů. Afinní transformace zachovává přímost čar a rovinnost povrchů.

Dávková normalizace

Dávková normalizace představuje mechanismus navržený v práci [24], který významně urychluje proces trénování hlubokých neuronových sítí. Také redukuje závislost gradientu na měřítku parametrů, což má příznivý efekt na tok gradientu sítí, a umožňuje tak volit vyšší hodnoty konstanty učení, aniž by došlo k divergenci.

Je známo, že trénování sítě konverguje rychleji, pokud její vstupy mají nulovou střední hodnotu a jednotkovou varianci a jsou dekorelované, čehož lze dosáhnout jejich lineární transformací zvanou bělení. Tento přístup se nabízí aplikovat na vstupy jednotlivých vrstev sítě, které jsou výstupy předešlých vrstev. Plné bělení vstupů každé vrstvy je ale výpočetně náročné.

Dávková normalizace spočívá v tom, že je normalizován každý vstup vrstvy nezávisle, a to tak, aby nabýval nulové střední hodnoty a jednotkové variance. Jinými slovy je normalizována každá dimenze vstupního vektoru $\mathbf{x} = \left(x^{(1)}, ..., x^{(d)}\right)$ vrstvy podle vztahu:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}\left[x^{(k)}\right]}{\sqrt{\mathrm{Var}\left[x^{(k)}\right]}}. \tag{6.38}$$

Tato normalizovaná hodnota je dále lineárně transformována:

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}, \tag{6.39}$$

kde parametry γ a β jsou trénovány. Pro jednoduchost se dále zaměřme pouze na k-tou dimenzi vstupního vektoru $x^{(k)}$ a vynechme horní index k v následujících vztazích. Za účelem

zjednodušení se pro výpočet střední hodnoty a variance nepoužívá celá trénovací množina, ale jen malá dávka o velikosti m vzorků, pro kterou je zavedeno označení $\mathcal{B} = \{x_1, ..., x_m\}$.

Vstupem algoritmu BN transformace (z anglického Batch Normalizing transform) jsou hodnoty x, které tvoří dávku $\mathcal{B} = \{x_1, ..., x_m\}$, a parametry k natrénování γ, β. Výstupem jsou hodnoty $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$, jejichž výpočet vypadá následovně:

$$\begin{aligned}
\mu_{\mathcal{B}} &\leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i, \\
\sigma_{\mathcal{B}}^2 &\leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2, \\
\widehat{x_i} &\leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}.
\end{aligned}$$

Hodnoty výstupu jsou pak dány vztahem:

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i). \tag{6.40}$$

Dávkovou normalizací se rozumí aplikace BN transformace, která je dána vztahem 6.40, na všechny vstupy následující vrstvy.

6.4 Ztrátová funkce

Otázkou zůstává, jak vhodně modelovat ztrátovou funkci, o jejíž minimalizaci se snaží trénovací algoritmy. Běžně používanými typy ztrátové funkce jsou například suma kvadrátů odchylek a křížová entropie, jimž je věnována pozornost v této kapitole.

Suma kvadrátů odchylek

Jednou ze základních ztrátových funkcí je míra založená na součtu kvadrátů odchylek, která je uváděna v práci [5] a jejíž předpis vypadá následovně:

$$E = \sum_{n} \sum_{k=1}^{c} \left(y_k^n - t_k^n\right)^2, \tag{6.41}$$

kde pokud je přiveden na vstup sítě n-tý trénovací obraz $\mathbf{x}^n$, tak y_k^n je prvek výstupního vektoru sítě pro třídu k a t_k^n je k-tý prvek vektoru požadovaného výstupu sítě. Písmeno c značí počet tříd.

Křížová entropie

Jak zmiňuje Bishop ve své práci [8], použití sumy kvadrátů jako ztrátové funkce nevede v klasifikační úloze k tak rychlému trénování jako použití křížové entropie, což zjistili Simard aj. v jejich práci [25]. Křížovou entropií v úloze klasifikace do více tříd se zabývá Bishop také ve své další práci [14], odkud jsou čerpány následující informace.

Za předpokladu, že se jedná o úlohu klasifikace do c tříd, řídí se výpočet ztrátové funkce ve smyslu křížové entropie vztahem:

$$E = -\sum_{n} \sum_{k=1}^{c} t_k^n \ln y_k^n, \tag{6.42}$$

kde stejně jako pro vztah 6.41 platí, že pokud je přiveden na vstup sítě n-tý trénovací obraz $\mathbf{x}^n$, tak y_k^n je prvek výstupního vektoru sítě pro třídu k a t_k^n je k-tý prvek vektoru požadovaného výstupu sítě, který je ve formě one-hot vektoru, což znamená, že obsahuje hodnotu 1 na pozici třídy, do které příslušný vstupní obraz $\mathbf{x}^n$ náleží, a 0 jinde. Písmeno c značí celkový počet tříd. V ideálním případě, kde by šlo o naprosto přesnou klasifikaci, tedy $y_k^n = t_k^n$ pro všechny hodnoty k a n, by nabývala ztrátová funkce minima:

$$E_{min} = -\sum_{n} \sum_{k=1}^{c} t_k^n \ln t_k^n. \tag{6.43}$$

V této práci jsou k dispozici anotace ve formě one-hot vektorů, tudíž by v ideálním případě došlo podle vztahu 6.43 k minimalizaci ztrátové funkce na hodnotu $E_{min} = 0$.

Kapitola 7

Konvoluční neuronové sítě

Jak upozorňuje Bishop ve své práci [8], rozpoznávání obrazových dat pomocí neuronových sítí, které jsou plně propojené, jak je například znázorněno na obrázku 6.1, kde jsou na vstup každého neuronu jedné vrstvy připojeny s určitou vahou výstupy všech neuronů vrstvy předešlé, nerespektuje jednu zásadní vlastnost obrazových dat, a sice tu, že sousední pixely obrazu jsou mnohem více korelovány než ty, které jsou od sebe vzdálené. Síla konvolučních neuronových sítí neboli CNN (z anglického Convolutional Neural Networks) spočívá především v tom, že dokáží extrahovat a využít lokální vlastnosti spjaté s malými regiony obrazu a v dalších vrstvách tyto vlastnosti slučovat za účelem získání informace o celém obrazu, což umožňuje jejich nezávislost na nejrůznějších transformacích vstupu. Jak se uvádí v práci [9], konvoluční neuronové sítě jsou neuronové sítě, které používají konvoluci místo obecného násobení matic v alespoň jedné ze svých vrstev. V těchto konvolučních vrstvách jsou jednotlivé jednotky organizovány do rovin, z nichž každá se nazývá mapa příznaků, někdy též zvaná jako aktivační mapa (viz práce [26]). Každá jednotka v mapě příznaků pak obsahuje informace o malém regionu obrazu. Další zajímavou vlastností konvolučních neuronových sítí je, že jednotky v mapě příznaků sdílejí stejné hodnoty vah. Konvoluční vrstvě a dalším využívaným vrstvám je dále věnována pozornost v kapitole 7.1.

Konvoluční neuronové sítě se staly silným nástrojem v úlohách počítačového vidění. Jejich využití lze nalézt například v klasifikaci obrazů [16], detekci objektů v obraze [27] nebo v úlohách segmentace [28].

7.1 Vrstvy

Konvoluční neuronové sítě se skládají z různých typů vrstev, které byly přehledně shrnuty v práci [26], odkud byly čerpány informace v této kapitole. Mezi základní typy vrstev patří:

- konvoluční vrstva,
- pooling vrstva.

Konvoluční vrstva

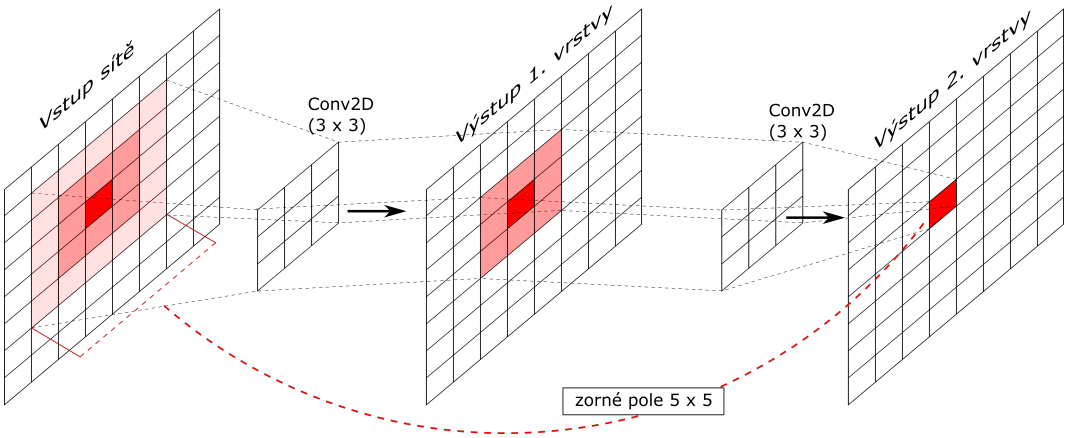
Základním typem vrstvy v konvoluční neuronové síti je konvoluční vrstva, která se skládá z předem definovaného počtu filtrů s pevně danou výškou a šířkou. Třetí rozměr, tedy hloubka konvolučních jader neboli filtrů, vždy odpovídá hloubce vstupu do dané vrstvy, kterým bývá typicky obraz o hloubce větší než 1. Například pokud je na vstupu konvoluční vrstvy RGB obrázek, je jeho hloubka rovna počtu kanálů, tedy 3. Filtry konvoluční vrstvy pak mají rozměry $m \times n \times 3$. Konvolucí každého filtru se vstupem sítě vznikne 2D mapa příznaků. Počet filtrů se volí pevně pro každou vrstvu. Například v síti VGG16 [29] byl zvolen počet jader první konvoluční vrstvy na 64 a v dalších vrstvách byl zvyšován, přičemž byl jejich počet vždy roven mocnině čísla 2. Je zřejmé, že počet jader konvoluční vrstvy určuje hloubku jejího výstupu neboli aktivační mapy.

Místo vah, které jsou přítomny ve vrstvách klasických neuronových sítí (viz kapitola 6.1), jsou v konvolučních vrstvách trénovány parametry jednotlivých filtrů. To znamená, že se zvyšujícím se počtem jader v konvoluční vrstvě se zvyšuje také počet parametrů, které je potřeba natrénovat.

Každý prvek výstupní 3D struktury konvoluční vrstvy lze interpretovat jako výstup neuronu, jehož zorné pole (viz další odstavce) zahrnuje pouze malé okolí vstupu. Pokud je i vstup vrstvy interpretován jako 3D struktura výstupů neuronů předchozí vrstvy, lze říci, že je daný neuron aktuální vrstvy spojen s výstupy neuronů, které se v předchozí vrstvě nacházejí na jeho pozici a v přilehlém okolí vymezeném daným konvolučním filtrem. Za váhy těchto spojení je pak možné považovat hodnoty na příslušných pozicích filtru. Každý takto interpretovaný neuron tedy využívá informaci jen z malého regionu a je spojen jen s okolními neurony uvnitř tohoto regionu z předchozí vrstvy a nikoliv se všemi, jako tomu je u plně propojených vrstev. To vede k výraznému snížení počtu parametrů k natrénování.

Vedle počtu konvolučních jader a jejich rozměrů je při inicializaci konvoluční vrstvy třeba určit také velikost kroku, o který se konvoluční jádro posune během prostupování obrázkem. Problém nastává, pokud je požadováno, aby výstup konvoluční vrstvy měl stejnou výšku a šířku jako její vstup. V tom případě je zapotřebí uměle rozšířit okraje vstupního obrázku o takový počet řádků a sloupců, aby bylo možné spočítat hodnotu odezvy filtru i ve všech rozích obrázku. Takovéto umělé rozšíření nejčastěji spočívá ve vložení nulových hodnot podél okrajů obrázku, což je způsob, který byl využíván i v této práci.

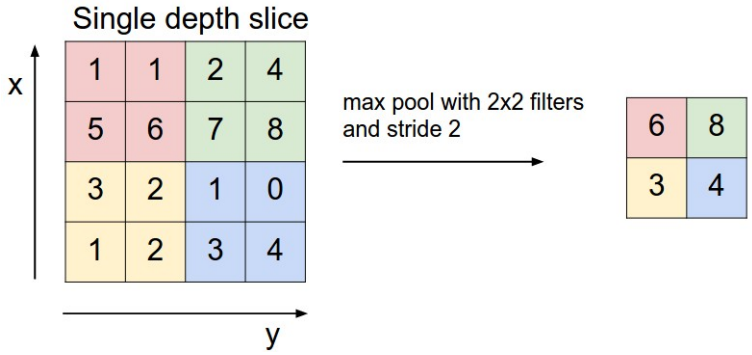
Velikost kroku a rozměry filtrů určují další důležitou vlastnost vrstvy sítě, a to tzv. zorné pole, které udává, z jak velkého prostoru ve vstupním obrázku obsahuje informaci výstup neuronu, tedy prvek výstupní mapy příznaků. Je zřejmé, že velikost zorného pole je rovna rozměrům použitých konvolučních jader. V této práci bylo ale především zkoumáno, jak bude velké zorné pole v jednotlivých vrstvách vzhledem ke vstupu celé sítě, nikoliv ke vstupu samotných vrstev. Zorné pole bude v této práci dále označovat zorné pole vzhledem ke vstupu celé sítě. Lze si jednoduše odvodit, že se jeho výška, resp. šířka, postupně s přidáváním dalších konvolučních vrstev zvyšuje, pokud mají tyto vrstvy konvoluční jádra výšku, resp. šířku, větší než 1. Tento případ je demonstrován na obrázku 7.1, kde jsou použity dvě za sebou jdoucí konvoluční vrstvy o rozměrech jader 3×3.



Obrázek 7.1: Ukázka zorného pole v hlubších vrstvách

Pooling vrstva

Pooling vrstvy obecně slouží ke snížení rozměrů reprezentace obrazu v dalších vrstvách kvůli redukci počtu trénovaných parametrů sítě. Jak je zmíněno v práci [30], dalším přínosem pooling vrstev je to, že jejich použití vede k získání příznaků, které jsou nezávislé na translaci. Výstupem pooling vrstvy je mapa příznaků, která odpovídá mapě příznaků předcházející vrstvy. Hodnota každého prvku výstupní mapy příznaků je spočtena z malého regionu nacházejícího se ve vstupní mapě na odpovídající pozici. Jedním způsobem je počítání průměru hodnot v daném regionu. Druhým způsobem je spočítat pouze maximální hodnotu vyskytující se v daném region, což je princip tzv. max-pooling vrstvy, který je v současné době používanější a v práci [30] dosahoval lepších výsledků než přistup zahrnující průměrování. U pooling vrstvy se nastavují jak rozměry daných regionů, tak velikost kroku mezi dvěma sousedícími regiony, ze kterých jsou výstupy vrstvy počítány.



Obrázek 7.2: Ukázka toho, jak funguje vrstva max-pooling. Obrázek byl přejat z práce [26].

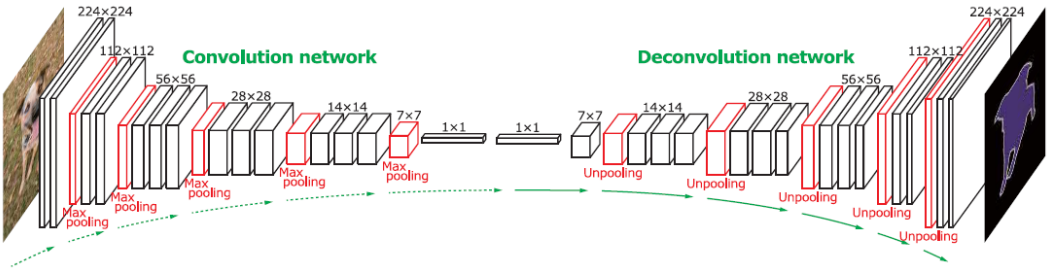
Jak je uváděno v práci [16], tradiční metodou je volit parametry pooling vrstvy tak, aby se jednotlivé regiony nepřekrývaly. Ve zmíněné práci jsou ale využívány max-pooling vrstvy s takovým nastavením, že se regiony překrývají. V architektuře sítě VGG16 [29] byly oproti tomu použity max-pooling vrstvy o velikosti jader 2×2 s krokem o velikosti 2, což znamená, že je vstup rozdělen na nepřekrývající se regiony o velikosti 2×2, z nichž

je získána jejich maximální hodnota. Tato metoda, jak je uváděno v práci [29], překonala v prováděných experimentech přístup navržený v práci [16], kde se jednotlivé regiony u max-pooling vrstvy překrývaly. Ukázka toho, jak funguje max-pooling vrstva použitá v architektuře sítě VGG16, je k vidění na obrázku 7.2.

7.2 Dekonvoluční síť

Jelikož jsou v této práci k dispozici anotace per pixel, tedy informací od učitele je segmentovaný obraz, nabízí se využít takovou síť, která vstupní obraz také segmentuje. Výstup dobře natrénované sítě by pak byl v ideálním případě totožný se skutečnou segmentační mapou.

Takový princip je využíván i práci [2], kde je navržena architektura sítě pro sémantickou segmentaci obrazu, kdy je každému pixelu vstupního obrazu přiřazena značka některé třídy. Schéma této architektury, zvané DeconvNet, je k znázorněno na obrázku 7.3, kde je vidět, že se síť skládá ze dvou částí.



Obrázek 7.3: Architektura sítě DeconvNet navržené v práci [2], odkud byl tento obrázek přejat. Je zde k vidění napojení dekonvoluční sítě na výstup konvoluční sítě a také znázorněný výstup sítě pro daný vstupní obrázek.

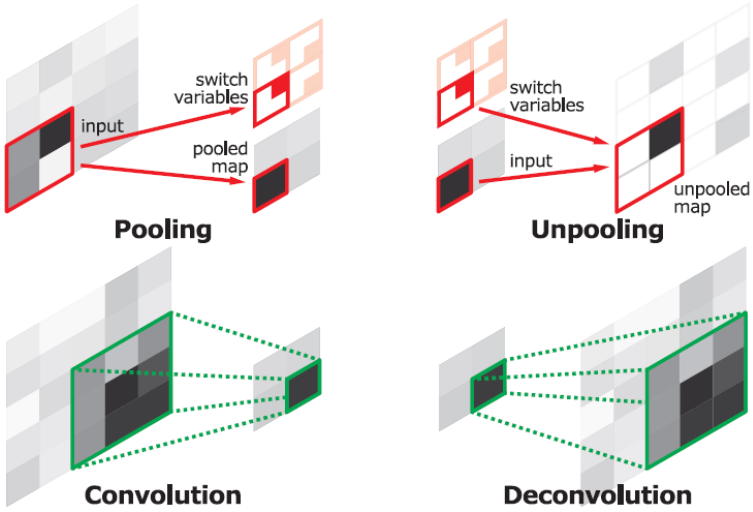
První část tvoří konvoluční neuronová síť, která odpovídá extraktoru příznaků. Ve zmíněné práci [2] byla použita konvoluční síť na bázi VGG16 [29], kde byla odebrána poslední klasifikační vrstva. Její schéma je k vidění v levé polovině obrázku 7.3. Druhá část představuje dekonvoluční síť, která je připojena na výstup konvoluční sítě a zastává funkci generátoru tvaru, který produkuje segmentaci objektů z příznaků extrahovaných konvoluční sítí. Výstupem celé sítě je pak pravděpodobnostní mapa, která udává pravděpodobnost náležitosti jednotlivých pixelů do daných tříd. Dekonvoluční síť je znázorněna v pravé polovině obrázku 7.3, kde si lze všimnout, že je zrcadlovou verzí konvoluční sítě.

Zatímco konvoluční síť obsahuje konvoluční a pooling vrstvy a redukuje velikost mapy příznaků, dekonvoluční síť zahrnuje unpooling a dekonvoluční vrstvy a velikost mapy příznaků naopak zvyšuje.

Podobnou zrcadlovou architekturu má i síť SegNet, která byla navržena v práci [31], kde je konvoluční síť označována jako kódovací a dekonvoluční síť jako dekódovací. V této práci bude používáno ještě jednodušší značení, a sice enkodér pro kódovací síť a dekodér pro síť dekódovací.

Unpooling vrstva

Unpooling vrstva provádí oproti pooling vrstvě zpětnou operaci, kterou rekonstruuje původní velikost mapy příznaků, jak je znázorněno na obrázku 7.4. Během operace pooling jsou zaznamenávány lokace vybraných prvků, které se využijí v unpooling vrstvě k umístění příslušných prvků do původní pozice.



Obrázek 7.4: V horní části je znázorněna funkce unpooling vrstvy, dole je pak k vidění proces dekonvoluce. Obrázek byl přejat z práce [2].

Dekonvoluční vrstva

Výstupem unpooling vrstvy je zvětšená řídká mapa příznaků. Dekonvoluční vrstva tuto mapu zahušťuje pomocí operace podobné konvoluci. Oproti konvolučním vrstvám, které připojují několik vstupů uvnitř okna filtru k jednomu výstupu, spojují dekonvoluční vrstvy jeden vstup s několika výstupy, jak je znázorněno na obrázku 7.4. Stejně jako konvoluční vrstvy obsahují i dekonvoluční vrstvy stanovený počet filtrů, které jsou trénovány.

Upsampling vrstva

V knihovně Keras [32] je implementována tzv. upsampling vrstva, která má s unpooling vrstvou společné to, že rekonstruuje původní velikost mapy příznaků. Rozdíl je v tom, že neumisťuje jednotlivé vstupy na zapamatované pozice, ale jen opakuje řádky a sloupce vstupní mapy příznaků tolikrát, kolikrát je jí zadáno.

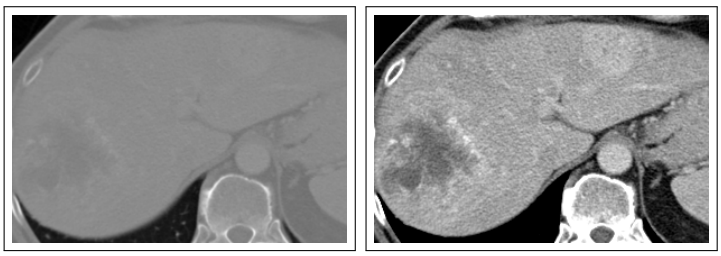
Kapitola 8

Data

Obrazovými daty byly CT snímky zaměřené na oblast jater. Jednalo se tedy o 3D obrázky o rozměrech $m \times n \times k$, kde každá z k úrovní představovala jeden řez, tedy 2D snímek o rozměrech $m \times n$. Ke všem obrazům byla dále k dispozici maska stejných rozměrů, která obsahovala informaci o třídě, do které dané pixely obrazu náleží. Maska byla výsledkem manuální segmentace, která byla prováděna lékaři. Tyto anotace bylo nezbytné mít k dispozici nejen pro trénování modelu, ale také pro ohodnocení natrénovaného klasifikátoru.

8.1 Obrazová data

Jako trénovací data sloužily CT snímky, ze kterých byly vytaženy 2D řezy. Pixely v surových obrazových datech, které byly pro tuto úlohu použity, nabývají hodnot od -1024 do 2420 HU. Na každý řez bylo aplikováno CT okno, jehož princip je vysvětlen v kapitole 2.2 a kterým byly vytaženy pixely s hodnotami -125 až 225 HU. Následně došlo k rovnoměrnému rozložení těchto hodnot na interval jasových hodnot 0 až 255. Ukázku původního neupraveného obrázku a obrázku po aplikaci CT okna lze vidět na obrázku 8.1.



(a) (b)

Obrázek 8.1: Vlevo se nachází surový CT řez, vpravo je tentýž řez po aplikaci CT okna.

Jako data pro obě úlohy pak sloužily obrázky po aplikaci CT okna. Pro úlohu detekce ohraničujících obdélníků pomocí přístupu kombinujícího metody HoG a SVM bylo ještě prováděno předzpracování obrazu metodami, které byly představeny v kapitole 3. U neuronových sítí k žádnému předzpracování nedocházelo.

8.2 Anotace

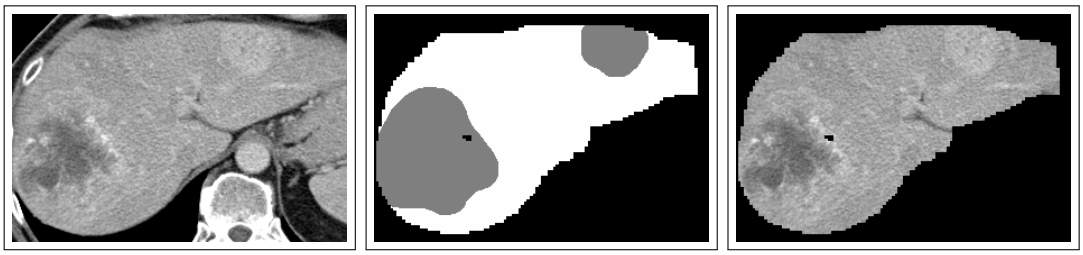
Anotace byly výstupem manuální segmentace jater a jaterních lézí. Každý pixel nabýval jedné ze tří hodnot podle toho, zda představoval lézi, zdravou jaterní tkáň, či okolí, tedy struktury mimo játra (viz tabulka 8.1).

0	1	2
Okolí jater	Jaterní léze	Zdravá jaterní tkáň

Tabulka 8.1: Číselné hodnoty v anotacích a jejich význam

Jak u přístupu založeného na histogramu orientovaných gradientů, tak i u přístupu založeného na neuronových sítích bylo vycházeno z předpokladu, že u obrazových dat, nad nimiž má být provedena detekce jaterních lézí, došlo již k segmentaci jater a je tudíž k dispozici jejich maska. V obou případech byla tato informace využita jiným způsobem. U prvního přístupu, spočívajícího v detekci pozitivních obdélníků obsahujících léze, byly automaticky odstraňovány takové detekce, které se nacházely mimo oblast jater.

U neuronových sítí byla prováděna detekce per pixel, tedy každý pixel byl sítí zařazován do jedné ze tří tříd. Pro usnadnění trénovacího procesu byla oblast v obraze, ve které se jaterní tkáň nevyskytuje, přebarvena na nulovou hodnotu pro všechny pixely. Ukázka 2D řezu před přebarvením a po přebarvení nejaterních struktur je znázorněna na obrázku 8.2.



(a) (b) (c)

Obrázek 8.2: Ukázka přebarvení pozadí podle masky jater. Zleva je k vidění vstupní obrázek, maska a obrázek po přebarvení pozadí podle masky.

Na obrázku 8.2b je k vidění maska jednoho z anotovaných obrazů. Šedá barva znázorňuje jaterní léze, bílá zdravou jaterní tkáň a černé je pozadí. V praxi u klasifikační úlohy samozřejmě sice nebude k dispozici informace o lézích, ale obraz bude rozdělen na játra a pozadí, což pro jeho přebarvení v originálním snímku postačí.

Kapitola 9

HoG experimenty

Cílem testování bylo určit takovou konfiguraci, která by vedla k nejlepším výsledkům ve smyslu stanoveného kritéria. V kapitole 9.1 byla prováděna křížová validace, na základě jejíž výsledků došlo k redukci počtu konfigurací testovaných v kapitole 9.2, kde byla zkoumána míra překrytí anotovaných lézí a ohraničujících obdélníků detekovaných klasifikátorem při daném nastavení parametrů, z nichž byly testovány následující:

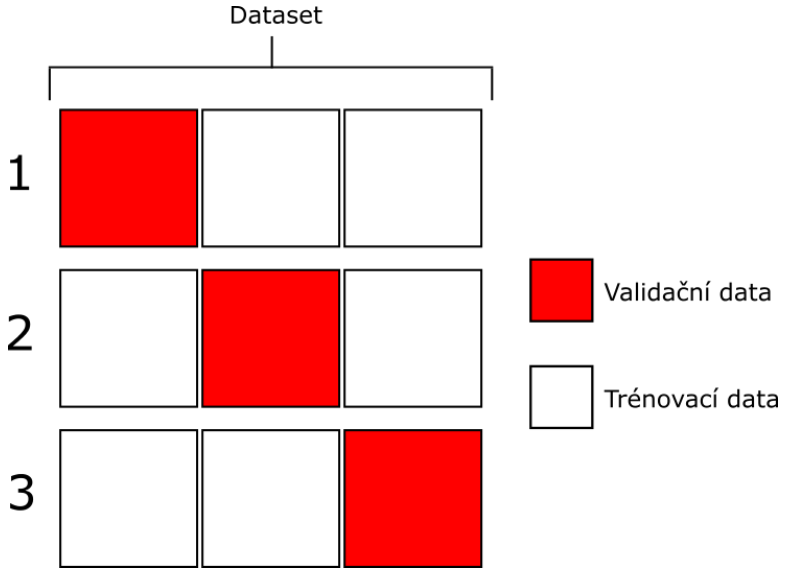
- parametry HoG deskriptoru:
  - počet orientací,
  - velikost buněk,
  - počet buněk v každém bloku,
- metoda zpracování obrazu:
  - bilaterální filtr pro různé parametry σ_r, σ_d a velikosti jádra (viz kapitola 3.2),
  - mediánový filtr pro různé velikosti jádra.

Křížová validace posloužila především k výběru několika nejlepších parametrů HoG deskriptoru, jehož optimální nastavení a metoda zpracování obrazu byly určeny až na základě výsledků detekce lézí pomocí klouzavého okénka.

Všechny experimenty byly implementovány v jazyce Python, přičemž pro metody strojového učení byla využita knihovna scikit-learn [10] a pro zpracování obrazu knihovna OpenCV [7].

9.1 Křížová validace

Jako validační metoda byla využita k-násobná (z anglického k-fold) křížová validace, během které je množina dat, jak je popsáno v práci [33], rozdělena do k nepřekrývajících se stejně velkých podmnožin. Poté je klasifikátor natrénován $k\times$, a to vždy tak, že jedna z podmnožin je použita jako validační a zbytek dat je předán klasifikátoru pro natrénování. Validační data se po natrénování klasifikátoru použijí pro jeho ohodnocení dle zvolené metriky. Tento proces je proveden pro všech k podmnožin. Výsledná metrika ohodnocující klasifikátor je obvykle spočtena z aritmetického průměru k dílčích hodnot metriky získaných během tohoto iterativního procesu. Ukázka rozdělení datové sady pro k-násobnou křížovou validaci, kde $k = 3$, je k vidění na obrázku 9.1.



Obrázek 9.1: Ukázka rozdělení datové sady při k-násobné křížové validaci, kdy $k = 3$.

9.1.1 Použité hodnotící metriky

Pro definici hodnotících metrik je důležité zkonstruovat tzv. matici záměn, kterou popisují Visa aj. ve své práci [34]. Matice záměn pro klasifikaci do dvou tříd je uvedena v tabulce 9.1.

		Skutečnost	
		P	N
Predikce	P	a	b
	N	c	d

Tabulka 9.1: Matice záměn pro klasifikaci do dvou tříd

Srozumitelnější forma zápisu této matice, která je uváděna v práci [35], je znázorněna v tabulce 9.2.

		Skutečnost	
		P	N
Predikce	P	TP	FP
	N	FN	TN

Tabulka 9.2: Matice záměn

Jelikož se jedná o klasifikaci do dvou tříd, budou jednotlivá data spadat buď do třídy N (negativní), nebo do třídy P (pozitivní). Prvky matice záměn v tabulce mají pro účely této práce následující význam:

- **TP** (z anglického true positives) značí počet případů, kdy byl obdélník správně označen jako pozitivní.
- **FP** (z anglického false positives) značí počet obdélníků, které byly nesprávně zařazeny mezi pozitivní.
- **FN** (z anglického false negatives) označuje počet případů, ve kterých došlo k vyhodnocení obdélníku jako negativního, přestože se ve skutečnosti jednalo o pozitivní obdélník.
- **TN** (z anglického true negatives) představuje počet obdélníků správně vyhodnocených jako negativní.

Je zřejmé, že míra kvality klasifikátoru určitým způsobem koresponduje s hodnotami uvnitř matice záměn. Snahou je obecně zvýšit počet správných detekcí, tedy TP a TN, a naopak snížit počet FP a FN. K exaktnímu porovnání klasifikátorů na základě hodnot prvků matice záměn se využívají následující míry, které jsou popsány v pracích [35] a [9]:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{9.1}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{9.2}$$

$$\text{F-míra} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{9.3}$$

Především byl kladen důraz na to, aby bylo detekováno co nejvíce lézí, a to i za cenu zvýšeného množství falešných detekcí, tedy false positives (FP). Proto byl recall nejdůležitější mírou. Její hodnoty se stejně jako u ostatních uvedených metrik pohybují v intervalu $\langle 0,\ 1\rangle$, přičemž k dosažení maximální hodnoty 1 dojde jen tehdy, když je počet false negatives roven $\text{FN} = 0$, tedy v případě, že jsou všechny léze správně detekovány. Proto bylo cílem tuto míru maximalizovat.

9.1.2 Výsledky

Extrakce dat z CT snímků probíhala tak, že pro každý řez, kde se vyskytovaly léze, byly spočteny souřadnice obdélníků ohraničujících jejich oblast. Obrazová data získaná vyříznutím obdélníků z řezu byla následně uložena mezi pozitivní obrázky obsahující léze. V řezech, kde se žádné léze nevyskytovaly, bylo extrahováno několik menších snímků z oblasti uvnitř jater a ty byly následně uloženy mezi negativní snímky, které obsahují pouze zdravou jaterní tkáň. Jejich přesný počet byl zvolen tak, aby bylo pro trénování k dispozici přibližně stejné množství pozitivních a negativních obrazových dat.

V této části byla prováděna trojnásobná křížová validace, během níž byla data rozdělena na trénovací a testovací v poměru 2 : 1 a prováděny byly 3 iterace. Data v obou množinách byla augmentována pomocí afinní transformace obrazu. Dále bylo zajištěno, aby počet pozitivních a negativních dat byl přibližně stejný a aby se každý vektor příznaků vyskytoval v testovací množině právě jednou.

Hodnota recall dosahovala pro všechny provedené křížové validace průměrné hodnoty 0.9009. Pro precision byla tato hodnota vyšší, a sice 0.9499. Řada konfigurací vykazovala výrazně nižší hodnoty recall nebo precision oproti běžným výsledkům, čímž byla vysoce ovlivněna průměrná skóre. Navíc bylo účelem stanovit nejlepší konfiguraci a výsledky vykazující podprůměrné hodnoty recall nejsou pro porovnání příliš důležité. Pokud vybereme pouze takové výsledky, kde recall ≥ 0.90 a precision ≥ 0.95, získáme relevantní výsledky, z nichž lze spočítat průměrné hodnoty recall pro různé konfigurace. Výsledky křížové validace je možné vidět v tabulkách 9.3, 9.4 a 9.5, kde jsou zaznamenána průměrná skóre recall pro různé hodnoty HoG parametrů.

Orientace	Recall
12	0.9221
9	0.9201
6	0.9170

Tabulka 9.3: Průměrná hodnota recall pro různý počet orientací

PPC	Recall
4	0.9239
6	0.9226
8	0.9184
10	0.9099

Tabulka 9.4: Průměrná hodnota recall pro různé velikosti buněk

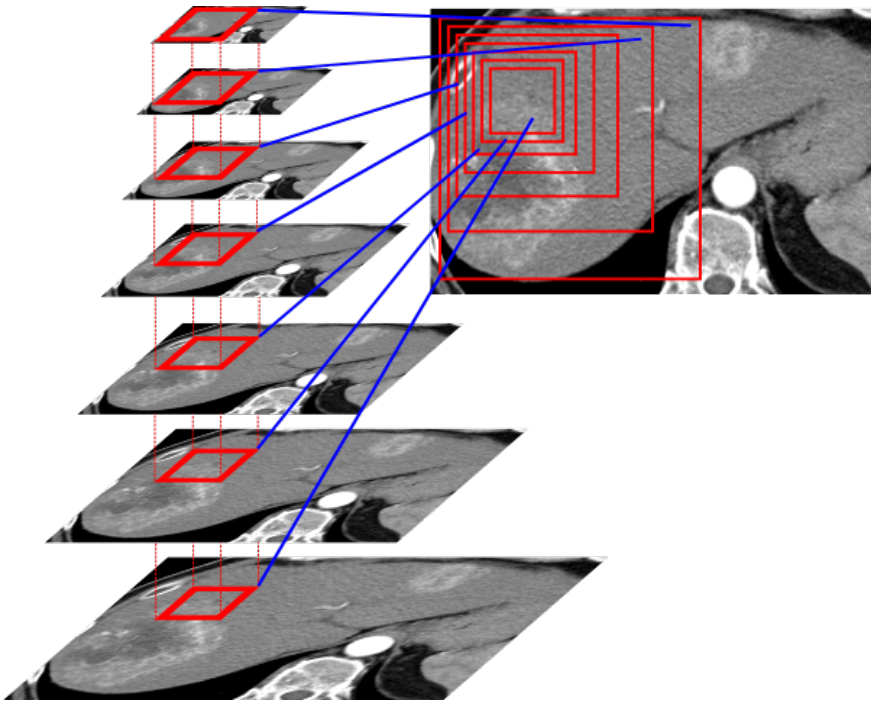
CPB	Recall
2	0.9202
3	0.9192

Tabulka 9.5: Průměrná hodnota recall pro dané počty buněk v bloku

Tabulka 9.5 ukazuje, že 2 buňky na blok vykazují nepatrně lepší výsledky než 3. Jelikož méně buněk na blok implikuje menší velikost vektoru příznaků, je hodnota CPB = 2 více žádoucí. U obou dalších parametrů, tedy u orientace a velikosti buňky, dosahovala nejlepších výsledků hodnota parametru vedoucí ke zvýšenému počtu příznaků. Pro další testování byla tedy vybrána i druhá nejlepší nastavení obou parametrů, která vedou k redukci velikosti vektoru příznaků, a tím i ke snížení paměťových nároků. Dále tedy byly testovány počty orientací o hodnotách 12 a 9 a buňky o velikostech 4 a 6.

9.2 Testování pyramidy a klouzavého okénka

Během křížové validace byly používány již extrahované čtvercové obrázky o velikosti 48×48 pixelů a ty byly následně umístěny do trénovací nebo testovací sady. Poté byla na testovacích datech zkoumána úspěšnost klasifikace modelu SVM natrénovaného na trénovací množině. V další části byla použita stejná trénovací sada, testovacími obrázky ale byly celé řezy, ze kterých byly, podobně jako v práci [36], postupně extrahovány malé regiony, o nichž natrénovaný klasifikátor rozhodoval, zda patří mezi pozitivní data. Pro každý region bylo nutné provést stejné předzpracování obrazu a extrakci vektoru příznaků jako během vytváření trénovacích dat. Regiony byly extrahovány z okénka, které se pohybovalo po celém řezu s pevně daným horizontálním a vertikálním krokem. Toto okénko se nazývá klouzavé okénko a jeho velikost kroku byla nastavena na stejnou hodnotu pro oba směry. Kdyby byla velikost kroku rovna 48 pixelů, tedy stejná jako velikost okénka, dva za sebou jdoucí výřezy produkované pomocí klouzavého okénka by se vůbec nepřekrývaly. Pokud by se léze vyskytovala na okraji těchto výřezů, nemusela by být detekována. Proto bylo zapotřebí zjemnit krok klouzavého okénka, aby byly takovéto léze zachyceny. V takovém případě se se snižujícím krokem zvyšuje míra překrytí dvou za sebou jdoucích výřezů. Jako možná procentuální překrytí se nabízely například hodnoty $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ atd. Jako dostatečné překrytí se ukázala hodnota $\frac{1}{8}$ velikosti okénka, tedy 6 pixelů. Další snížení kroku na 3 pixely sice vedlo k ještě většímu zjemnění, ale za cenu čtyřnásobného prodloužení času detekce.



Obrázek 9.2: Pyramida obrazu a projekce klouzavého okénka do jejích jednotlivých vrstev

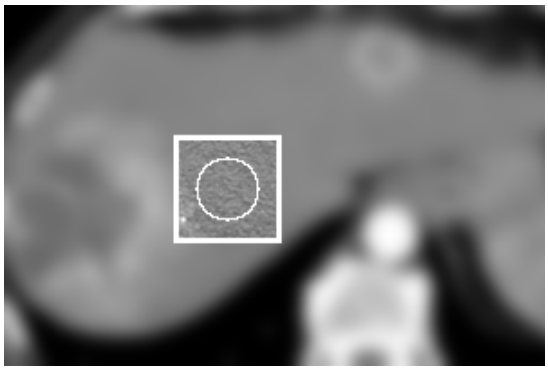
Pokud by všechny léze měly ve všech řezech fixní rozměry, které by byly navíc předem známy, bylo by možné podle nich nastavit velikost klouzavého okénka, kterému by stačil

jediný průchod snímkem k detekci všech objektů. Rozměry i tvar lézí se nicméně vyznačují vysokou variabilitou a detekovat všechny tyto artefakty pomocí jednoho průchodu okénka s pevnou velikostí by bylo nemožné. Jako řešení se nabízí tzv. pyramida obrazu, která je popsána v práci [37]. Pyramida se skládá z vrstev, z nichž každá obsahuje tentýž obraz, ale v jiném měřítku, jak uvádí dokumentace OpenCV [38]. V této úloze obsahuje nejspodnější vrstva pyramidy původní obraz. Každá vyšší vrstva je poté tvořena obrázkem s nižším rozlišením než předcházející nižší vrstva. V knihovně OpenCV [38] se standardně používá pyramida, kde pokud má obrázek rozlišení $M \times N$, ve vyšší vrstvě bude mít rozlišení $\frac{M}{2} \times \frac{N}{2}$, tedy čtyřikrát menší. Při takto velkých změnách měřítka by se mohlo stát, že některé artefakty, které je cílem detekovat, budou příliš velké na to, aby byly detekovány okénkem v nižší vrstvě, ale rovněž moc malé pro detekci v bezprostředně následující vyšší vrstvě. Proto bylo vhodné zjemnit změnu měřítka. Můžeme říci, že obraz v původní pyramidě s rozlišením $M \times N$ měl v následující vyšší vrstvě rozměry $\frac{M}{k} \times \frac{N}{k}$, kde $k = 2$ si označme jako změnu měřítka. Zjemnění přechodů mezi vrstvami bylo zajištěno jejím snížením na $k = 1.25$, což na druhou stranu vedlo ke zvýšení počtu vrstev a tím i k prodloužení doby prohledávání jednoho řezu okénkem. Pyramida pro jeden z testovacích řezů s projekcí klouzavého okénka do každé vrstvy je vidět na obrázku 9.2. V jeho pravém horním rohu jsou vykresleny hranice ploch, které okénko v jednotlivých vrstvách pokrývá. Je nutné si ovšem uvědomit, že ve vyšších vrstvách jsou obrazy mnohonásobně zmenšeny, tudíž po zobrazení v původní velikosti by byly ve skutečnosti značně rozmazány.

Pro každé okno byla provedena extrakce vektoru příznaků a jeho následná klasifikace do jedné ze dvou tříd. Podmínkou detekce bylo, aby pravděpodobnost, že obraz náleží do třídy 1, tedy mezi pozitivní data, dosahovala alespoň stanovené minimální hodnoty $P \geq P_{min}$. Pro následující testování byla tato prahová hodnota nastavena na $P_{min} = 0.75$. Výběrem optimální hodnoty P_{min} se budeme zabývat v kapitole 9.3.

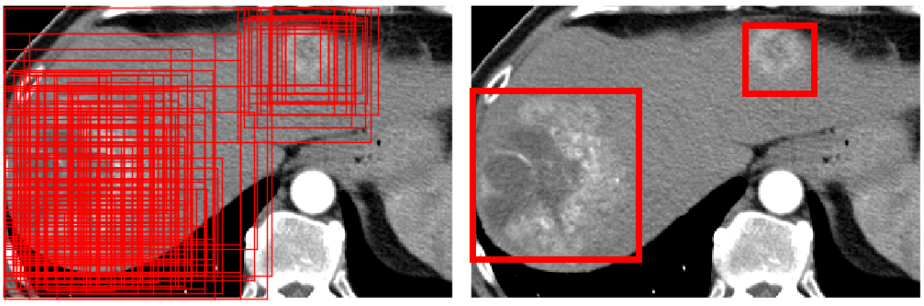
Může se stát, že k detekci dojde také v oblastech mimo játra, kde se mohou vyskytovat struktury podobné lézím, tedy artefakty podobného tvaru s tmavší, nebo světlejší intenzitou, než je pozadí. Jelikož úloha probíhá na segmentovaných datech, kde je k dispozici maska jater, může být navíc přidána podmínka minimálního procentuálního zastoupení jaterní tkáně v detekovaném obrázku. Tato prahová hodnota byla nastavena na hodnotu 50% a navíc byla přidána další podmínka, že středová oblast detekovaného obrazu musí být pokryta maskou jater alespoň z 90%, přičemž středovou oblastí se rozumí plocha ohraničená elipsou s počátkem ve středu obrazu a s poloměrem rovným 60% poloviny délky hrany obrázku. Jelikož byly oba rozměry okna nastaveny na stejnou hodnotu w = 48 pixelů, jedná se o kruhovou oblast. Ukázka klouzavého okénka se zvýrazněnou hranicí středové oblasti je k vidění na obrázku 9.3.

Jeden z velkých problémů této metody spočívá v tom, že pokud je velikost kroku klouzavého okénka zvolena tak, že se dva za sebou jdoucí výřezy značně překrývají, jak tomu je i v tomto případě, dochází k detekci lézí hned v několika za sebou jdoucích výřezech. Rovněž také může dojít k detekci té samé léze ve více úrovních pyramidy obrazu. Je tedy velice pravděpodobné, že pokud by byla léze detekována, byla by ohraničena více než



Obrázek 9.3: Ukázka klouzavého okénka se zvýrazněnou hranicí kruhové středové oblasti

jedním ohraničujícím obdélníkem, což by mimo jiné způsobilo i problémy s ohodnocením výsledků (viz kapitola 9.2.1). Samozřejmě je požadováno, aby jedna léze byla detekována pouze jedním obdélníkem. Toho lze dosáhnout metodou potlačení nemaxim neboli NMS (z anglického Non-Maximum Suppression), kterou se zabývá práce [39]. Jak se zde uvádí, metodou NMS lze odstranit nadbytečné detekce. Výstupem detektoru pak nejsou všechny pozitivní detekce, ale pouze ty, které jsou výstupem NMS. Metoda NMS vyžaduje jako vstup množinu detekovaných obdélníků a k nim přiřazených pravděpodobností, že jde skutečně o pozitivní data. Tyto pravděpodobnosti slouží jako skóre. Jak se uvádí v práci [40], detekce jsou nejdříve seřazeny podle svého skóre a následně přidávány na výstup, přičemž dochází k přeskakování těch, pro které již existuje detekce s překrytím, které je větší nebo rovno nějakému předem definovanému prahu. V této práci byl zvolen práh minimálního překrytí v metodě NMS empiricky na hodnotu 2%. Byly tedy vybírány pouze obdélníky, které se téměř nepřekrývají. Ukázka, jaký vliv má aplikace NMS na redukci počtu detekovaných obdélníků, je k vidění na obrázku 9.4.



(a) (b)

Obrázek 9.4: Vlevo jsou do testovaného obrázku vykresleny všechny detekované ohraničující obdélníky, vpravo jsou vykresleny jen obdélníky, které byly zachovány po aplikaci NMS.

Dále stojí za zmínku, že ještě před vyhodnocením výsledků každého experimentu byla provedena metoda HNM (z anglického Hard Negative Mining), která je stručně popsána v práci [41]. Zde je metoda HNM popsána jako proces, kdy je model natrénovaný na datové množině obsahující pozitivní a negativní vzorky následně aplikován na větší datovou množinu k získání vzorků, které byly nesprávně detekovány, tedy FP. Tyto FP jsou poté přidány do trénovací množiny a následně je model znovu natrénován. Tento proces se obvykle provádí pouze jednou. V této práci byly získávány FP tak, že byl proces detekce pomocí klouzavého okénka a pyramidy obrazu aplikován na řezy, z nichž byla extrahována trénovací data a které žádné léze neobsahovaly. V kapitole 9.4 je pak zkoumán vliv druhé iterace HNM na hodnoty zkoumaných metrik.

9.2.1 Výsledky

Pro ohodnocení výsledků následujícího testování byly využívány dvě důležité míry, kterými se zabývají Ruskó a Perényi ve své práci [42]:

- TPR (z anglického True Positive Rate) je ekvivalentní k recall a vyjadřuje senzitivitu metody:
$$\mathrm{TPR} = \mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{9.4}$$
kde TP, resp. FN, značí celkový počet TP, resp. FN, pro celou testovací množinu.

- FPC (z anglického False Positive per Case):
$$\mathrm{FPC} = \frac{\mathrm{FP}}{N}, \tag{9.5}$$
kde N je množství testovacích obrázků a FP je počet falešných detekcí v nich.

Počítání TP, FP a FN probíhá podle následujících pravidel:

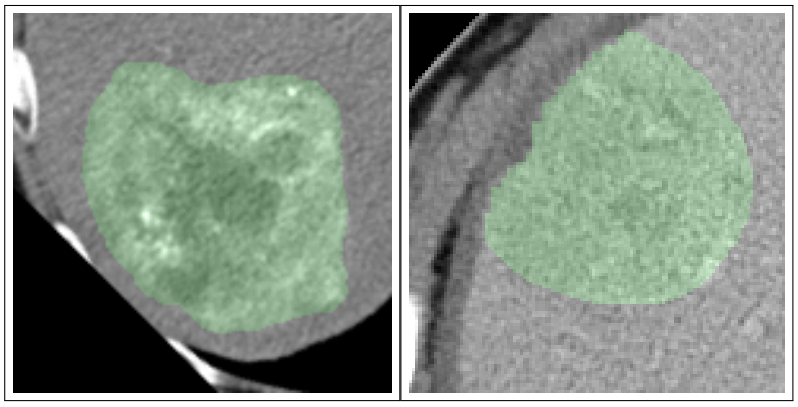
1. Pro každou anotovanou lézi se zkoumá, zda se překrývá s nějakým ohraničujícím obdélníkem, přičemž o překrytí se jedná, pokud jsou splněny obě následující podmínky:
   (a) Léze je alespoň z 50% pokryta ohraničujícím obdélníkem.
   (b) Tento ohraničující obdélník je alespoň ze 40% pokryt lézí.

   Pokud jsou všechny výše uvedené podmínky splněny, počet TP vzroste o 1. V opačném případě se jedná o FN.

2. Všechny ostatní detekované ohraničující obdélníky, které se nepřekrývají s žádnou anotovanou lézí, se označí jako FP.

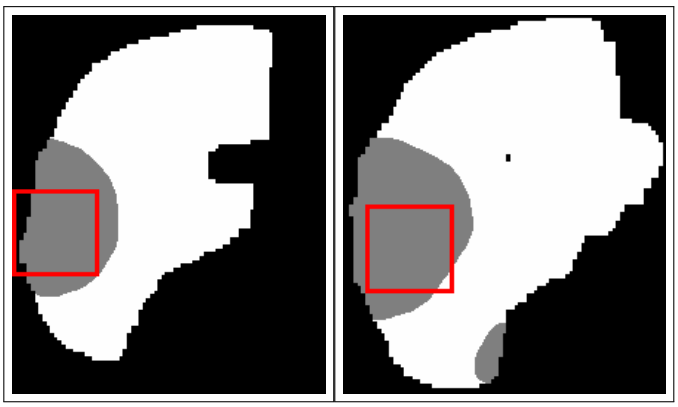
Prahová hodnota 50% byla inspirována hodnotou IoU_{min} využitou u PASCAL výzvy [43] (viz dále kapitola 10.3.2). IoU_{min} má sice poněkud jiný význam, ale na základě pozorování byla stejná hodnota zvolena jako vhodná i pro tento práh, tedy minimální pokrytí léze ohraničujícím obdélníkem.

Ukázky procentuálních pokrytí obdélníku lézí jsou k vidění na obrázku 9.5. Prahová hodnota 40% minimálního zastoupení léze v ohraničujícím obdélníku byla zvolena empiricky. Ukázky procentuálních pokrytí léze obdélníkem jsou k vidění na obrázku 9.6. Zajímavý je také graf závislosti TPR na hodnotě minimálního zastoupení léze v ohraničujícím obdélníku, který je vykreslen na obrázku 9.7 a byl získán z výsledků pro nejlepší konfiguraci (viz obrázek 9.8).



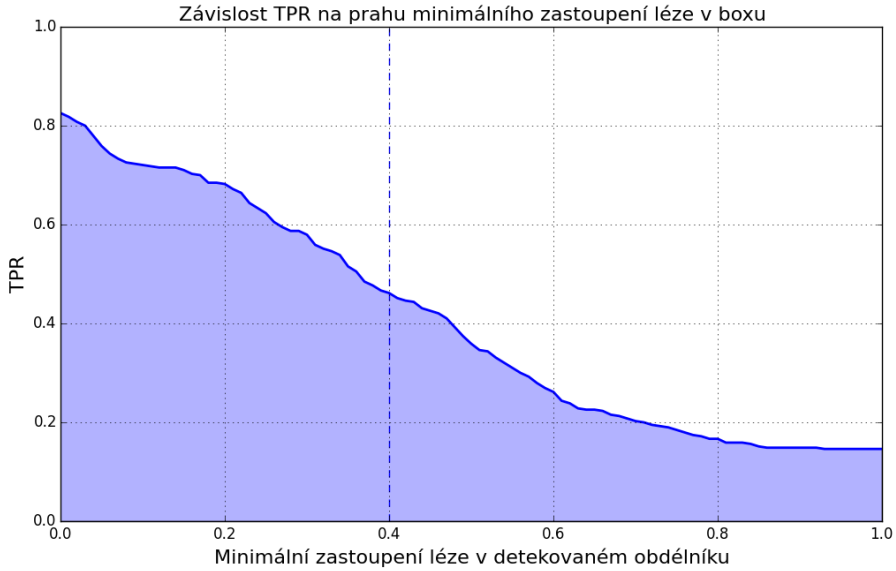
(a) (b)

Obrázek 9.5: Procentuální zastoupení léze, která je zvýrazněna zelenou barvou, uvnitř ohraničujícího obdélníku. Prahová hodnota pro posouzení obdélníku jako TP je 40%. Obrázek vlevo je ze 40.41% pokryt lézí a tuto podmínku splňuje. Na obrázku vpravo je zastoupení léze rovno 36.11% a podmínku pro označení jako TP nesplňuje.

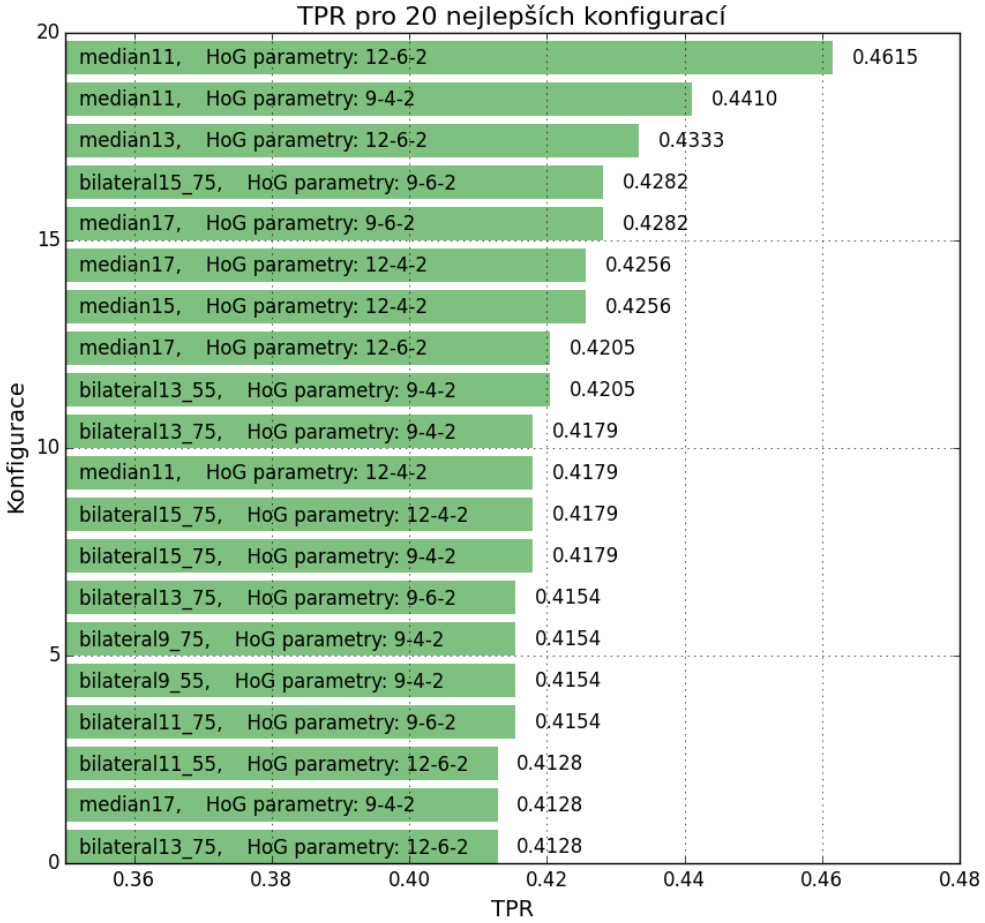


(a) (b)

Obrázek 9.6: Ukázka procentuálního pokrytí léze ohraničujícím obdélníkem. Bílá barva znázorňuje zdravou jaterní tkáň, šedá lézi a černá barva představuje okolí. Prahová hodnota pro posouzení obdélníku jako TP je 50%. Léze na obrázku vlevo je z 55.40% pokryta obdélníkem a tuto podmínku splňuje. Na obrázku vpravo je zastoupení obdélníku v lézi rovno 46.00% a podmínku pro označení jako TP nesplňuje.

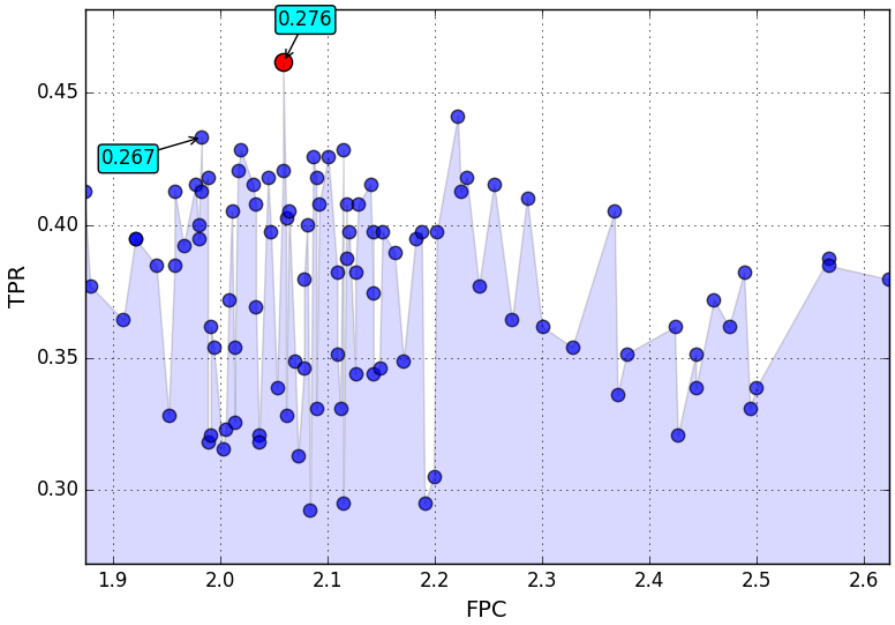


Obrázek 9.7: Graf závislosti TPR na hodnotě prahu minimálního pokrytí ohraničujícího obdélníku lézí



Obrázek 9.8: Znázornění 20 nejlepších konfigurací podle TPR. Popisek konfigurace je ve tvaru: typ a velikost filtru, HoG parametry (zleva: počet orientací histogramu, velikost buňky, počet buněk v blocích). Druhé číslo u bilaterálního filtru značí hodnotu parametrů σ_r a σ_d, která byla nastavována vždy pro oba parametry stejně.

Jak je možné vyčíst z grafu na obrázku 9.8, nejvyšší hodnotu TPR = 0.4615 vykazovala konfigurace se zpracováním obrazu mediánovým filtrem o velikosti jádra 11 pixelů. Počet orientací gradientu byl nastaven na 12, velikost buňky na 6 pixelů a počet buněk v bloku na 2. Přestože toto nastavení vykazovalo výrazně vyšší hodnotu TPR než ostatní, nedosahovalo nejnižší hodnoty FPC, jak dokazuje graf na obrázku 9.9, kde jsou vykreslené hodnoty FPC a TPR pro testované konfigurace.



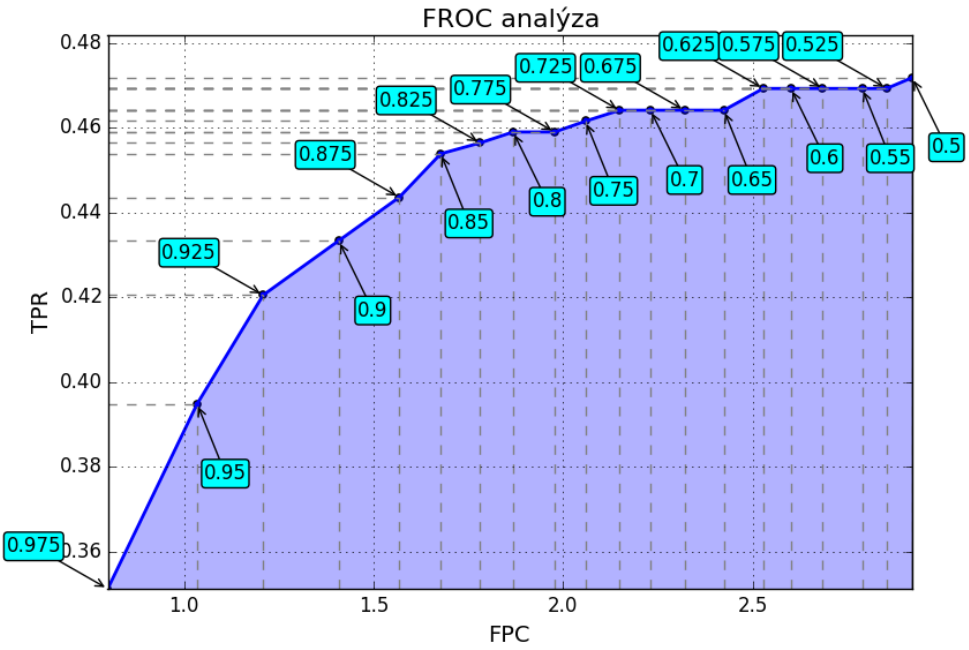
Obrázek 9.9: Vykreslení hodnot TPR a FPC pro různé konfigurace. Červená barva značí konfiguraci s nejlepší TPR. Popisek obsahuje hodnotu F-míry pro daný experiment.

Při pohledu na graf na obrázku 9.9 si lze všimnout mnoha konfigurací s nižší hodnotou FPC, než vykazovala ta s nejvyšší TPR.

Pokud je minimální pravděpodobnost detekce zvýšena, lze očekávat pokles hodnoty FPC. Rovněž ale lze předpokládat také snížení míry TPR. Bylo tedy vhodné zkoumat, jak se budou vyvíjet míry TPR a FPC při změnách prahové hodnoty minimální pravděpodobnosti detekce, čímž se zabývá kapitola 9.3.

9.3 FROC analýza

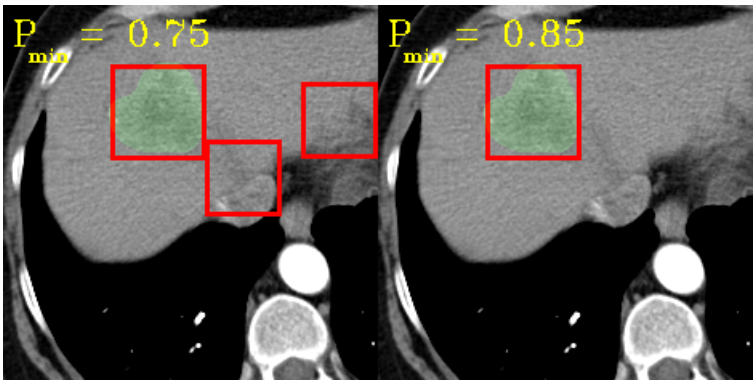
Až doposud platilo, že byly detekovány pouze takové regiony, které byly klasifikátorem ohodnoceny jako pozitivní s pravděpodobností $P \geq P_{min} = 0.75$. Tato kapitola se bude zabývat výběrem optimální hodnoty P_{min}, k čemuž byla využita FROC analýza (z anglického Free response Receiver Operating Characteristic [44]), kterou se zabývají Ruskó a Perényi ve své práci [42]. FROC analýza slouží k zobrazení vývoje TPR a FPC při měnící se hodnotě prahové pravděpodobnosti P_{min}. Lze předpokládat, že při zvyšující se P_{min} bude docházet ke snížení počtu detekovaných ohraničujících obdélníků a pravděpodobně tedy i počtu TP a FP, což je potvrzeno na obrázku 9.10.



Obrázek 9.10: FROC analýza pro nejlepší testovanou konfiguraci ve smyslu TPR. Bodové popisky grafu ukazují hodnotu P_{min}, pro kterou byl výsledek získán.

Z grafu na obrázku 9.10 je vidět rostoucí charakter křivky. Při výběru optimální hodnoty P_{min} je vhodné zkoumat kompromis mezi hodnotami FPC a TPR. V kapitole 9.2 bylo prováděno testování při hodnotě $P_{min} = 0.75$. Z grafu lze určit, že po zvýšení této hodnoty na $P_{min} = 0.85$ dojde k lehkému snížení TPR, ale také k významnému poklesu hodnoty FPC. Další zvýšení P_{min} by už vedlo k příliš vysokému poklesu TPR. Optimální hodnotou je tedy $P_{min} = 0.85$, při které bylo dosaženo TPR = 0.4538 a FPC = 1.6770. Takováto konfigurace vykazuje nejlepší hodnotu TPR i FPC napříč testovanými konfiguracemi, což si lze ověřit i z grafu na obrázku 9.9, kde jsou tyto hodnoty zaneseny. Toto nastavení tedy může být označeno za optimální.

Na obrázku 9.11 je k vidění řez, kde prahová hodnota pravděpodobnosti detekce $P_{min} = 0.75$ vedla k chybnému nalezení dvou ohraničujících obdélníků s pravděpodobnostmi 0.8413 a 0.8128, že se jedná o pozitivní rámec. Změna hodnoty prahu na $P_{min} = 0.85$



Obrázek 9.11: Ukázka detekovaných ohraničujících obdélníků v jednom z řezů při rozdílných hodnotách P_{min}. Zelená barva zvýrazňuje anotovanou lézi, červená značí hranice detekovaného ohraničujícího obdélníku.

způsobí zařazení těchto dvou obdélníků mezi negativní, což je vidět na obrázku 9.11 vpravo, kde tyto false positives chybí a výsledný obrázek obsahuje jen správně klasifikovaný true positive. Tento příklad skvěle ukazuje, jak zvýšení prahu minimální pravděpodobnosti detekce vede ke snížení skóre FPC.

9.4 Vliv Hard Negative Mining na úspěšnost klasifikátoru

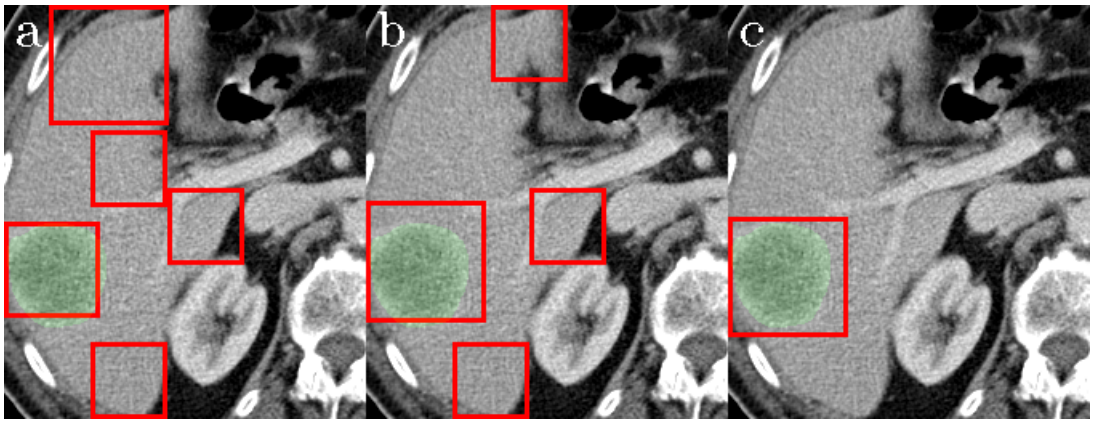
V předchozích kapitolách bylo vybráno optimální nastavení ve smyslu TPR pro detekci jaterních lézí pomocí HoG deskriptorů, kde počet orientací gradientu byl nastaven na 12, jako velikost buněk byla vybrána hodnota 6 pixelů a jejich počet v každém bloku byl stanoven na 2. Jako optimální se ukázalo zpracování obrazu pomocí mediánového filtru o velikosti jádra 11 pixelů.

V této části byla provedena metoda Hard Negative Mining pro zbylé trénovací řezy, a to jak pro negativní, tak i pro ty, které obsahují léze. Hodnota $P_{min}(HNM)$ byla nastavena podle optimální hodnoty P_{min}, zvolené na základě FROC analýzy. Tím mělo podle očekávání dojít k eliminaci FP, a tedy ke snížení hodnoty FPC.

Hard Negative Mining	TPR	FPC
bez HNM	0.2641	2.4663
pro 50 řezů s $P_{min}(HNM) = 0.5$	0.4538	1.6770
pro ostatní řezy s $P_{min}(HNM) = 0.85$	**0.4564**	**0.9073**

Tabulka 9.6: Výsledky pro jednotlivé způsoby aplikace HNM. Poslední řádek prvního sloupce tabulky znamená, že byla provedena metoda HNM pro prvních 50 řezů s největším zastoupením jater a po přetrénování bylo provedeno totéž pro všechny zbylé trénovací řezy s $P_{min}(HNM) = 0.85$.

Z tabulky 9.6 je patrné, že se předpoklad poklesu FPC pro Hard Negative Mining provedené na větším počtu snímků potvrdil, což lze pozorovat i na obrázku 9.12. Výsledek zcela bez provedení Hard Negative Mining (obrázek 9.12a) ukazuje vysoký počet false positives. Dva další snímky ukazují výsledné obdélníky za použití HNM, přičemž u prvního obrázku 9.12b bylo spuštěno HNM pro prvních 50 řezů s největším zastoupením jater. Zde je k vidění o jeden false positive méně. Dále byl klasifikátor přetrénován a následovalo Hard Negative Mining pro zbytek trénovacích řezů, což vedlo k výsledku na obrázku 9.12c, kde se stejně jako v tabulce 9.6 potvrdil výrazný pokles počtu false positives, a tím pádem také FPC. Také hodnota TPR se v tomto případě mírně zvýšila. Po nahlédnutí do grafu na obrázku 9.8 lze zjistit, že by se tato konfigurace zařadila na 2. místo ve smyslu nejvyšší hodnoty TPR. Takovéto další přetrénování by tedy mohlo být jedním ze způsobů, jak značně eliminovat počet false positives, aniž by klesla hodnota TPR.



Obrázek 9.12: Ukázka ohraničujících obdélníků pro jeden z řezů. Zelená barva znázorňuje anotovanou lézi, červená pak hranice detekovaného ohraničujícího obdélníku. Zleva: výsledek bez HNM, výsledek s HNM pro 50 řezů s $P_{min}(HNM) = 0.5$ a totéž navíc pro ostatní řezy s $P_{min}(HNM) = 0.85$.

Jako nejlepší model využívající histogram orientovaných gradientů jako extraktor vektoru příznaků a SVM jako klasifikátor byl stanoven ten, který byl natrénován s nejlepší konfigurací určenou v kapitole 9.2.1 a následně přetrénován po dalším spuštění HNM. Výsledné metriky pro vybraný nejlepší model jsou k vidění v tabulce 9.6.

Kapitola 10

CNN experimenty

Pro realizaci experimentů s konvolučními neuronovými sítěmi byl použit jazyk Python. Pro vytváření architektury konvolučních neuronových sítí a práci s metodami strojového učení byla využita knihovna Keras [32]. Jako nástroj pro práci s obrazovými daty sloužila knihovna OpenCV [7].

10.1 Příprava dat

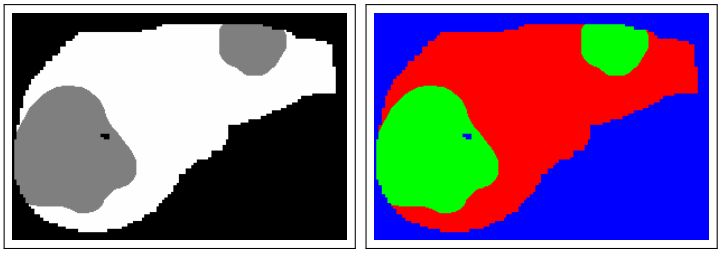
Jak již bylo popsáno v kapitole 8.2, pro experimenty s konvolučními neuronovými sítěmi byla použita obrazová data upravená tak, aby vše, co se vyskytuje mimo segmentovanou jaterní tkáň, bylo přebarveno na nulovou hodnotu. Od sítě tak bylo očekáváno, že se velice rychle naučí rozpoznávat okolí, tedy černou barvu, a dále se bude koncentrovat především na detekci lézí uvnitř jater.

Následně byla data augmentována pomocí geometrické afinní transformace s využitím různých parametrů zkosení a rotace. Jako parametry rotace byly použity hodnoty $\theta \in \{0, 1, 2, ..., 8\}$, které jsou udávány ve stupních. Hodnoty zkosení byly vybírány z množiny $\{0, 0.1, 0.2\}$. Objem dat pro tuto úlohu se tím zvětšil 27krát oproti původní velikosti bez augmentace. Důvod, proč oproti SVM byla datová množina mnohem více rozšířena, je ten, že u neuronové sítě bylo nutné natrénovat až 30 milionů parametrů, k jejichž natrénování bylo zapotřebí více různých dat. Rovněž tím lze předejít hrozícímu přetrénování.

Problémem, který bylo zapotřebí v této úloze vyřešit, byly rozměry jednotlivých 2D řezů. V první řadě bylo nutné zajistit, aby byly pro všechny obrázky stejné. Dále navíc musely rozměry dat vyhovovat architektuře sítě. Byla zde totiž využívána max-pooling vrstva o velikosti 2×2 s krokem 2, tedy takovým, aby nedocházelo k překrytí. Jak je možné si ověřit v kapitole 7.1, kde je princip operace max-pooling popsán, výška i šířka obrazu v následující vrstvě se sníží na polovinu. Je tedy nutné, aby jak výška, tak i šířka vstupního obrazu byla dělitelná číslem 2^n, kde n je počet takovýchto použitých max-pooling vrstev v architektuře. Pokud by tato podmínka splněna nebyla, mohlo by dojít k případu, kdy by na vstup max-pooling vrstvy vstoupila mapa příznaků o výšce $h = 31$ prvků. Výstupem max-pooling vrstvy by pak byla mapa příznaků o výšce $h = 15$ prvků.

Po následném převzorkování, tedy na výstupu upsampling vrstvy, by měla mapa příznaků výšku jen $h = 30$ prvků, což by vedlo k tomu, že by výstup sítě měl odlišné rozměry oproti anotaci.

Jako vhodná struktura anotovaných dat se ukázala forma one-hot vektorů, kde byly původní pixely obsahující jen index třídy, do které patří, nahrazeny jinak reprezentovanými pixely, z nichž každý obsahoval vektor o velikosti rovné počtu tříd, který na místě indexu třídy obsahoval hodnotu 1 a hodnotu 0 jinde. Taková data bylo následně možné porovnávat s výstupy sítě, kde pixely obsahovaly stejný vektor, ale s hodnotami, které značily pravděpodobnost náležitosti pixelu do příslušné třídy. Oba způsoby reprezentace masky jsou znázorněny na obrázku 10.1



(a) (b)

Obrázek 10.1: Ukázka upravení masky obrazu. Na obrázku vlevo nabývají pixely tří hodnot, kde bílá barva značí zdravá játra, šedá barva léze a černě je obarveno pozadí. Na obrázku vpravo nabývají pixely hodnoty 1 na indexu třídy, do které náleží, a 0 jinde. Takovou masku lze velice přehledně vykreslit jako RGB obrázek, kde červená barva označuje játra, zelená barva označuje léze a modré je pozadí.

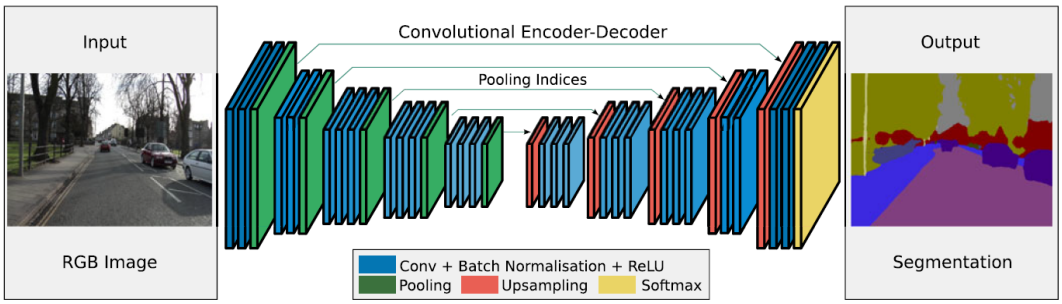
Nakonec bylo zapotřebí rozdělit data na trénovací, validační a testovací. Validační data nebyla síti předávána během jejího trénování, ale sloužila pro ohodnocení modelu na konci každé epochy, kdy byla spočtena hodnota kriteriální funkce a metriky. Díky tomu bylo možné sledovat, zda je síť schopna klasifikovat neznámá data a zda nedošlo k přetrénování. K tomu by došlo v případě, že by s rostoucí hodnotou metriky vyhodnocené na trénovací sadě klesala hodnota stejné metriky vyhodnocené na sadě validační. Jinými slovy by to znamenalo, že se síť dobře učí reagovat pouze na viděné vstupy, ale ne na data, která jí nebyla během trénování dodána. Testovací data pak byla použita až pro ohodnocení natrénované sítě různými metrikami, kterými se zabývá kapitola 10.3. Data byla rozdělena na trénovací, validační a testovací v poměru $3 : 1 : 1$.

10.2 Architektura

Zatímco bylo u přístupu HoG + SVM prováděno testování pro různé parametry metody histogramu orientovaných gradientů a také pro různé metody předzpracování obrazu (viz kapitola 9), v této úloze byly pro změnu testovány různé architektury sítě. Všechny však vycházely z architektury SegNet popsané v práci [31].

10.2.1 SegNet

Byla využívána architektura sítě typu enkodér-dekodér, která byla popsána v kapitole 7.2. Konkrétně byla inspirována architekturou SegNet, která je popsána v práci [31] a ilustrována na obrázku 10.2. SegNet se skládá z části enkodéru a odpovídající části dekodéru. Na



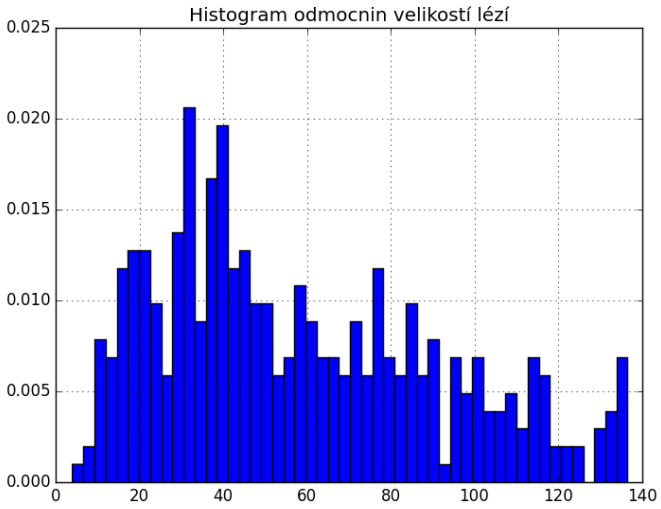
Obrázek 10.2: Ilustrace architektury SegNet, která byla převzata z práce [31].

výstup dekodéru je napojena klasifikační vrstva, která provádí zařazení pixelů do jednotlivých tříd. Enkodér se v základní architektuře SegNet skládá z několika na sebe navazujících bloků. Tyto bloky jsou tvořeny posloupností konvolučních vrstev s aktivační funkcí ReLU, z nichž na každou je aplikována dávková normalizace, a následnou max-pooling vrstvou, která zvyšuje velikost zorného pole a zároveň snižuje velikost mapy příznaků v dalších vrstvách. Na výstup každé konvoluční vrstvy je navíc aplikována dávková normalizace.

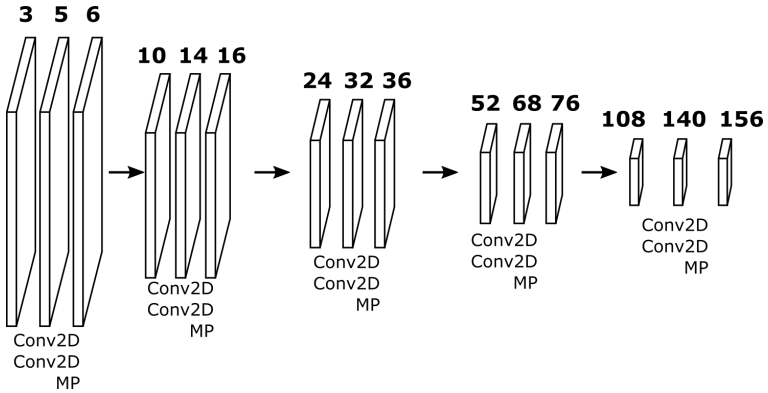
10.2.2 Návrh architektury

Architektura využívaná v této práci obsahuje jeden enkodér a jeden jemu odpovídající dekodér, čímž se liší od architektury SegNet-basic, zmíněné v práci [31], která obsahuje čtyři enkodéry a stejný počet dekodérů. Zároveň byla architektura zjednodušena tak, že každý blok obsahuje jen dvě za sebou jdoucí konvoluční vrstvy, na rozdíl od architektury znázorněné na obrázku 10.2, kde některé bloky obsahují například tři konvoluční vrstvy. Jádra všech konvolučních vrstev byla nastavena na velikost 3×3. U max-pooling vrstev byla zvolena velikost okna 2×2 a velikost kroku nastavena na 2. To znamená, že došlo k dvojnásobnému zvýšení výšky a šířky mapy příznaků v další vrstvě. Vlastnost, která byla zkoumána primárně, byla velikost zorného pole v jednotlivých vrstvách. Počet jader v jednotlivých konvolučních vrstvách byl v každém následujícím bloku zdvojnásoben. První blok obsahoval konvoluční vrstvy o 32 jádrech, v posledním pátém bloku enkodéru pak bylo přítomno 512 jader konvoluční vrstvy.

Během návrhu architektury byla nejdříve zkoumána její hloubka. Bylo požadováno, aby velikost zorného pole na výstupu enkodéru zhruba odpovídala rozměrům největších lézí obsažených v anotovaných datech. Histogram odmocnin velikostí obdélníků ohraničujících anotované léze, které by zhruba měly odpovídat jejich výškám a šířkám, je k vidění na obrázku 10.3. Vzhledem k nastaveným parametrům jednotlivých vrstev, popsaným v předešlém odstavci, by velikost zorného pole na výstupu jednotlivých bloků architektury nabývala hodnot, které jsou znázorněny na obrázku 10.4, kde jsou udávány nad příslušnými vrstvami enkodéru, který byl popsán v odstavci výše.



Obrázek 10.3: Histogram odmocnin velikostí lézí obsažených v datech



Obrázek 10.4: Znázornění vrstev enkodéru, nad nimiž je udána velikost zorného pole jejich výstupu.

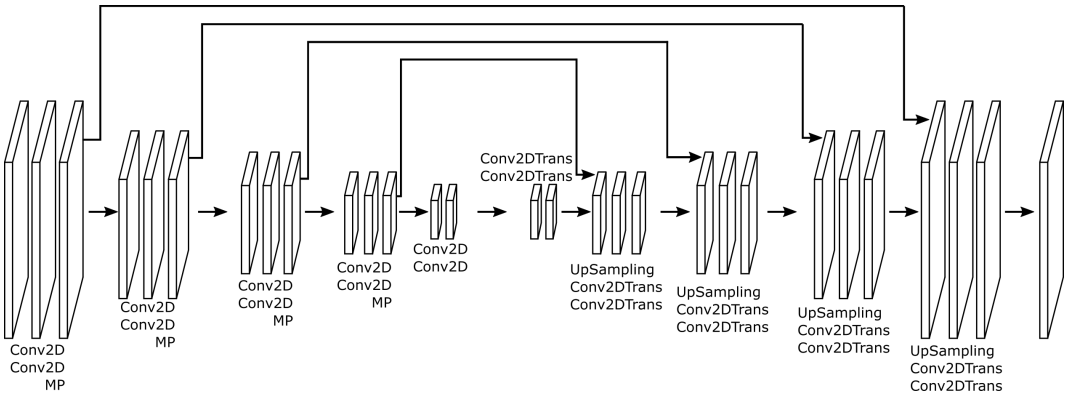
Jak je možné vyčíst z obrázku 10.4, optimální by bylo zvolit buď 4, nebo 5 bloků skládajících se ze dvou konvolučních vrstev a jedné max-pooling vrstvy. Pokud by byly zvoleny pouze 4 bloky, na výstupu enkodéru by nabývalo zorné pole velikosti pouze 76×76. Výstup pátého bloku už ale má zorné pole o velikosti 156×156. Po vynechání max-pooling

vrstvy v posledním pátém bloku by výstup enkodéru odpovídal výstupu druhé konvoluční vrstvy v pátém bloku, jenž disponuje zorným polem o velikosti 140×140, která již odpovídá požadované hodnotě. Následně byl navržen odpovídající dekodér, kde analogicky došlo k vynechání upsampling vrstvy prvního bloku. Ostatní bloky dekodéru pak byly složeny z upsampling vrstvy, za kterou následovaly dvě dekonvoluční vrstvy. Všechny parametry vrstev dekodéru, tedy například velikosti a počty jader, velikost kroku či aktivační funkce, byly převzaty z odpovídajících vrstev enkodéru.

Výstup enkodéru tedy disponoval zorným polem srovnatelným s maximálními velikostmi lézí obsažených v anotovaných datech. Toto zorné pole bylo nadále převzorkováno dekodérem a propagováno na výstup sítě. Snahou však bylo propagovat na výstup také zorná pole o rozměrech, které by odpovídaly velikostem menších lézí.

U přístupu HoG a SVM byla detekce lézí různých velikostí umožněna vytvořením pyramidy obrazu (viz kapitola 9.2), kde docházelo k detekci ohraničujících obdélníků v různých měřítkách. U architektury sítě enkodér-dekodér by mohla být detekce lézí různých rozměrů provedena propagací takových zorných polí, která by jejich rozměrům svou velikostí odpovídala, na výstup sítě. Jako realizace tohoto přístupu se nabízelo připojit výstupy vrstev, které vykazovaly zorné pole o požadované velikosti, na vstupy odpovídajících vrstev dekodéru. Tím došlo k přeskočení hlubších vrstev sítě, což je princip, který byl navržen v práci [45], kde jsou tato přeskočení nazývána anglicky shortcut connections, případně skip connections, jak je uváděno v článku [46], což je název, který byl použit i v této práci. Do každé vrstvy dekodéru je však již přiváděn výstup předcházející vrstvy. Připojení výstupu vrstvy enkodéru s požadovaným zorným polem bylo realizováno pomocí konkatenace se všemi vstupy, které do cílové vrstvy dekodéru vstupují. Až struktura vzniklá takovýmto zřetězením pak vstupovala do požadované vrstvy dekodéru, přičemž byla tvořena z částí s odlišnými velikostmi zorného pole. Na výstup sítě tak bylo propagováno nejen zorné pole výstupu celého enkodéru, ale také zorná pole všech vrstev, které byly takto připojeny na vstup jim odpovídajících vrstev dekodéru. Tím byla umožněna lepší klasifikace oblastí o rozměrech všech propagovaných zorných polí.

Pro posouzení, od jakých vrstev vést skip connections, byly porovnány obrázky 10.3, kde je k vidění histogram rozměrů lézí, a 10.4, kde jsou znázorněny velikosti zorného pole výstupů jednotlivých vrstev enkodéru. Nakonec bylo zvoleno vedení skip connections od každé max-pooling vrstvy enkodéru k odpovídající upsampling vrstvě dekodéru, což mělo v tomto případě za důsledek propagaci zorného pole o velikostech, které přibližně odpovídaly často se vyskytujícím rozměrům anotovaných lézí. Architektura sítě s takto použitými skip connections je vykreslena na obrázku 10.5, kde Conv2DTrans je označení pro dekonvoluční vrstvu podle knihovny Keras [32]. Podobný přístup, tedy přivádění výstupu každé max-pooling vrstvy na vstup odpovídající upsampling vrstvy, byl použit také u architektury sítě U-Net [47], která byla, stejně jako naše síť, inspirována architekturou SegNet [31].



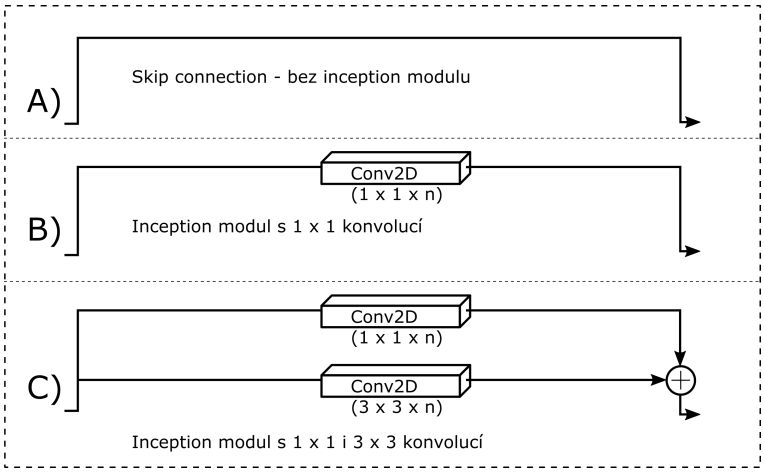
Obrázek 10.5: Architektura sítě se skip connections

Podobný princip pro zachycení informace z odlišných měřítek byl využit v práci [48], kde ale skip connection zahrnuje navíc konvoluční vrstvu o velikosti jader 1×1. Tento způsob byl inspirován tzv. inception architekturou, která byla popsána v práci [49].

Využití velikosti jádra 1×1 nezpůsobuje změnu velikosti zorného pole, ale obraz je transformován pomocí konvolučních jader, jejichž parametry jsou trénovány. Jedním z cílů experimentů bylo prozkoumat, zda takováto transformace povede ke vhodnější reprezentaci obrazu, na kterou by síť lépe reagovala.

Další rozšíření bylo inspirováno principem naivního inception modulum který byl popsán v práci [49], kde bylo na výstup předchozí vrstvy aplikováno několik konvolucí s jádry odlišných velikostí, jejichž výstupy byly před vstupem do následující vrstvy sítě zřetězeny. Uvedený přístup byl v naší architektuře využit ve zjednodušené formě, a to tak, že inception modul obsahoval pouze dvě konvoluční vrstvy o velikostech jader 1×1 a 3×3. Bylo tedy zapotřebí vést od výstupů požadovaných vrstev hned dvě větve, z nichž jedna by zahrnovala konvoluční vrstvu s velikostí jádra 1×1, která byla popsána výše. V konvoluční vrstvě druhé větve by pak byla velikost jádra rovna 3×3. Následně byly výstupy vrstev v obou větvích zřetězeny. Jelikož bezprostředně poté následovala další konkatenace, konkrétně s vrstvou dekodéru, mohla být pro zjednodušení provedena až konkatenace všech tří výstupů najednou. Přidání další větve s konvoluční vrstvou o velikosti jádra 3×3 mělo přínos zejména v propagaci většího počtu různých zorných polí, jenž se po konvoluci 3×3 zvyšují, na výstup sítě. Mohlo by se zdát, že jelikož je inception modul napojen na max-pooling vrstvu, na kterou v enkodéru navazuje konvoluční vrstva s jádrem velikosti 3×3, je větev inception modulu s konvoluční vrstvou o stejné velikosti jader zbytečná, protože by mohl být její výstup nahrazen výstupem vrstvy enkodéru. Rozdíl je ale v tom, že parametry vrstev základního enkodéru jsou během trénování nastavovány tak, aby detekovaly objekty odpovídající svou velikostí zornému poli celého enkodéru, tedy 140×140. Vrstvy inception modulů se oproti tomu učí propagovat na výstup sítě informace z jiných měřítek a přináší tak informaci navíc. Ukázka pro všechny tři popsané metody, tedy skip connection a dva typy inception modulů, je vidět na obrázku 10.6. Stojí za zmínku, že počet jader konvolučních vrstev v každém inception modulu byl vždy nastaven tak, aby se rovnal

hloubce mapy příznaků výstupu vrstvy enkodéru, na níž byl modul napojen. Použití inception modulů má za následek další zvýšení počtu parametrů, které je třeba natrénovat, čímž se zvyšují jak paměťové nároky, tak i doba trénování. Také proto nebylo na výstup sítě propagováno zorné pole z každé vrstvy enkodéru, ale jen z každé max-pooling vrstvy.



Obrázek 10.6: Ukázka tří různých testovaných přístupů. Pro představu, jak v jednotlivých případech vypadá architektura, lze jednoduše zaměnit skip connections z architektury na obrázku 10.5 za vybraný modul. Počet jader n v konvolučních vrstvách je nastaven tak, aby odpovídal hloubce mapy příznaků vstupu inception modulu.

Stejně jako u enkodéru byla i ve všech inception modulech jako aktivační funkce konvolučních vrstev nastavena ReLU a na jejich výstup byla aplikována dávková normalizace.

Dále bylo zkoumáno, zda by nebylo možné dosáhnout lepších výsledků přidáním konvolučních vrstev mezi poslední vrstvu dekodéru a finální klasifikační vrstvu a naučit tak síť některé morfologické operace, jako je například odfiltrování malých objektů. Byly zvoleny dvě za sebou jdoucí konvoluční vrstvy s 32 jádry o velikosti 3×3.

Byly tedy testovány tři různé druhy architektury, které se lišily pouze v tom, zda obsahovaly jen skip connections, nebo inception moduly, případně zda tyto inception moduly obsahovaly konvoluční vrstvy s velikostí jader 1×1, nebo i 3×3. Pro každou z nich bylo navíc ještě testováno připojení dvou konvolučních vrstev mezi výstup dekodéru a finální výstup sítě. Nakonec tedy počet testovaných architektur stoupl na 6.

10.3 Hodnotící metody

10.3.1 Přesnost per pixel

V první řadě je důležité si uvědomit, že se zde jedná o per pixel detekci a ne o detekci obdélníků, jak tomu bylo u přístupu HoG a SVM. Natrénovaný model v knihovně Keras [32] lze ohodnotit jednoduchou již implementovanou metodou, která je schopna spočítat přesnost (anglicky accuracy). Tu si lze představit jako pravděpodobnost, že pixel bude správně klasifikován. V testovacích datech mají nádorové pixely zastoupení pouze 1.86%. Zbylých 98.14% tvoří buď zdravá jaterní tkáň, nebo pozadí. Pokud by byly všechny nádorové pixely vyhodnoceny sítí jako zdravá játra a všechny pixely pozadí byly vyhodnoceny správně, byla by hodnota přesnosti rovna 98.14%, která by sice budila dojem, že se jedná o relativně dobrý výsledek, avšak z hlediska této úlohy, tedy detekce lézí, by byly výsledky zcela nevyhovující. Proto byla místo celkové pravděpodobnosti správné detekce počítána podmíněná pravděpodobnost:

$$P\left(\omega_j|\Omega_i\right) = \frac{1}{N_i}\sum_{n=1}^{N}\sum_{q\in Q_i} I_j\left(q\right), \tag{10.1}$$

kde Q_i je množina všech pixelů náležících ve skutečnosti do třídy Ω_i, $I_j\left(q\right)$ je hodnota jasové funkce pixelu q výstupu sítě, tedy sítí predikovaná pravděpodobnost, že pixel q náleží do třídy ω_j, N je počet testovacích obrazů a N_i je počet pixelů náležících ve skutečnosti do třídy Ω_i napříč všemi testovacími daty. Výstup sítě rovněž mohl být převáděn na one-hot vektory, kde vektor $\vec{\mathbf{v}}$ tří pravděpodobností náležitosti pixelu do daných tříd byl transformován na vektor obsahující hodnotu 1 na pozici maxima původního vektoru a 0 na zbylých pozicích, například:

$$\vec{\mathbf{v}} = (0.1,\ 0.6,\ 0.3) \Rightarrow \vec{\mathbf{v}}_{one-hot} = (0,\ 1,\ 0)\,.$$

Lze si všimnout, že pro výstup sítě reprezentovaný ve formě one-hot vektorů udává výraz $P\left(\omega_j|\Omega_i\right)$ hodnotu pravděpodobnosti, že pokud pixel ve skutečnosti náleží do třídy Ω_i, bude sítí klasifikován do třídy ω_j. Indexy jednotlivých tříd pro tuto úlohu jsou zapsány v tabulce 8.1. Nabízelo se počítat především hodnotu $P\left(\omega_1|\Omega_1\right)$, která udává, jaká bude pravděpodobnost toho, že pokud je pixel jako léze anotován, bude sítí také jako léze vyhodnocen. Ve výše popsaném případě by byla tato pravděpodobnost rovna 0%, jelikož by zde byly všechny nádorové pixely označeny jako jaterní. Samozřejmě může dojít i k opačnému jevu, kdy by byly detekovány nádorové pixely po celé ploše jater. Zde by byla zmíněná podmíněná pravděpodobnost rovna $P\left(\omega_1|\Omega_1\right) = 1$, která by ale neudávala informaci o tom, zda jsou pixely v oblasti jater, kde se léze nevyskytují, klasifikovány správně. Bylo tedy vhodné zkoumat všechny možné podmíněné pravděpodobnosti. Jelikož jsou k dispozici 3 třídy, jedná se o celkem $3 \times 3 = 9$ podmíněných pravděpodobností, jejichž hodnoty byly zaneseny do matice přesnosti o rozměrech 3×3:

$$A = \begin{bmatrix} P\left(\omega_0|\Omega_0\right) & P\left(\omega_1|\Omega_0\right) & P\left(\omega_2|\Omega_0\right) \\ P\left(\omega_0|\Omega_1\right) & P\left(\omega_1|\Omega_1\right) & P\left(\omega_2|\Omega_1\right) \\ P\left(\omega_0|\Omega_2\right) & P\left(\omega_1|\Omega_2\right) & P\left(\omega_2|\Omega_2\right) \end{bmatrix}. \tag{10.2}$$

Je zcela evidentní, že čím přesnější bude klasifikace jednotlivých pixelů, tím budou prvky na diagonále matice přesnosti A nabývat vyšších hodnot neboli hodnot blížících se k číslu 1 zleva. Naopak prvky mimo diagonálu se budou snižovat směrem k nule. Matice přesnosti A byla použita k porovnávání výsledků jednotlivých experimentů. Jako způsob, jak matice pro jednotlivé výsledky mezi sebou porovnat, byl zvolen výpočet aritmetického průměru prvků na diagonále, tedy stopy matice vydělené počtem prvků na diagonále:

$$P = \frac{1}{3} \cdot \operatorname{trace}(A), \tag{10.3}$$

kde samozřejmě platí, že čím vyšší hodnoty bylo dosaženo, tím vykazoval experiment lepší výsledky.

Stejným způsobem mohl být proveden výpočet metriky s vynecháním transformace hodnot výstupu sítě na one-hot vektory. Pokud by v praxi bylo požadováno zachování modelem predikovaných pravděpodobností náležitosti pixelů do daných tříd, byl by tento způsob pro ohodnocení modelu vhodnější.

Jako další možné rozšíření by mohlo být zavedeno vážení jednotlivých prvků na diagonále matice A pro výpočet hodnoty metriky daný vztahem 10.3. Zde by se místo stopy matice A mohl počítat vážený součet prvků na její diagonále, přičemž hodnoty vah by zde závisely na tom, u které třídy je požadována správná detekce primárně. V této práci však byl využit pouze jednoduchý vztah 10.3 bez vážení jednotlivých prvků.

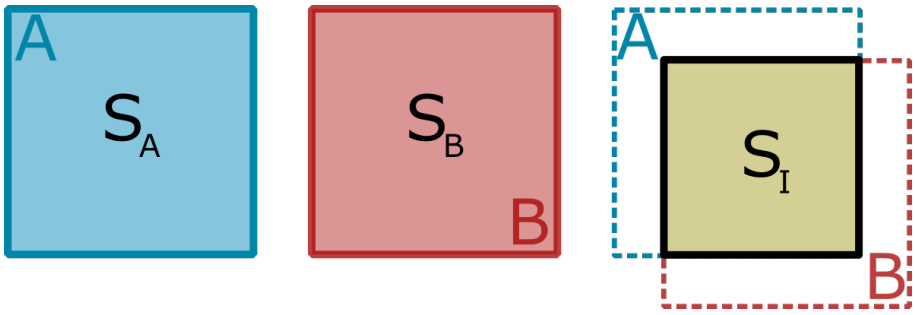
10.3.2 Jaccardův index

Přestože se jedná o plně per pixel úlohu, není problém získat ohraničující obdélníky anotovaných nebo detekovaných lézí, což vede k možnostem aplikovat hodnotící metody založené na ohraničujících obdélnících i na tuto úlohu. Tím by mohlo být umožněno v budoucnosti porovnávat výsledky podobných úloh, jež by byly založeny čistě na ohraničujících obdélnících, s výsledky experimentů v této práci.

Pokud jsou k dispozici detekované i anotované objekty, tedy v našem případě léze, ve formě ohraničujících obdélníků, nabízí se pro porovnání výsledků vyhodnocovat tzv. Jaccardův index, jinak označovaný jako IoU (z anglického Intersection over Union), který byl použit u PASCAL výzvy [43]. Hodnota IoU udává, do jaké míry se dva ohraničující obdélníky, označené například jako A a B, navzájem překrývají, neboli matematicky:

$$IoU\,(A,B) = \frac{S_{A\cap B}}{S_{A\cup B}} = \frac{S_{A\cap B}}{S_A + S_B - S_{A\cap B}}, \tag{10.4}$$

kde S_A, resp. S_B, je obsah ohraničujícího obdélníku A, resp. B, a $S_{A\cap B}$, resp. $S_{A\cup B}$, je obsah oblasti, která vznikne průnikem, resp. sjednocením, obdélníků A a B.



Obrázek 10.7: Ukázka překrytí dvou ohraničujících obdélníků. Z obsahů S_A, S_B a S_I se spočte hodnota IoU podle vzorce 10.4, kde je S_I ekvivalentní k výrazu $S_{A\cap B}$, tedy obsahu oblasti průniku obdélníků A a B.

Experimenty mohly být tedy ohodnoceny pomocí metrik recall a precision, jejichž výpočet se řídí podle rovnic 9.1 a 9.2. K tomu bylo zapotřebí získat počet TP, FP a FN, což bylo prováděno následujícím způsobem:

1. Pro každou anotovanou lézi ohraničenou obdélníkem A se zkoumá, zda existuje nějaká detekovaná léze ohraničená obdélníkem B taková, aby platilo:

   $$IoU\,(A,B) \geq IoU_{min}. \tag{10.5}$$

   Pokud je tato podmínka splněna, počet TP vzroste o 1.

2. Všechny ostatní detekované ohraničující obdélníky, které se s žádnou lézí nepřekrývají ve smyslu podmínky 10.5, se označí jako FP.

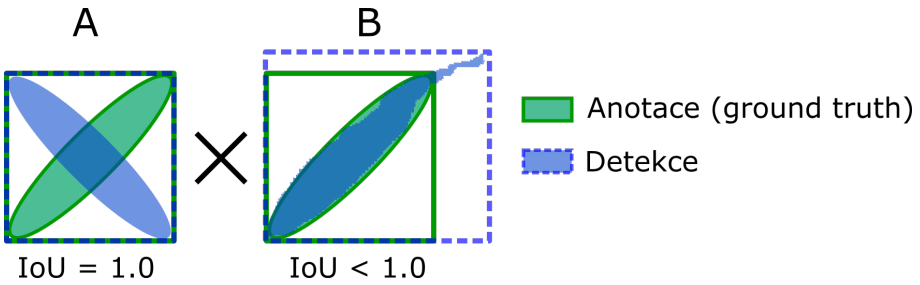
3. Všechny ostatní léze, pro které není podmínka 10.5 splněna, jsou označeny jako FN.

Pro počítání hodnot TP, FP a FN bylo potřeba stanovit si prahovou hodnotu IoU_{min}. Za tu byla zvolena, stejně jako u PASCAL výzvy [43], hodnota $IoU_{min} = 0.5$.

Tato hodnotící metoda pracuje s detekcemi i anotacemi ve formě ohraničujících obdélníků. My ale máme k dispozici detekce i anotace s rozlišením na pixely. Z výstupů sítě a masek obrazů, tedy anotací, bylo nutné extrahovat ohraničující obdélníky pro výpočet jejich vzájemného překrytí. Pro vytažení ohraničujících obdélníků z výstupu sítě bylo nutné provést navíc následující kroky:

1. rozhodnutí o zařazení pixelů do tříd, tedy získání vektoru $\vec{\mathbf{v}}_{one-hot} = (v_0,\ v_1,\ v_2)$, kde $v_{i^*} = 1 \Leftrightarrow i^* = \underset{i}{\operatorname{argmax}}\,(\vec{\mathbf{v}})$, kde $\vec{\mathbf{v}}$ je původní vektor,
2. aplikace morfologických operací - například odstranění malých detekovaných objektů, které jsou irelevantní.

Jelikož je tato úloha založena na per pixel detekci a k dispozici jsou rovněž per pixel anotace, není metoda ohodnocení pomocí metrik recall s využitím IoU vhodná pro porovnání výsledků experimentů, jelikož by převedením per pixel detekcí na ohraničující obdélníky došlo ke ztrátě informace. Místo toho byla využita metoda popsaná v kapitole 10.3.1. Problém, který může nastat u metody vyhodnocování IoU, je skvěle ilustrován na obrázku 10.8.



Obrázek 10.8: Ukázka dvou případů překrytí anotované léze s detekovanou. Již na první pohled lze usoudit, že detekce A bude mnohem méně odpovídat anotaci než detekce B, což by potvrdila i hodnota metriky 10.3 spočtená pro oba případy. Překrytí ve smyslu Jaccardova indexu bude však u varianty A nabývat hodnoty $IoU = 1$, tedy evidentně vyšší než v případě B.

10.4 Výsledky

Experimenty byly prováděny tak, že byla nejdříve modelu přiřazena architektura a optimalizační metoda a poté byl model natrénován. Následně byla využita testovací data pro ohodnocení takto natrénovaného modelu. Testovány byly následující architektury:

- architektura obsahující jen skip connections bez inception vrstev,
- architektura s inception vrstvami s velikostí jader $1 \times 1 \times n$,
- architektura obsahující inception vrstvy s velikostí jader $1 \times 1 \times n$ a s nimi spojené inception vrstvy o rozměru konvolučních jader $3 \times 3 \times n$.

U každého typu architektury byla navíc využita i varianta se dvěma přídavnými konvolučními vrstvami vloženými mezi poslední dekonvoluční vrstvu dekodéru a výstupní klasifikační vrstvu. Celkem tedy bylo testováno 6 rozdílných architektur. Pro každou architekturu byly provedeny tři experimenty, které se lišily v použité optimalizační metodě, z nichž byly testovány následující:

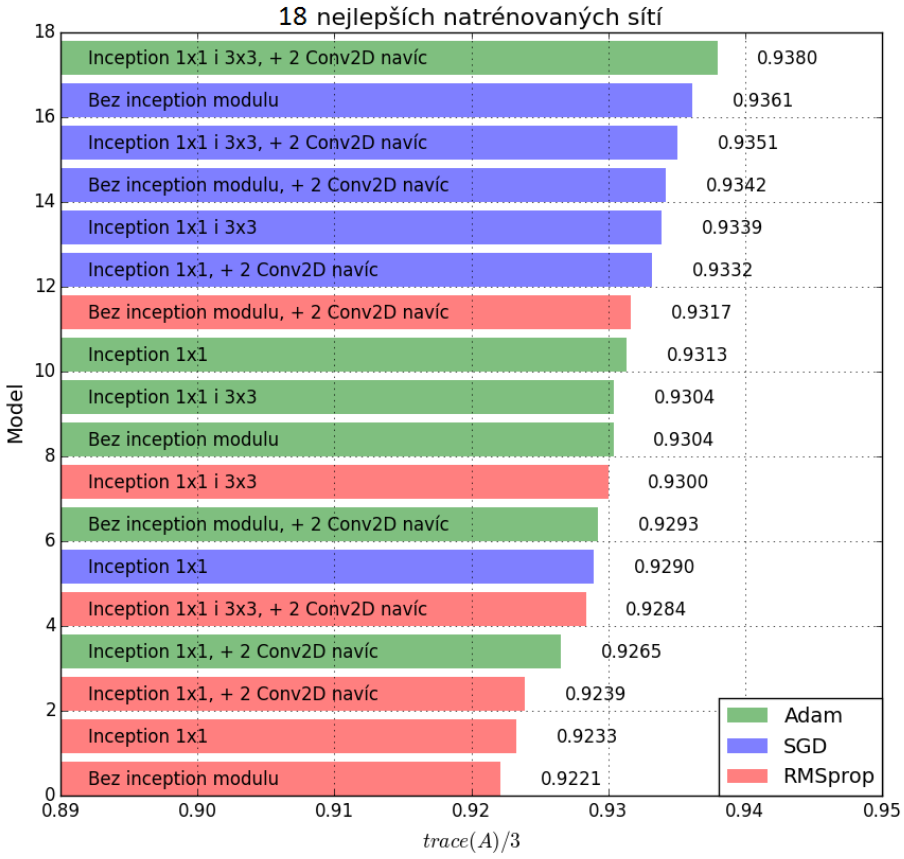
- SGD s využitím metody Nesterova momentu,
- RMSprop,
- Adam.

Bylo tedy prováděno 18 experimentů, jejichž výsledky, jak již bylo uvedeno v kapitole 10.3, byly porovnány na základě hodnoty výrazu 10.3. Během trénování bylo prováděno 10 epoch a jako kriteriální funkce byla využita kategorická křížová entropie. Jako metrika, podle které byl model hodnocen po každé epoše na validačních datech, byla použita přesnost (anglicky accuracy). Počáteční hodnota míry učení byla nastavena na $LR = 0.01$, která se v dalších epochách snižovala, a to tak, že na konci první, třetí a páté epochy byla $10\times$ zmenšena.

Sloupcový graf hodnot porovnávací metriky 10.3 pro všechny experimenty je k vidění na obrázku 10.9. Lze si všimnout, že nejlepší výsledek vykazovala nejsložitější architektura ze všech šesti testovaných, tedy s inception vrstvami s velikostí jader $1 \times 1 \times n$ i $3 \times 3 \times n$ a dvěma přídavnými konvolučními vrstvami na konci dekodéru, přičemž nejlepšího výsledku dosáhla při natrénování s optimalizátorem Adam. Pro ostatní architektury dosahovala metoda Adam průměrných výsledků v porovnání s ostatními optimalizátory.

Hodnota matice A pro nejlepší model byla rovna:

$$A = \begin{bmatrix} 9.9999 \cdot 10^{-1} & 3.5359 \cdot 10^{-7} & 4.8687 \cdot 10^{-6} \\ 3.7389 \cdot 10^{-5} & 8.2801 \cdot 10^{-1} & 1.7195 \cdot 10^{-1} \\ 3.6296 \cdot 10^{-5} & 1.4067 \cdot 10^{-2} & 9.8590 \cdot 10^{-1} \end{bmatrix}.$$



Obrázek 10.9: Porovnání výsledných hodnot hodnotící metriky 10.3 pro všechny experimenty. Architektura bez inception modulu značí, že bylo využito pouze skip connections.

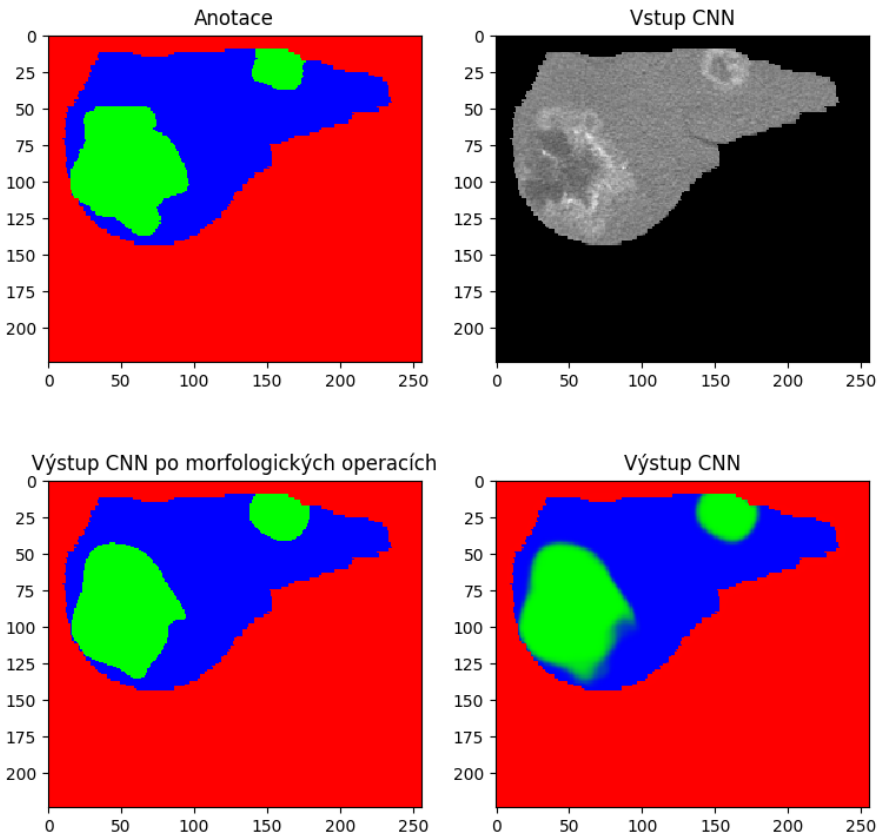
Pro nejlepší model byla vyhodnocena také míra recall, jejíž výpočet je popsán v kapitole 10.3.2, kde jsou rovněž uvedeny morfologické operace, které byly na výstupy modelu aplikovány před výpočtem metriky. Dva parametry těchto operací, a to velikost strukturálního elementu operace uzavření a minimální nutnou velikost objektu pro jeho zachování, bylo potřeba otestovat, aby byla hodnota recall maximalizována. Stojí za připomenutí, že v celé práci byla používána hodnota $IoU_{min} = 0.5$, která byla stanovena v kapitole 10.3.2. Na základě testování se jako optimální ukázala minimální velikost objektu 100 pixelů a velikost strukturálního elementu 3×3, kdy bylo dosaženo hodnot metrik, které jsou zaneseny do tabulky 10.1. Na obrázku 10.10 je k vidění ukázka výstupu sítě pro jeden z testova-

Míra	Hodnota
Recall (TPR)	0.6125
Precision	0.7226
FPC	0.1493

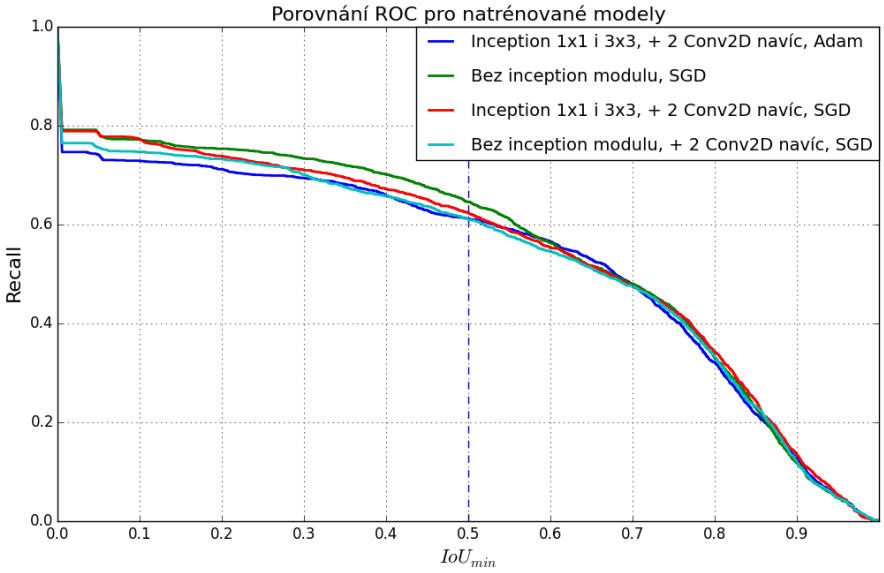
Tabulka 10.1: Hodnoty metrik pro optimální parametry morfologických operací. FPC, stejně jako v kapitole 9.2.1, udává počet false positives na jeden 2D řez.

cích snímků. Na první pohled lze usoudit, že v tomto případě došlo k poměrně přesné lokalizaci obou lézí v obraze. Porovnáním obou obrázků vlevo, tedy skutečné anotované masky a morfologickými operacemi upraveného výstupu sítě, lze dojít k závěru, že chybně predikované pixely se nacházely z velké části na okrajích objektů, což mohlo ovlivnit tvar ohraničujícího obdélníku příslušné léze a tím zvýšit odlišnost mezi jím a skutečným obdélníkem. Také snížení hodnoty porovnávací metriky 10.3 lze dát za vinu především nepřesnostem v okolí hran objektů. Vzhledem k obtížnosti samotné úlohy detekce lézí expertem by takovéto nepřesnosti byly očekávány dokonce i v případě, kdy by byly mezi sebou porovnávány dvě anotace vyprodukované odlišnými experty.

Pro zajímavost byly spočteny míry recall pro výsledky všech experimentů za použití těchto optimálních hodnot parametrů morfologických operací. Recall v závislosti na minimálním prahu překrytí obdélníků IoU_{min} je k vidění na obrázku 10.11, kde jsou kvůli přehlednosti vykresleny pouze ROC křivky prvních čtyř nejlepších modelů dle tabulky na obrázku 10.9.



Obrázek 10.10: Porovnání výstupu sítě a skutečné anotace. Modrá barva představuje zdravou jaterní tkáň, zelená lézi a červená barva značí pozadí. Vlevo dole je pak vykreslen obrázek vzniklý aplikací morfologických operací s výše určenými optimálními parametry.



Obrázek 10.11: Porovnání průběhů ROC křivky pro 4 nejlepší natrénované modely (viz graf na obrázku 10.9). Svislá přerušovaná čára slouží k odečtení hodnot míry recall, pokud je minimální překrytí nastaveno na $IoU_{min} = 0.5$.

Pro případ, že by bylo požadováno porovnání experimentů s jinými pracemi, kde by byly výsledky hodnoceny právě podle přístupu využívajícího Jaccardův index, byla u všech výsledků spočtena metrika recall, jejíž výsledné hodnoty pro jednotlivé experimenty jsou k vidění v tabulce 10.2, přičemž byla použita prahová hodnota překrytí obdélníků $IoU_{min} = 0.5$ (viz kapitola 10.3.2). Na druhém řádku se nachází nejlepší model ve smyslu maximalizace kritéria 10.3. Ani v jedné z uvedených metrik nevykazoval tento experiment nejlepší hodnotu v porovnání s ostatními. Pro porovnání provedených experimentů však tato hodnotící metoda není vhodná, jak již bylo zdůrazněno v kapitole 10.3.2, kde je také uveden na obrázku 10.8 problém, jehož výskyt může významně ovlivnit výpočet metrik. Pro výběr nejlepšího modelu byly využity pouze výsledky z tabulky na obrázku 10.9.

Model	Optimalizace	Recall	Precision	FPC
Inception 1×1 i 3×3	Adam	0.5588	0.6482	0.1925
Inception 1×1 i 3×3, + 2 Conv2D navíc	Adam	0.6125	0.7226	0.1493
Inception 1×1	Adam	0.5986	0.7359	0.1364
Inception 1×1, + 2 Conv2D navíc	Adam	0.5630	0.7281	0.1335
Bez inception modulu	Adam	0.5894	0.7537	0.1223
Bez inception modulu, + 2 Conv2D navíc	Adam	0.5644	0.7087	0.1473
Inception 1×1 i 3×3	SGD	0.6116	0.7446	0.1332
Inception 1×1 i 3×3, + 2 Conv2D navíc	SGD	0.6245	0.7490	0.1329
Inception 1×1	SGD	0.5787	0.7066	0.1526
Inception 1×1, + 2 Conv2D navíc	SGD	0.6384	0.8088	0.0958
Bez inception modulu	SGD	0.6458	0.7802	0.1155
Bez inception modulu, + 2 Conv2D navíc	SGD	0.6111	0.7441	0.1335
Inception 1×1 i 3×3	RMSprop	0.6204	0.7283	0.1470
Inception 1×1 i 3×3, + 2 Conv2D navíc	RMSprop	0.5981	0.7499	0.1267
Inception 1×1	RMSprop	0.5583	0.7111	0.1440
Inception 1×1, + 2 Conv2D navíc	RMSprop	0.5648	0.7101	0.1464
Bez inception modulu	RMSprop	0.5199	0.7203	0.1282
Bez inception modulu, + 2 Conv2D navíc	RMSprop	0.5875	0.7251	0.1414

Tabulka 10.2: Výsledky ohodnocení experimentů s využitím IoU. Model vyhodnocený jako nejlepší podle kritéria 10.3 se nachází na druhém řádku.

Shrnutí

V této kapitole byly provedeny experimenty za účelem výběru nejlepšího modelu založeného na konvolučních neuronových sítích. Bylo testováno 6 různých architektur a pro každou z nich 3 různé optimalizační metody. Celkem 18 provedených experimentů bylo porovnáno na základě hodnoty výrazu 10.3, jehož maximalizace byla požadována. Jako nejlepší ve smyslu tohoto kritéria se ukázal model s architekturou využívající inception moduly obsahující konvoluční vrstvy s velikostmi jader 1×1 i 3×3 za využití optimalizátoru Adam, přičemž se jednalo o variantu se dvěma přídavnými konvolučními vrstvami vloženými mezi poslední vrstvu dekodéru a výstupní klasifikační vrstvu.

Pro účely porovnání s jinými pracemi byl nejlepší experiment ohodnocen také pomocí metody fungující na principu IoU, kde byly na základě testování vybrány optimální morfologické operace potřebné k vytažení ohraničujících obdélníků detekovaných lézí. Následně byl zkoumán také vliv prahové hodnoty IoU_{min} na vývoj metriky recall a nakonec byly spočteny hodnoty metrik recall, precision a FPC pro všechny experimenty, které ale nesloužily k porovnání experimentů mezi sebou, nýbrž pouze k budoucímu možnému porovnání s jinými pracemi, kde nejsou k dispozici buď anotace, nebo detekce per pixel. Prováděné experimenty byly mezi sebou porovnávány tedy jen podle maximální hodnoty výrazu 10.3.

10.5 Porovnání obou přístupů

Posledním úkolem bylo porovnat výsledky nejlepších natrénovaných modelů pro oba využité přístupy, tedy neuronové sítě a metodu kombinující HoG a SVM. U druhé jmenované metody byla získána optimální konfigurace parametrů, kde předzpracování obrazu bylo prováděno mediánovým filtrem o velikosti jádra 11×11, počet orientací byl roven 12, velikost buňky byla nastavena na 8×8 pixelů a počet buněk na blok byl stanoven na 2. Z neuronových sítí se vyznačoval nejlepšími výsledky model zahrnující architekturu s inception moduly o velikosti jader 1×1 a 3×3, dvěma přídavnými konvolučními vrstvami na konci dekodéru a natrénovaný s využitím optimalizátoru Adam.

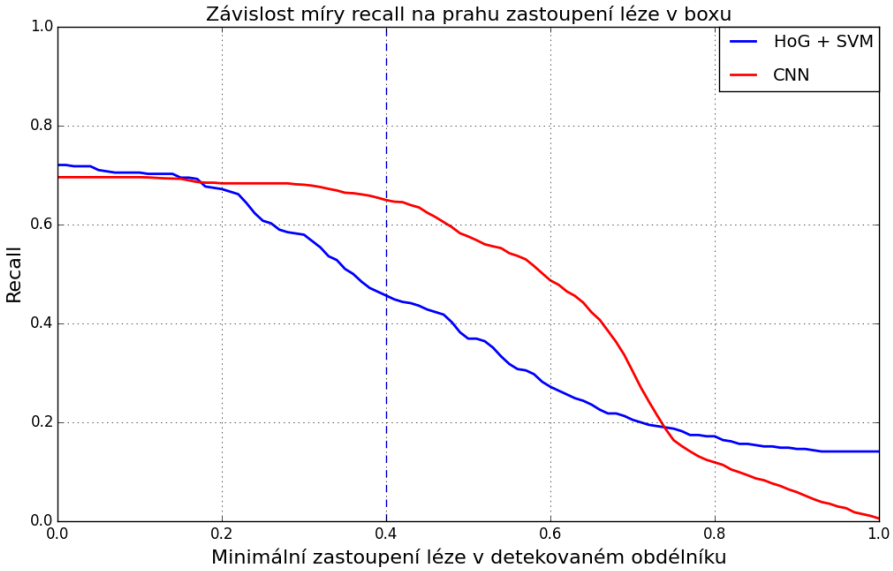
Přestože u neronových sítí se jednalo striktně o per pixel detekci, metodou HoG a SVM byly získávány pouze detekce ve formě ohraničujících obdélníků. Bylo tedy nutné z výstupu neuronové sítě extrahovat ohraničující obdélníky a získat tím stejnou formu detekovaných lézí jako u HoG a SVM. Extrakce ohraničujících obdélníků probíhala způsobem, který je popsán v kapitole 10.3.2, a za využití optimálních parametrů morfologických operací, které byly získány v kapitole 10.4. Jako nejvhodnější se ukázalo porovnat modely na základě hodnoty recall spočtené tak, jak je uváděno v kapitole 9.2.1, tedy stejným způsobem, jakým byly mezi sebou porovnávány jednotlivé konfigurace metody HoG a SVM. Také prahové hodnoty minimálního zastoupení ohraničujícího obdélníku v lézi a léze v ohraničujícím obdélníku zůstaly stejné.

Pro nejlepší model HoG a SVM byly již hodnoty metrik recall a FPC spočteny. Zbývalo tedy jen aplikovat stejný postup na ohraničující obdélníky získané z predikovaných výstupů modelu neuronové sítě. Získané hodnoty pro oba nejlepší modely jsou zapsány v tabulce 10.3.

Model	Recall (TPR)	FPC
Modifikovaný SegNet	0.6468	0.1576
HoG + SVM	0.4564	0.9073

Tabulka 10.3: Porovnání přístupu HoG + SVM s neuronovými sítěmi podle metody popsané v kapitole 9.2.1

Jak je možné vyčíst z tabulky 10.3, neuronová síť vykazovala vyšší hodnotu TPR než metoda kombinující HoG a SVM a rovněž dosáhla několikanásobně nižší hodnoty FPC. Navíc je vhodné připomenout, že metoda, pomocí které byly oba přístupy porovnány, byla navržena přímo pro úlohu HoG + SVM, která i přesto dosáhla horších výsledků. Další výhoda navržené neuronové sítě spočívá v tom, že jejím výstupem jsou per pixel predikce. Lze zhodnotit, že neuronová síť se ukázala jako vhodnější pro tuto úlohu, tedy detekci lézí z řezů CT snímků, jestliže jsou k dispozici trénovací data s per pixel anotacemi a během úlohy samotné klasifikace je známa binární maska jater. Graf vývoje TPR při měnící se hodnotě minimálního nutného pokrytí ohraničujícího obdélníku lézí je pak k vidění na obrázku 10.12.

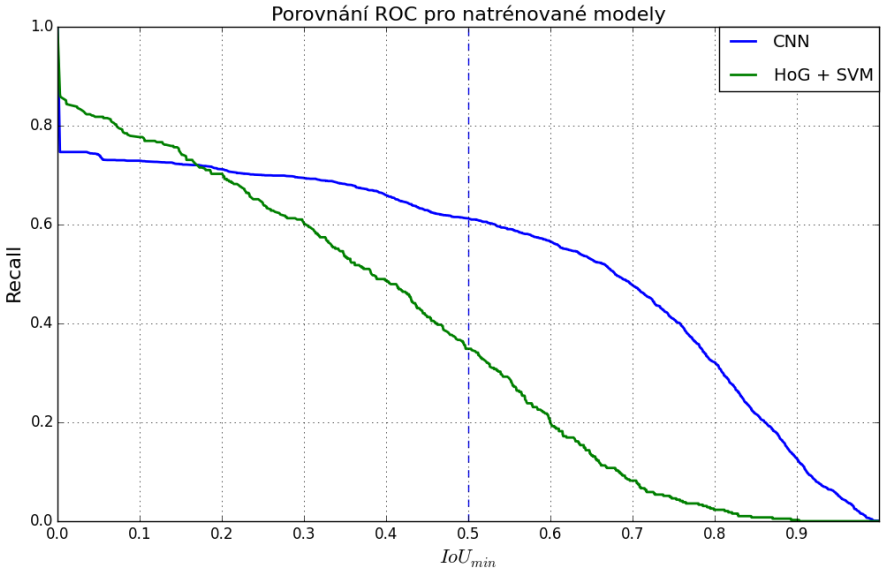


Obrázek 10.12: Graf závislosti recall na hodnotě prahu minimálního pokrytí ohraničujícího obdélníku lézí pro oba přístupy

Pro zajímavost byly oba modely porovnány rovněž podle metody využívající IoU, popsané v kapitole 10.3.2. ROC křivky obou modelů, které znázorňují závislost hodnoty recall na měnícím se prahu IoU_{min}, jsou vykresleny na obrázku 10.13. Lze zde vidět, že CNN vykazovala mnohem vyšší hodnotu recall pro oblast, kde $IoU_{min} > 0.1714$. Výsledky pro IoU_{min} hluboko pod hodnotou 0.5 nejsou pro porovnání příliš užitečné. Například práh $IoU_{min} = 0.1714$ nelze považovat za dostatečné překrytí anotovaných a detekovaných obdélníků. Zajímavá je především hodnota $IoU_{min} = 0.5$, která byla zvolena v kapitole 10.3.2. Hodnoty recall v tomto bodě grafu jsou k vidění v tabulce 10.4, kde je uvedena i získaná hodnota FPC. Pro tuto a dále vyšší hodnoty $IoU_{min} > 0.5$, které znamenají přísnější podmínku minimálního překrytí obdélníků, dosahovala neuronová síť jednoznačně lepších výsledků. Porovnávání výsledků podle metody na principu IoU však není pro tuto úlohu prospěšné, jelikož jsou k dispozici anotace per pixel a u obou přístupů by tak došlo ke ztrátě informace. Ze stejného důvodu nebylo vhodné využívat tuto hodnotící metodu ani k výběru nejlepšího modelu na principu HoG a SVM.

Model	Recall (TPR)	FPC
Modifikovaný SegNet	0.6125	0.1493
HoG + SVM	0.3487	0.9382

Tabulka 10.4: Porovnání přístupu HoG + SVM s neuronovými sítěmi na základě metody využívající IoU při nastavené hodnotě $IoU_{min} = 0.5$



Obrázek 10.13: Porovnání ROC křivek pro nejlepší modely neuronové sítě a metody HoG + SVM

Shrnutí

V této kapitole byly porovnány výsledky dosažené pomocí přístupu HoG a SVM s těmi, kterých bylo dosaženo za použití konvolučních neuronových sítí, a to tak, že byly mezi sebou porovnávány nejlepší modely obou těchto metod. Hodnotící metoda, která byla během porovnání používána, byla stejná jako při výběru nejlepší konfigurace přístupu HoG s klasifikátorem SVM (viz kapitola 9.2.1). Pomocí této metody byly spočteny pro oba modely metriky recall a FPC, přičemž hodnota recall byla rozhodujícím kritériem porovnávání. Lepších výsledků ve smyslu hodnoty metriky recall dosáhl model konvoluční neuronové sítě.

Kapitola 11

Další možná vylepšení

Tato práce otevírá příležitosti pro další možná vylepšení obou navržených přístupů. Těmi hlavními, která by mohla být v navazující práci použita, se zabývá tato kapitola, kde je rovněž uveden příklad praktického využití navržených metod automatické lokalizace lézí v CT snímcích.

11.1 Využití třetího rozměru CT snímku

Jelikož se jedná o čistě dvourozměrnou úlohu, tedy pouze o lokalizaci jaterních lézí ve 2D řezech, může se stát, že pro dva za sebou jdoucí řezy, které mají velmi podobnou strukturu, jsou získány značně odlišné výsledky. Například může jít o případ, kdy je v jednom řezu detekována léze, ale v žádném z několika okolních řezů nikoliv. Pak by se nabízelo tento izolovaný detekovaný ohraničující obdélník z daného řezu odstranit na základě lokální informace využívající třetí rozměr. Pravděpodobnost, že se léze nachází v jediném snímku a v okolních nikoliv, je velice nízká, obzvláště pokud jde o lézi velkých rozměrů. Předpokladu podobnosti dvou sousedících řezů by mohlo být také využito například v malé úpravě umístění detekovaného obdélníku v každém řezu s využitím informace o umístění odpovídajících obdélníků v řezech okolních. Propojením detekcí v po sobě jdoucích řezech by mohlo dojít k vytvoření 3D modelu léze. Pokud by byl takovýto trojrozměrný objekt příliš nekompaktní nebo malý, mohlo by se rovněž uvažovat o jeho odstranění.

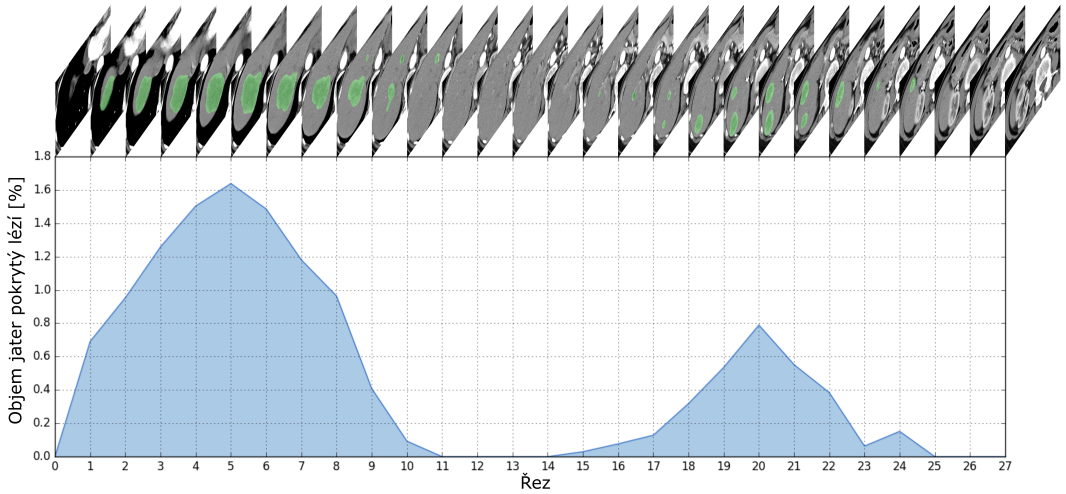
11.2 Využití velikostí voxelů

V této práci bylo předpokládáno, že všechny voxely, tedy i pixely, pokud je řeč o samotných řezech, mají velmi podobné rozměry pro všechny CT snímky. Kdyby ale na vstup úlohy přišla data s výrazně odlišnými rozměry voxelů, mohl by se u obou použitých přístupů vyskytnout problém. U metody detekce pomocí HoG a SVM by pevné nastavení velikosti klouzavého okénka mohlo způsobovat například takové problémy, že by mohlo zabírat v takto odlišných datech mnohem větší plochu než u dat trénovacích, čímž by pravděpodobně nedošlo k detekci malých lézí. Zde by se jako řešení nabízelo upravení prostorových parametrů, jako jsou například velikost kroku a šířka klouzavého okna nebo velikost fil-

tru, tak, aby byly úměrné velikosti pixelů. U konvolučních neuronových sítí by pak nastal problém při výpočtu matice přesnosti, která je ovlivněna pouze počtem správně a špatně detekovaných pixelů. Počty pixelů by mohly být poté nahrazeny plochami, které tyto pixely zaujímají. Další problémy by u konvolučních neuronových sítí způsobovala pevná velikost konvolučních jader, která byla v této práci nastavena na 3×3. Pokud by pixely vstupního obrázku nabývaly například dvojnásobné velikosti, do druhé vrstvy by pak bylo propagováno dvojnásobně velké zorné pole, než jaké bylo propagováno u trénovacího snímku. Řešením by mohlo být škálování natrénovaných filtrů první vrstvy tak, aby zorné pole jejího výstupu mělo stejnou velikost jako v případě trénovacích dat. Dále, pokud by velikost pixelů vstupního obrázku byla několikanásobná a řez by obsahoval pixelů několikanásobně méně než u trénovacích dat, by bylo vhodné dosáhnout požadované velikosti vstupu sítě změnou měřítka tohoto obrázku. V této práci však byly rozdíly ve velikostech voxelů jednotlivých CT snímků zanedbatelné.

11.3 Křivka zastoupení lézí v jednotlivých řezech

Využití by mohla mít automatická detekce lézí mimo jiné i v tom, že by okamžitě po pořízení CT snímku a následné automatické lokalizaci jater a lézí v jeho jednotlivých řezech mohla být sestrojena křivka udávající, kolik procent celého řezu, případně jen jater, zaujímají pixely označené klasifikátorem jako léze. V případě detekce ohraničujících obdélníků by se mohlo jednat o procentuální plochu, kterou tyto detekované obdélníky zabírají. Tato křivka by pak mohla posloužit lékařům jako orientační náhled ještě před detailním prozkoumáním jednotlivých řezů. Mohli by tím například získat informaci, na jaké řezy se zaměřit pozorněji. Tyto řezy by se pravděpodobně nacházely v blízkosti lokálních maxim křivky. Na obrázku 11.1 je k vidění ukázka, jak by mohla tato křivka vypadat. Hodnoty na ose x značí, kolik procent celkového objemu jater v CT snímku zabírá léze v daném řezu.



Obrázek 11.1: Křivka zastoupení lézí v jednotlivých řezech

Kapitola 12

Závěr

Cílem této práce bylo navrhnout metodu pro automatickou lokalizaci jaterních lézí v CT snímcích. Celkem byly navrženy dva přístupy. Prvním z nich byla metoda pro detekci obdélníků ohraničujících jaterní léze v jednotlivých 2D řezech. Zde bylo využíváno předzpracování obrazu pomocí mediánového či bilaterálního filtru, histogramu orientovaných gradientů (HoG) jako extraktoru vektoru příznaků a lineárního klasifikátoru s podpůrnými vektory (SVM). Druhý přístup byl založen na sémantické segmentaci pomocí konvolučních neuronových sítí, kde byla využívána architektura inspirovaná sítí SegNet [31]. Jako aktivační funkce byla pro konvoluční vrstvy využívána ReLU a za ztrátovou funkci byla zvolena křížová entropie. Na výstup konvolučních vrstev byla navíc aplikována dávková normalizace. Dalším důležitým nastavením během trénování sítě byla optimalizační metoda, jejíž tři typy byly během experimentů testovány, přičemž se jednalo o SGD, RMSprop a Adam.

U přístupu využívajícího metody HoG a SVM byly testovány typy a velikosti filtrů pro předzpracování obrazu a různé parametry HoG extraktoru, jako jsou počet orientací, velikost buňky a počet buněk na blok. Nejprve byla prováděna křížová validace, kde jako trénovací i validační data sloužily již extrahované výřezy, které buď obsahovaly lézi, nebo nikoliv. Na základě křížové validace byl počet testovaných parametrů redukován. Finální testování již probíhalo pomocí klouzavého okénka a pyramidy obrazu, kde pro testovací 2D řezy došlo k vygenerování pozitivních ohraničujících obdélníků obsahujících léze. Využívanými hodnotícími metrikami byly především recall a FPC, které závisely na počtech TP, FP, a FN, pro jejichž vyhodnocení byla navržena metoda pracující mimo jiné s minimálním nutným pokrytím detekovaného obdélníku lézí. Nejlepších výsledků dosahoval model využívající mediánový filtr o velikosti jádra 11 pixelů a parametry HoG extraktoru, kde počet orientací byl nastaven na 12, velikost buňky na 6 pixelů a počet buněk v bloku na 2. Pro nejlepší model byla následně provedena FROC analýza, která vedla k určení optimálního prahu minimální pravděpodobnosti, které musela hodnota přiřazená klasifikátorem danému obdélníku dosahovat, aby byl označen jako pozitivní. Tento práh byl nastaven na hodnotu $P_{min} = 0.85$. Nakonec byla pro nejlepší model ještě provedena metoda Hard Negative Mining a došlo k opětovnému natrénování klasifikátoru, což vedlo k výraznému snížení míry FPC.

Experimentů s konvolučními neuronovými sítěmi bylo prováděno celkem 18, přičemž bylo testováno 6 různých architektur a pro každou z nich 3 různé optimalizační metody, tedy SGD, RMSprop a Adam. Všechny testované architektury měly společné to, že vycházely z architektury SegNet [31]. Lišily se pouze v tom, jakým způsobem byla na výstup sítě propagována různá zorná pole. Zde byly testovány tři způsoby. Nejjednodušší z nich využíval pouze skip connections. Další architektura zaměnila skip connections za inception moduly s konvolučními vrstvami o velikosti jader 1×1. Nejsložitější architektura obsahovala stejné inception vrstvy a navíc s nimi spojovala inception vrstvy o rozměru konvolučních jader 3×3. Pro každou takovouto architekturu byla navíc testována ještě varianta se dvěma přídavnými konvolučními vrstvami vloženými bezprostředně před klasifikační vrstvu. Dále byla navržena hodnotící metrika pro výběr nejlepšího natrénovaného modelu, která byla ovlivněna trojicí podmíněných pravděpodobností, s jakou budou pixely správně zařazeny do jedné ze tří tříd, pokud do této třídy patří i ve skutečnosti. Nejlepších výsledků ve smyslu této metriky dosáhl model s architekturou sítě s inception vrstvami o velikostech konvolučních jader 1×1 i 3×3 a dvěma přídavnými konvolučními vrstvami, natrénovaný s optimalizátorem Adam. Pouze pro účely porovnání výsledků s výsledky dosaženými v jiných pracích byla vyhodnocena také metrika využívající IoU, která byla použita u PASCAL výzvy [43].

V další kapitole byly mezi sebou porovnány nejlepší natrénované modely pro oba přístupy, a to podle metody, která byla navržena pro výběr nejlepší konfigurace přístupu využívajícího HoG a SVM. Také zde byla zkoumána především míra recall, která dosahovala mnohem vyšší hodnoty pro výsledky získané pomocí konvoluční neuronové sítě. Tento přístup byl tedy vyhodnocen pro úlohu automatické lokalizace jaterních lézí v CT snímcích jako vhodnější.

V poslední kapitole byly uvedeny dvě možnosti dalšího rozšíření práce. Jednou z nich je využití informace o detekovaných lézích v okolních řezech snímku k úpravě výsledků aktuálního řezu. Druhým možným rozšířením by pak mohlo být využití informace o velikosti voxelů v jednotlivých CT snímcích. Nakonec bylo navrženo možné praktické využití automatické lokalizace jaterních lézích, které spočívá ve vykreslení křivky znázorňující procentuální zastoupení jaterních lézí v jednotlivých řezech CT snímku.

Výsledky, kterých bylo v této práci dosaženo, dokazují, že obě navržené metody mohou být pro automatickou detekci jaterních lézí použity. Účelem automatizace procesu detekce lézí je totiž usnadnění práce lékařům, čehož by v případě nasazení obou přístupů v praxi dosaženo bylo. Výsledky této práce lze tedy zhodnotit jako velmi přínosné. Nasazování podobných nových softwarových řešení v klinické praxi však není jednoduchou záležitostí. Takovéto systémy by totiž měly disponovat určitou certifikací, což je zdlouhavý a náročný proces.

Literatura

[1] LING, Haibin, et al. Hierarchical, learning-based automatic liver segmentation. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008. p. 1-8.

[2] NOH, Hyeonwoo; HONG, Seunghoon; HAN, Bohyung. Learning deconvolution network for semantic segmentation. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015. p. 1520-1528.

[3] FERDA, Jiří, et al. *Základy zobrazovacích metod*. Galén, 2015.

[4] LIM, Seong-Jae; JEONG, Yong-Yeon; HO, Yo-Sung. Automatic liver segmentation for volume measurement in CT Images. *Journal of Visual Communication and Image Representation*, 2006, 17.4: 860-875.

[5] SONKA, Milan; HLAVAC, Vaclav; BOYLE, Roger. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[6] TOMASI, Carlo; MANDUCHI, Roberto. Bilateral filtering for gray and color images. In: *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998. p. 839-846.

[7] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[8] BISHOP, Christopher M. *Pattern recognition and machine learning*. New York: Springer, c2006. Information science and statistics. ISBN 9780387310732.

[9] GOODFELLOW, Ian, et al. *Deep learning*. Cambridge: MIT press, 2016.

[10] PEDREGOSA, F., et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825-2830, 2011.

[11] DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005. p. 886-893.

[12] DÉNIZ, Oscar, et al. Face recognition using histograms of oriented gradients. *Pattern Recognition Letters*, 2011, 32.12: 1598-1603.

[13] MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 1943, 5.4: 115-133.

[14] BISHOP, Christopher M. *Neural networks for pattern recognition*. Oxford university press, 1995.

[15] ROSENBLATT, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. *Washington DC, Spartan*, 1961.

[16] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. 2012. p. 1097-1105.

[17] NAIR, Vinod; HINTON, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010. p. 807-814.

[18] GLOROT, Xavier; BORDES, Antoine; BENGIO, Yoshua. Deep sparse rectifier neural networks. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011. p. 315-323.

[19] RUMELHART, David E., et al. *Parallel distributed processing*. Cambridge, MA: MIT press, 1987.

[20] BOTTOU, Léon. Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*. Physica-Verlag HD, 2010. p. 177-186.

[21] NESTEROV, Yurii. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In: *Soviet Mathematics Doklady*. 1983. p. 372-376.

[22] KINGMA, Diederik P.; BA, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] FITZPATRICK, J. Michael; HILL, Derek L. G.; MAURER Jr., Calvin R. *Handbook of medical imaging: Image registration*, 2000, 2: 447-513.

[24] IOFFE, Sergey; SZEGEDY, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning*. 2015. p. 448-456.

[25] SIMARD, Patrice Y., et al. Best practices for convolutional neural networks applied to visual document analysis. In: *ICDAR*. 2003. p. 958-962.

[26] KARPATHY, Andrej. Convolutional Neural Networks: Architectures, Convolution / Pooling Layers. *CS231n Convolutional Neural Networks for Visual Recognition* [online]. 2017 [cit. 2018-03-27]. Dostupné z: `http://cs231n.github.io/convolutional-networks/`

[27] LIU, Wei, et al. Ssd: Single shot multibox detector. In: *European conference on computer vision*. Springer, Cham, 2016. p. 21-37.

[28] LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. p. 3431-3440.

[29] SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] SCHERER, Dominik; MÜLLER, Andreas; BEHNKE, Sven. Evaluation of pooling operations in convolutional architectures for object recognition. In: *International conference on artificial neural networks*. Springer, Berlin, Heidelberg, 2010. p. 92-101.

[31] BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2017, 39.12: 2481-2495.

[32] CHOLLET, F. Keras [online]. 2015 [cit. 2018-03-27]. Dostupné z: `http://github.com/fchollet/keras`

[33] KOHAVI, Ron, et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Ijcai*. 1995. p. 1137-1145.

[34] VISA, Sofia, et al. Confusion Matrix-based Feature Selection. In: *MAICS*. 2011. p. 120-127.

[35] POWERS, David Martin. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.

[36] VIOLA, Paul; JONES, Michael. Rapid object detection using a boosted cascade of simple features. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. IEEE, 2001. p. I-I.

[37] ADELSON, Edward H., et al. Pyramid methods in image processing. *RCA engineer*, 1984, 29.6: 33-41.

[38] Image Pyramids. *OpenCV: Open Source Computer Vision* [online]. 23. 12. 2016 [cit. 2018-03-27]. Dostupné z: `https://docs.opencv.org/3.2.0/dc/dff/tutorial_py_pyramids.html`

[39] HOSANG, Jan; BENENSON, Rodrigo; SCHIELE, Bernt. Learning non-maximum suppression. *arXiv preprint*, 2017.

[40] FELZENSZWALB, Pedro F., et al. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 2010, 32.9: 1627-1645.

[41] SHRIVASTAVA, Abhinav; GUPTA, Abhinav; GIRSHICK, Ross. Training region-based object detectors with online hard example mining. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016. p. 761-769.

[42] RUSKÓ, László; PERÉNYI, Ádám. Automated liver lesion detection in CT images based on multi-level geometric features. *International journal of computer assisted radiology and surgery*, 2014, 9.4: 577-593.

[43] EVERINGHAM, Mark, et al. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 2015, 111.1: 98-136.

[44] BANKMAN, Isaac (ed.). *Handbook of medical image processing and analysis*. Elsevier, 2008.

[45] HE, Kaiming, et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. p. 770-778.

[46] JACKSON, Aaron S., et al. Large pose 3D face reconstruction from a single image via direct volumetric CNN regression. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017. p. 1031-1039.

[47] RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015. p. 234-241.

[48] WANG, Fei, et al. Residual attention network for image classification. *arXiv preprint arXiv:1704.06904*, 2017.

[49] SZEGEDY, Christian, et al. Going deeper with convolutions. In: *CVPR*, 2015.

Seznam tabulek

Seznam obrázků