Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-92025

Matej Horniak

Spracovanie obrazových dát metódami umelej inteligencie

Bakalárska práca

Vedúci práce: Ing. Marek Jakab

Máj 2020

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-92025

Matej Horniak

Spracovanie obrazových dát metódami umelej inteligencie

Bakalárska práca

Študijný program: Informatika

Študijný odbor: 9.2.1 Informatika

Miesto vypracovania: FIIT STU, Bratislava

Vedúci práce: Ing. Marek Jakab

Máj 2020

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

ZADANIE BAKALÁRSKEHO PROJEKTU

Meno študenta:	**Horniak Matej**
Študijný odbor:	Informatika
Študijný program:	Informatika
Názov projektu:	**Spracovanie obrazových dát metódami umelej inteligencie**

Zadanie:
Spracovanie obrazových dát metódami umelej inteligencie, predovšetkým s využitím hlbokých neurónových sietí, je stále významnejšia súčasť metód počítačového videnia. Doteraz boli navrhnuté, overované a publikované rôzne architektúry hlbokých neurónových sietí vhodné pre rôzne aplikácie. Analyzujte metódy hlbokého učenia vhodné pre spracovanie obrazových dát, ktoré majú charakter rôznych textúr. Príkladom takýchto dát môžu byť medicínske histologické mikroskopické snímky alebo iné obrazové dáta bohaté na rôzne textúrovaný obsah. Zamerajte sa predovšetkým na metódy využívajúce konvolučné hlboké neurónové siete.
Navrhnite a overte viaceré metódy na klasifikáciu medicínskych histologických snímkov a iných textúrových obrazových dát s využitím analyzovaných moderných prístupov. Navrhnuté metódy realizujte s využitím knižníc a rámcov vhodných na spracovanie obrazových dát. Porovnajte navzájom realizované prístupy, ich presnosť, robustnosť a časovú efektívnosť spracovania. Výsledky porovnajte tiež s inými publikovanými riešeniami.

Práca musí obsahovať:

Anotáciu v slovenskom a anglickom jazyku
Analýzu problému
Opis riešenia
Zhodnotenie
Technickú dokumentáciu
Zoznam použitej literatúry
Elektronické médium obsahujúce vytvorený produkt spolu s dokumentáciou

Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU, Bratislava
Vedúci projektu: Ing. Marek Jakab

Termín odovzdania práce v letnom semestri : 14.5.2020

Bratislava 17.2.2020

doc. Ing. Peter Lacko, PhD.
riaditeľ ÚISI

Čestne vyhlasujem, že som túto prácu vypracoval(a) samostatne, na základe konzultácií s Ing. Marek Jakab, a s použitím uvedenej literatúry.

V Bratislave, ...

10.12.2019 Matej Horniak

Poďakovanie

Moje poďakovanie patrí najmä, Ing. Marek Jakab a doc. Ing. Vanda Benešová, PhD., za všetky odborné rady, usmernenia, ľudský prístup a ochotu pomôcť, ktoré mi poskytli počas vypracovávania.

Osobitné poďakovanie patrí mojej celej rodine, hlavne matke za jej podporu, pomoc a morálnu oporu, ktorú mi poskytla počas vypracovávania bakalárskej práce.

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Informatika

Autor: Matej Horniak
Bakalárska práca: Spracovanie obrazových dát metódami umelej inteligencie
Vedúci bakalárskeho projektu: Ing. Marek Jakab
Máj 2020

V dnešnej dobe je čo raz viac v popredí spracovávanie obrazov pomocou počítacom. Táto úloha je veľmi náročná, či už na hardvérové požiadavky, ale najmä na softwarové. Čo sa týka softvéru, tejto úlohe sa venujú viaceré odvetvia počítačovej vedy, a to hlavne počítačové videnie. Do popredia v tejto úlohe sa dostáva najmä odvetvie umelej inteligencie, ktoré preukazuje nesmierne pozitívne výsledky, čo sa týka zmenšovania potrebnej pamäťovej a časovej náročnosti. Táto skutočnosť prináša rôzne výhody ako efektívnejšie a rýchlejšie spracovávať obrazových dát alebo zväčšenie presnosti či už klasifikácie alebo segmentácie.

V tejto práci sa najskôr zameriavame na spracovanie obrazov, klasifikáciu nad obrazovými dátami. Ďalej sa zameriame na metódy hlbokého učenia a na porovnanie rôznych metód generovania filtrov pre konvolučné neurónové siete. Predstavíme rôzne metódy automatického a manuálneho generovania filtrov.

Cieľom tejto práce bude implementovať konvolučné neurónové siete, ktoré budú klasifikovať histologické dáta a dáta rôznych texturóvych povrchov. Pričom budeme využívať rôzne metódy generovania filtrov na prvej vrstve. Ku koncu porovnáme úspešnosti jednotlivých neurónových sietí.

Annotation

Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies
Degree Course: Informatika

Author: Matej Horniak
Bachelor Thesis: Image data processing using the methods of artificial intelligence.
Supervisor: Ing. Marek Jakab
Máj 2020

Nowadays, computer image processing is at the forefront as much as possible. This task is very demanding, whether for hardware requirements, but especially for software requirements. As far as software is concerned, several branches of computer science, especially computer vision, are dedicated to this task. In particular, the artificial intelligence industry is at the forefront of this role, showing extremely positive results in terms of reducing the memory and time required. This provides us with a lot of benefits, such as efficient and faster image data processing or increased accuracy of either classification or segmentation.

In this work, firstly we focus on image processing, classification over image data. Next, we will focus on deep learning methods and on the comparison of different methods of generating filters for convolutional neural networks. We will introduce various methods of automatic and manual generation of filters.

Our goal is to implement convolutional neural networks, which will classify the histological data and data of various textural surfaces. We will use different methods of generating filters on the first layer. Finally, we compare the success of individual neural networks.

Obsah

1 Úvod **1**

2 Analýza **3**

2.1 Umelá inteligencia 3

2.2 Počítačové videnie 5

2.2.1 Proces klasifikácie 6

2.2.2 Algoritmy klasifikácie 6

2.2.3 Operácia konvolúcie 7

2.2.4 Gáborové filtre 9

2.3 Neurónové siete a hlboké učenie 11

2.3.1 Neurón ako základná jednotka 11

2.3.2 Neurónové siete 12

2.3.2.1 Perceptrón 13

2.3.3 Konvolučné neurónové siete 13

2.3.3.1 Členenie dát konvolučných neurónových sietí 14

2.3.3.2 Pooling operácia (združovacia) 16

2.3.4 Autoenkóder 17

2.3.5 Transfer learning 19

2.3.5.1 Vytvorenie modelu od začiatku 20

2.3.5.2 Využitie natrénovaného modulu 21

2.3.5.3 Príklad transfer learning 21

2.4 Publikované práce 22
2.4.1 Publikácia s názvom A novel deep learning based framework for the detection and classification of breast cancer using transfer learning 22
2.4.2 Publikácia s názvom Classification of breast cancer histology images using Convolutional Neural Networks 24
2.4.3 Publikácia Transfer learning based deep CNN for segmentation and detection of mitoses in breast cancer histopathological images 28
2.4.4 Publikácia Rotation Equivariant CNNs for Digital Pathology 31

3 Návrh **35**
3.1 Koncept riešenia 35
3.1.1 Architektúra 36
3.1.1.1 Implementácia architektúry 36
3.1.2 Metóda generovania filtrov s využitím backpropagation . . . 38
3.1.2.1 Implementácia backpropagation 38
3.1.2.2 Vizualizácia filtrov 39
3.1.3 Metóda generovania filtrov s využitím autoenkóder 39
3.1.3.1 Implementácia autoencoderu 40
3.1.3.2 Vizualizácia filtrov 41
3.1.4 Metóda generovania filtrov s využitím Transfer learning . . . 41
3.1.4.1 Implementácia transfer learningu 42
3.1.4.2 Vizualizácia filtrov 43
3.1.5 Metóda generovania filtrov s využitím Gáborových filtrov . . 43
3.1.5.1 Implementácia Gáborových filtrov 44
3.1.5.2 Vizualizácia filtrov 45

4 Implementácia **46**

4.1 Použitý hardvér a softvér 46

4.2 Implementácia navrhnutého riešenia 47

4.2.1 Opis implementácie vytvárania modelov 47

4.2.2 Opis implementácie hlavnej časti 48

5 Popis experimentov a evaluácia **50**

5.1 Použité dáta 50

5.1.1 Dataset "PatchCamelyon" (PCAM) 50

5.1.1.1 Preprocessing 50

5.1.2 Dataset "Deutsche Arbeitsgemeinschaft für Mustererkennung" (DAGM) 51

5.1.2.1 Preprocessing 52

5.1.2.2 Kĺzavé okno 53

5.2 Vyhodnocovanie jednotlivých metód generovania filtrov 54

5.2.1 Použité metriky 54

5.2.2 Výsledky na datasete PCAM 55

5.2.2.1 Transfer learning 56

5.2.2.2 Backpropagation 57

5.2.2.3 Autoenkodér 57

5.2.2.4 Gáborové filtre 58

5.2.3 Výsledky na datasete DAGM 58

5.2.3.1 Transfer learning 59

5.2.3.2 Backpropagation 59

5.2.3.3 Autoenkodér 59

5.2.3.4 Gáborové filtre 60

6 Záver **61**

Literatúra **64**

Príloha A: Inštalačná príručka

Príloha B: Plán na zimný semester 2019/2020

Príloha C: Plán na letný semester 2019/2020

Príloha D: Opis digitálnej časti práce

Kapitola 1

Úvod

V dnešnej dobe je čo raz viac v popredí spracovávanie obrazov pomocou počítacom. Táto úloha je veľmi náročná, či už na hardvérové požiadavky, ale najmä na softwarové. Čo sa týka softvéru, tejto úlohe sa venujú viaceré odvetvia počítačovej vedy, a to hlavne počítačové videnie. Do popredia v tejto úlohe sa dostáva najmä odvetvie umelej inteligencie, ktoré preukazuje nesmierne pozitívne výsledky, čo sa týka zmenšovania potrebnej pamäťovej a časovej náročnosti. Táto skutočnosť prináša rôzne výhody ako efektívnejšie a rýchlejšie spracovávať obrazových dát alebo zväčšenie presnosti či už klasifikácie alebo segmentácie.

Odvetvie umelá inteligencia sa venuje vytváraniu systémom a algoritmom, ktoré sú bežne vykonávané ľuďmi. Táto časť počítačovej vedy sa v dnešnej dobe používa na mnohých miestach a v mnohých aplikáciách, či už je to v automobiloch, smart telefónoch, televíziách, osobných počítačoch, vo výskumoch na spracovanie fyzikálnych alebo biologických dát. Vďaka všestrannosti umelej inteligencie a efektívnosti, čo sa týka počtu úloh, ktoré dokáže spracovať a vykonať niekoľkonásobne rýchlejšie ako človek, sa začala používať v biológii ako takej a hlavne v medicíne. V medicíne ide o spracovanie vizuálnych výstupov z diagnostických vyšetrení, aby sa

predchádzalo ľudským chybám, ako napríklad neskontrolovaniu určitých záznamov alebo nevšimnutiu si závažných úkazov, či už na röntgenových snímkach, snímkach z magnetickej rezonancie, CT alebo v histologických a mikroskopických dátach. Aj napriek vyspelosti umelej inteligencie zostáva posledné finálne rozhodnutie o určení diagnózy stále na človeku, doktorovi, odborníkovi. Vo väčšine prípadov slúži umelá inteligencia, konkrétnejšie vizualizácia, len na ohodnotenie výsledkov, či sú záväzné a či je potreba ich kontroly.

Vďaka nástupu hlbokého učenia neurónových sietí je možné zlepšiť presnosť výsledkov spracovávaných dát a množstvo dát, ktoré dokážeme spracovávať. Jedinou nevýhodou tejto metódy (spracovania obrazových dát pomocou neurónových sietí) je, že pre svoju existenciu a presnejšie určenie výsledkov potrebuje obrovské množstvo dát na trénovanie.

V práci sa budeme zameriavať hlavne na použitie rôznych metód na generovanie filtrov konvolučných neurónových sietí. Porovnáme si metódy generovania filtrov manuálne a automaticky, filtre, ktoré si neurónová sieť vytvára sama. Tieto filtre sa budú zameriavať na detekciu rakoviny v histologických dátach a na klasifikáciu v dátach rôznych štruktúrovaných povrchov. V analýze zistíme možné spôsoby tvorby filtrov a v návrhu niektoré vyskúšame a naimplementujeme vlastnú architektúru, pričom budeme vedieť vložiť rôzne metódy na generovanie filtrov. V evaluácií porovnáme efektívnosť a presnosť jednotlivých metód, opíšeme dáta, na ktorých budeme jednotlivé metódy a architektúru testovať.

Kapitola 2

Analýza

2.1 Umelá inteligencia

Definícia umelej inteligencie je veľmi komplikovaná či už je to kvôli veku tohto odvetvia, alebo z iných dôvodov. Stuart Russell a Peter Norvig (2009) vytvorili dve hlavné delenia umelej inteligencie a to na umelú inteligenciu, ktorá skúma, resp. sa usiluje o myšlienkové procesy a usudzovanie na jednej strane alebo o správanie sa na druhej strane. Druhé delenie spočíva v tom, že sa hodnotí úspech podľa podobnosti s ľuďmi alebo s racionálnosťou, čo znamená, že systém je racionálny, pokiaľ vykonáva správnu vec. Vďaka tomuto ju vieme rozdeliť na 4 hlavné pohľady.

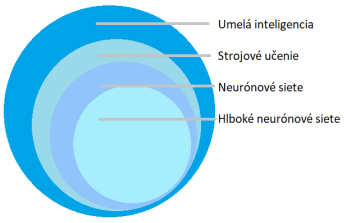
- Systémy, ktoré myslia ako ľudia. Kde cieľom nie je iba vyriešiť úlohu, ale aj vyriešiť ju tak, ako by ju riešil človek.
- Systémy, ktoré konajú ako ľudia. Hlavnými kritériami sú, aby systém dokázal spracovávať prirodzený jazyk, aby mal nejakú základnú reprezentáciu poznatkov, aby vedel používať zapísané informácie na zodpovedanie otázok

a aby sa dokázal učiť a rozpoznať vzorce v správaní.

- Systémy, ktoré myslia racionálne. Čo znamená, že systém dokáže usudzovať, čo je správne. Príkladom môže byť časť matematickej logiky – takzvané výroky a ich jednotlivé pravidlá.
- Systémy, ktoré konajú racionálne. Kde systém koná tak, aby dosiahol cieľ s ohľadom na tvrdenia, ktorým verí. Hlavnou myšlienkou je chápanie inteligencie ako racionálneho konania.[11]

Metódy umelej inteligencie dokážeme rozdeliť podľa širokého spektra faktorov, ako napríklad podľa konkrétnosti úloh na slabú a silnú umelú inteligenciu, pričom slabá dokáže riešiť jednu špecifickú úlohu a silná sa snaží riešiť úlohy na všeobecnejšej rovine, teda simulovať rozmýšľanie.

Jedným z najhlavnejším faktorom je rozdelenie, vďaka ktorému vyberáme umelú inteligenciu pre špecifickú úlohua to na strojové učenie, spracovanie prirodzeného jazyka, expertný systém, videnie, reč, plánovanie a roboty. Strojové učenie je veľmi rozsiahla časť, ktorá je prepojená s inými časťami umelej inteligencie, ako sú roboty, videnie alebo spracovanie prirodzeného jazyka. Dokážeme ju aj ďalej deliť na menšie časti, ako sú hlboké učenie, učenie s učiteľom, učenie bez učiteľa alebo učenie posilňovaním. Jeden z hlavných princípov, ktorý sa používa v strojovom učení, je neurónová sieť. Rozdelenie umelej inteligencií a spojitosť s neurónovou sieťou môžeme vidieť na obrázku č. 2.1. [5, 7]



Obr. 2.1: Množinové znázornenie rozdelenia umelej inteligencie

2.2 Počítačové videnie

Počítačové videnie je odvetvie umelej inteligencie a počítačovej vedy, ktorej cieľom je poskytnúť porozumenie o obrazoch alebo videí počítačom. Hlavnými úlohami, ktorými sa počítačové videnie zaoberá, sú spracovanie, analyzovanie a chápanie digitálnych obrazov. Oblastí, v ktorých vieme využiť počítačové videnie, je mnoho, napríklad:

- ovládanie procesov (autonómne vozidlá, priemyselné roboty),
- detekcia a rozpoznávanie objektov (napr. v priemyselných aplikáciách),
- detekcia javov (napr. sledovanie zmien v bezpečnostných kamerách),
- organizácia informácií (indexovanie databázy obrázkov),
- interakcia (gestá, eye tracking...),
- spracovanie medicínskych vizuálnych dát...

Jedným z hlavných využití počítačového videnia je spracovanie obrazov v medicíne. Táto oblasť je charakterizovaná získavaním informácií za účelom stanovenie diagnózy pacienta. Spracovávajú sa obrazy získané z mikroskopov, RTG, angiografie, tomografie alebo magnetickej rezonancie. Pomocou nich sa detekuje výskyt nádoru, aterosklerózy alebo iných chorobou napadnutých tkanív. Vďaka nimi vieme získavať aj iné informácie, ako sú veľkosť orgánov a tok krvi.

Počítačové videnie sa používa aj v priemysle. Možnosti využitia sú viaceré, či už ovládanie procesov, alebo strojové videnie, kde účelom je zrýchlenie výrobného procesu, teda kontrola kvality alebo určovanie pozície drobných predmetov.[18]

2.2.1 Proces klasifikácie

Proces klasifikácie je postupnosť operácií, do ktorej vstupuje určitá množina dát, ktorú chceme zaradiť do kategórií. V každej kategórii majú dáta rovnakú alebo podobnú vlastnosť. Operácia, ktorá sa vykonáva na zaradenie, sa nazýva reťazec operácií. Tento reťazec sa vo väčšine prípadov skladá z nasledujúcich operácií:

- Predspracovanie obsahu, kde úlohou je odfiltrovanie nežiadúcich objektov a potlačenie niektorých chýb dát.
- Výpočet príznakov, pričom príznakom sa rozumie výsledok merania, ktorý dostaneme určením rozsahu nejakej vlastnosti. Napríklad tvaru, textury, alebo intenzity. Jednou z hlavných výhod tejto operácie je, že zjednodušíme následnú úlohu klasifikácie.
- Klasifikácia - určenie triedy objektu.[18]

2.2.2 Algoritmy klasifikácie

Najznámejšie algoritmy, ktoré sa používajú na zaradenie dát do kategórií (tiež nazývané tradičné techniky klasifikácie) sú:

Support-vector machine (podporný vektorový stroj) vytvára hyperplane alebo skupinu hyperplane v multidimenzionálnom priestore, ktoré sa následne používajú na klasifikáciu. Kvalitná separácia sa dosahuje tak, že hyperplane bude mať veľkú vzdialenosť od najbližšieho bodu trénovacich dát. Pretože všeobecne platí, že čím je väčšie rozpätie, tým je menšia chyba generalizácie.

Random binary forest(náhodné binárne lesy) je technika klasifikácie, ktorá sa na trénovacích dátach najprv naučí špeciálne charakteristiky, vlastnosti, z ktorých následne vytvorí stromy (dátovú štruktúru), pričom vetvy obsahujú určitú charakteristiku dát. Pri rozdeľovaní sa prechádzajú tieto stromy (lesy) či dáta

obsahujú túto charakteristiku a pokiaľ nie, prechádza sa do inej vetvy. Takto sa prechádza celý strom/stromy, pokiaľ sa dáta nerozdelia do kategórií..

K-nearest neighbors algorithm (k-najbližších susedov) je algoritmus používaný pre klasifikáciu a regresiu. Trénovacia časť spočíva v ukladaní vektorov, vo viacdimenzionálnom priestore a označení tried trénovacích dát. Pri spojitých dátach sa využívajú hlavne euklidovské vzdialenosti týchto dát a pri diskrétnych sa používajú iné metriky, napríklad prekrývacia metrika (overlap) hammingova vzdialenosť.

2.2.3 Operácia konvolúcie

Operácia konvolúcie je hlavný nástroj pre lineárnu filtráciu a elementárnou časťou pre konvolučné neurónové siete. Vzorec pre diskrétnu konvolúciu je nasledovný

$$g(x) = \sum_{m=0}^{N-1} f(m)h(x-m)citeSikudova2013 \tag{2.1}$$

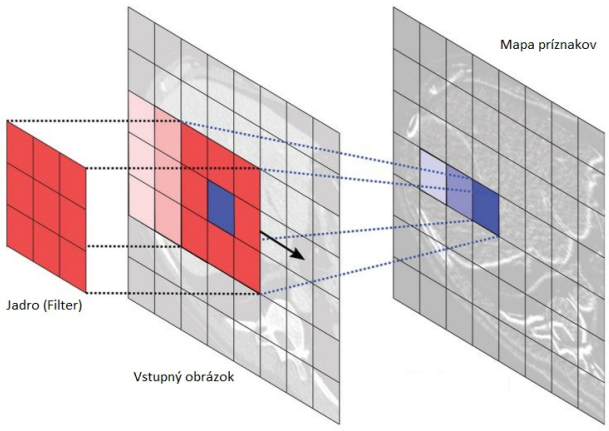
Kde $x = 0, ..., M + N - 2$, funkcia $f(m)$ je filter/jadro nazývané aj ako konvolučná maska. Pre spracovanie obrazu potrebuje funkcie dvoch premenných. Operáciu konvolúcie dokážeme zapísať ako hviezdičku. Následne skrátime tento tvar na

$$f(x, y) * h(x, y) \tag{2.2}$$

Kde f(x, y) je funkcia obrazu, ktorý ideme filtrovať a h(x, y) je filter.

Táto operácia spočíva v tom, že máme vstup, čo pri obrázkoch môže byť matica, ktorá reprezentuje jednotlivé pixely. Tento vstup je vektor hodnôt, na

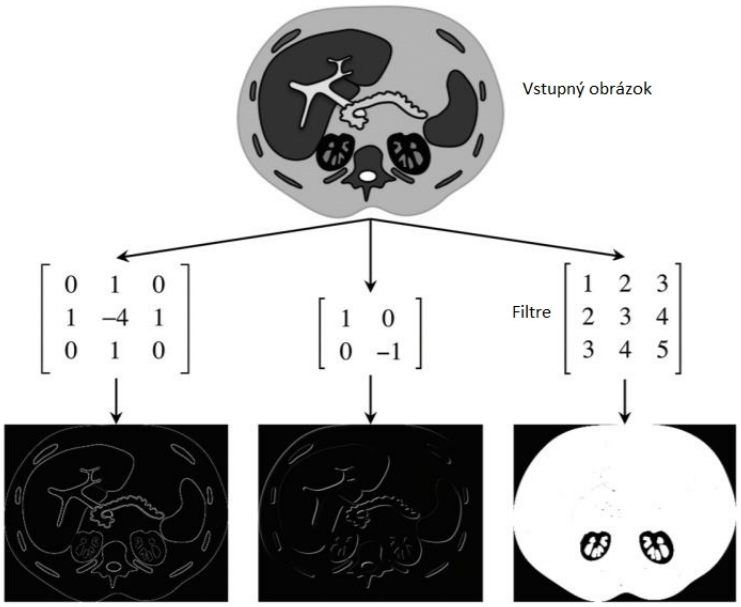
ktorý sa postupne aplikuje filter/jadro (kernel). Filter je podobne ako vstup vektor, ale s oveľa menšou veľkosťou. Tento filter sa vytvára pomocou rôznych spôsobov, či už manuálnym nastavením alebo inicializáciou na základe učenia. Následne sa porovná, či k danému vstupu filter pasuje. Toto sa opakuje pre celý vstup, z ktorého nakoniec vzniká výstup alebo tzv. mapa príznakov (feature map). Tento proces je znázornený na obrázku č. 2.2. Mapa príznakov má vždy menšie rozmery ako vstup, pretože vznikajú okraje bez vyplnených hodnôt. Aby sme predišli tomuto zmenšovaniu výstupu, k vstupu sa pridajú na okraj nulové pixely. Pokiaľ majú dáta viac signálov, napríklad ako farebné obrázky, tak aj veľkosť filtra sa bude zväčšovať a bude mať rovnakú veľkosť, ako je počet signálov. Toto zväčšenie sa bude aplikovať na 3. rozmer filtru. Napríklad, ak máme filter s veľkosťou 3x3, čo bude znamenať, že filter má 9 pixelov, ktoré vykrátime troma, čo značí každý signál.



Obr. 2.2: Aplikovanie filtru na vstupný obrázok [4]

Pre jednu vrstvu sa využije viacero filtrov, pričom každý bude detekovať inú charakteristiku, niektoré budú detekovať len hrany, iné zasa konkrétnejšie charakteristiky. Výsledné mapy príznakov sa neskôr budú ďalej spracovávať a podľa nich sa vyhodnotí klasifikácia. Je zvyklosťou, že najnižšie vrstvy spracovávajú obrázky

všeobecne, a čím prechádzame do vyššej vrstvy, používajú sa konkrétnejšie filtre. Na obrázku č. 2.3 môžeme vidieť príklad týchto filtrov. [5, 4]



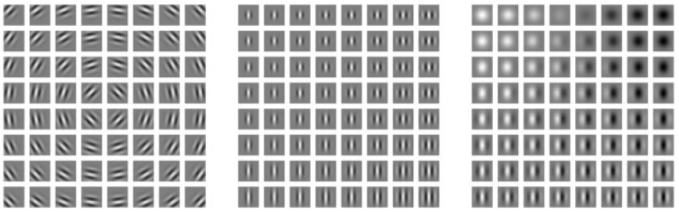
Obr. 2.3: Rôznorodosť filtrov aplikovaných na vstupný obrázok [4]

2.2.4 Gáborové filtre

Gáborové filtre, pomenované podľa Dennisa Gábora, sú lineárne filtre, ktoré sa používajú na detekciu okrajov, analýzu textúr alebo extrakciu znakov. Tieto filtre majú veľmi dobré lokalizačné vlastnosti, či už v priestorovej alebo frekvenčnej oblasti a práve preto sú vhodné pre segmentačné úlohy. Pomocou filtrov sa hľadá v obrázkoch špecifický frekvenčný obsah v danej oblasti alebo jej okolí. Pričom niektoré cicavce, a taktiež aj ľudský zrakový systém, modelujú obsah pomocou Gáborových filtrov. A to vďaka tomu, že je možné opísať niektoré profily jednoduchých bunkových receptívnych polí v mozgovej kôre ako orientované gáborové filtre.

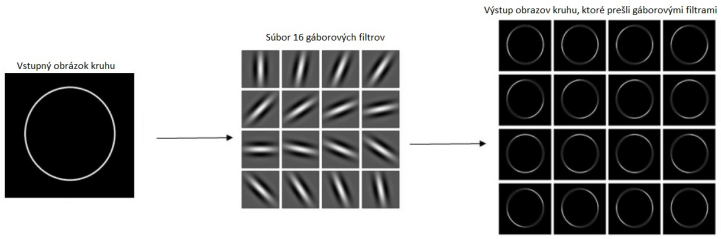
Gáborové filtre v 2D priestore sú tvorené pomocou gaussovej funkcie a si-

nusoidy, pričom je možné ich brať ako sinusoida modulovaná gaussovou funkciou, ktorá sa používa v rôznych orientáciách. Ukážku týchto filtrov môžeme vidieť na obrázku č. 2.4 .[5, 1]



Obr. 2.4: Ukážka gáborových filtrov [5]

Pre lepšie pochopenie aplikácií gáborových filtrov si predstavme biely kruh na čiernom pozadí. Na tento kruh aplikujeme niekoľko gáborových filtrov, hrana detekovaného kruhu je hrana orientovaná v uhle, pod ktorým je orientovaný Gáborov filter [13] (Obrázok 2.5):



Obr. 2.5: Aplikácia gáborových filtrov na biely kruh na čiernom pozadí [13]

Tvar Gáborovej funkcie. Gáborová funkcia je definovaná 2 vstupnými parametrami a 3 parametrami, ktoré kontrolujú tvar a veľkosť Gáborovej funkcie a sú to:

- θ - orientácia normálu na rovnobežné pruhy Gáborovej funkcie,
- ψ - fázový posun sínusovej funkcie,
- σ - sigma / štandardná odchýlka Gaussovej funkcie.

$$h(x,y) = exp\{-\frac{1}{2}[\frac{x^2}{\sigma^2} + \frac{y^2}{\sigma^2}]\} * cos(2\pi X + \Psi)\ [6] \tag{2.3}$$

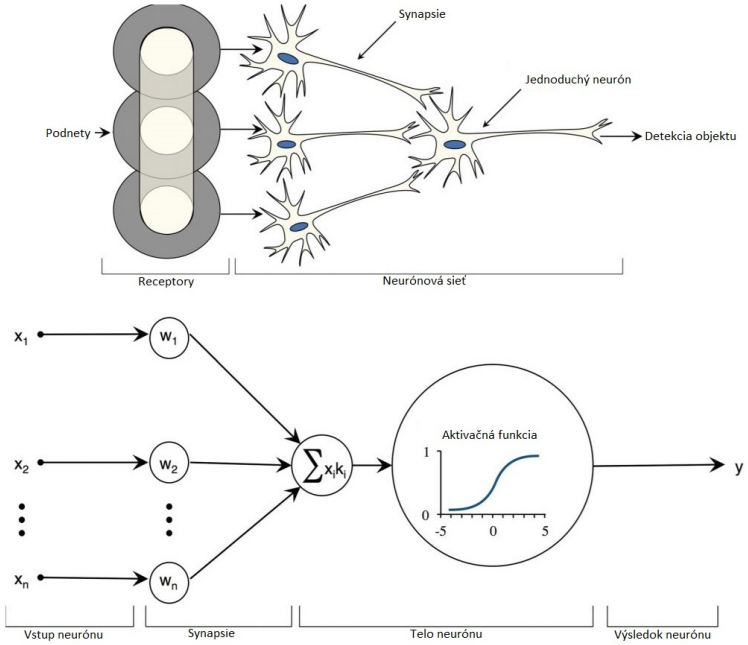
f

2.3 Neurónové siete a hlboké učenie

2.3.1 Neurón ako základná jednotka

Počítačové neurónové siete sú princíp riešenia úloh inšpirovaný prírodou a to zo živých organizmov a ich hlavného orgánu pre riadenie všetkého, mozgu. Mozog sa skladá z biliónov neurónov, ktoré vďaka receptorom, ktoré sú citlivé na určité podnety z vonkajšieho alebo vnútorného prostredia, získavajú informácie, ktoré spracujú a pošlú ďalším neurónom. Neuróny medzi sebou komunikujú chemickými a elektrickými synapsiami. Podľa počtu synapsií dokáže človek rýchlejšie spracovávať podnety. Spracovanú informáciu ako posledné dostanú špecifické neuróny. Po dokončení šírenia a spracovávania podnetu neurónová sieť, jej posledná vrstva, vytvorí výsledok, ako napríklad zdetekovanie objektu pred sebou.

Takto fungujú aj počítačové neurónové siete, dostanú nejaké „podnety" teda vstupy, ktoré sú následne poslané do neurónu, ktorý ich spracuje, či už nejakou jednoduchšou operáciou, akou je sčítanie, alebo zložitejšou, vyhodnotenie pomocou aktivačnej funkcie. Prepojením neurónov vytvárame neurónové siete, ktoré medzi sebou komunikujú a svoje výsledky, po vykonaní určitej operácie, posielajú. Neuróny daný výsledok znova spracujú a znova odošlú. Takéto odosielanie výsledkov prebieha len pri niektorých neurónoch. Aby malo zmysel odosielať výsledky neurónov iným neurónom, každý neurón musí zväčša vykonávať rozdielnu operáciu, teda je na inej úrovni špecifikácie, inej vrstve. Podobnosť ľudskej neurónovej siete a počítačovej neurónovej siete môžeme vidieť na obrázku č. [5]



Obr. 2.6: Podobnosť ľudskej neurónovej siete a počítačovej neurónovej siete [5]

2.3.2 Neurónové siete

Neurónová sieť je zvyčajne rozdelená do viacerých vrstiev, pričom každá vykonáva niečo iné.

Prvá vrstva sa nazýva vstupná vrstva, ktorá reprezentuje vstup dát, ako sú texty, teda jednorozmerné vektory, alebo viacrozmerné, ako pixely, voxely. Posledná sa nazýva výstupná vrstva, ktorá dodáva výsledné informácie, napríklad klasifikáciu obrázku. Existujú aj takzvané skryté vrstvy, ktoré sa nachádzajú vo viacvrstvovom perceptróne. Tieto vrstvy priamo netvoria viditeľný výstup, no počítajú prechodné reprezentácie vstupných prvkov, ktoré sú užitočné v ďalších vrstvách. [5]

2.3.2.1 Perceptrón

Jedna z najzákladnejších a najjednoduchších neurónových sietí sa nazýva perceptrón. Perceptrón alebo binárny klasifikátor je neurónová sieť, ktorá obsahuje len jeden neurón. Do perceptrónu vstupujú signály cez synaptické váhy, ktoré vytvárajú váhový vektor. Pomocou aktivačnej funkcie sa následne vyhodnotí a získame výstup, teda danú klasifikáciu, ktorá môže nadobudnúť len binárnu hodnotu, a to pravda, nepravda alebo jednotka a nula. Zvyčajne ako aktivačnú funkciu perceptrón používa sigmoid, ale v praxi sa viac využíva ReLu funkcia.

$$out(t) = 1 \; ak \; in(t) > 0 \; inak \; 0 \tag{2.4}$$

Perceptrón ako taký dokáže riešiť len lineárne separovateľné problémy. Pravdivostné funkcie, ktoré vie vyriešiť, sú konjunkcia, teda logický súčin (AND), disjunkcia, logický súčet (OR). Nedokáže riešiť exkluzívny súčet (XOR), pretože túto pravdivostnú funkciu nevieme linearizovať v priestore.

Učenie perceptrónu spočíva na takom princípe, že pre každý vstupný vektor z trénovacej množiny sa počíta výstupná funkcia perceptrónu, vďaka ktorým sa upraví váhový vektor tak, aby sa zminimalizovala chyba klasifikácie. [3]

2.3.3 Konvolučné neurónové siete

Konvolučné neurónové siete sú špeciálny druh neurónových sietí, ktoré spracovávajú viacdimenzionálne dáta, napríklad obraz. Aby neurónová sieť mohla byť považovaná za konvolučnú, musí obsahovať aspoň jednu konvolučnú vrstvu. V tejto vrstve prebieha konvolučná operácia, ktorá je opísaná v časti 2.2.3, ktorá vytvára robustnosť konvolučnej neurónovej siete, ktorú viacvrstvový perceptrón nemá. Ten musí kódovať nadbytočné informácie o tvare, orientácii alebo prípadne pozícii ur-

čitých vzorov v obrázkoch.

Typická konvolučná vrstva sa skladá z troch hlavných častí . V prvej časti sa niekoľkokrát paralelne vykonáva konvolučná (convolution) operácia, aby sa vytvorila sada lineárnych aktivácií. V druhej časti sa následne každá lineárna aktivácia použije v nelineárnej aktivačnej funkcii, ako napríklad oprávnená lineárna aktivačná funkcia. Táto časť sa často nazýva aj detekčná časť. Posledná časť využíva zhromažďujúcu (pooling) funkciu na upravenie výstupu konvolučnej vrstvy. Je zvykom, že konvolučná neurónová sieť obsahuje konvolučnú operáciu, aktivačnú funkciu, zvyčajne to je ReLu (rectified linear) [5, 1], ktorá je definovaná ako

$$f(x) = \ max(0, x) \tag{2.5}$$

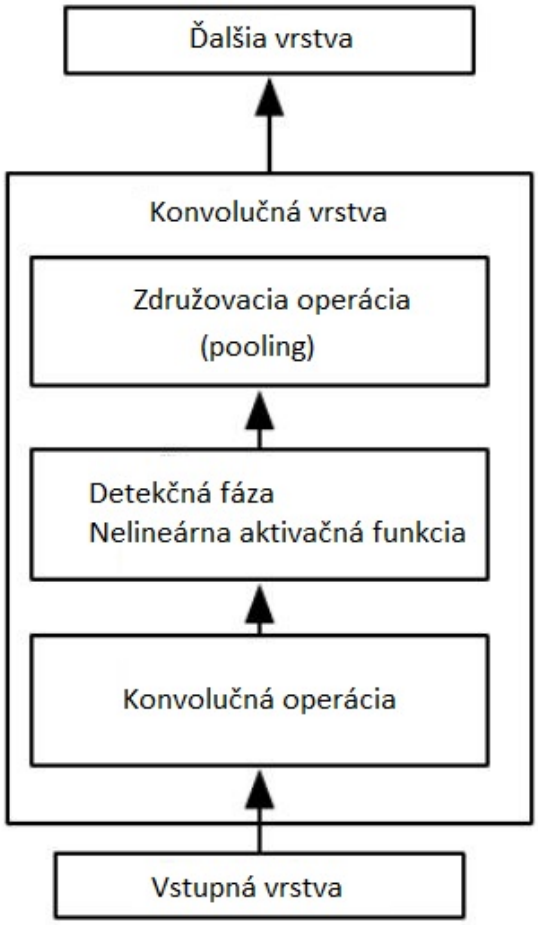
taktiež zhromažďujúca funkcia (pooling) a kvôli možnej ťažkej interpretácií sa ukončí softmax funkciou. Softmax prevádza súčet vstupov na jeden výstupný.[5] (Obrázok 2.7)

$$y = \frac{e^{yi}}{\sum_j e^y j} \text{ [5]} \tag{2.6}$$

2.3.3.1 Členenie dát konvolučných neurónových sietí

Konvolučné siete sú vďaka svojej odolnosti veľmi efektívne a presné, čo sa týka spracovania niekoľkodimenzionálnych dát, a to aj vďaka tomu, že dokážu spracovávať dáta, ktoré nemusia byť rovnakej veľkosti, rozdielnu výšku a šírku vstupných dát.

Hlavné využitie našli preto v spracovaní vizuálnych dát, v obrázkoch a audio dátach, ako sú zvukové vlny, hudba, reč. Tieto niekoľkodimenzionálne dáta obsa-



Obr. 2.7: Ukážka konvolučnej vrstvy [5]

hujú rôzny počet signálov, pričom každý signál predstavuje inú veličinu, či už je to pozícia v priestore, farba alebo samotný čas. Podľa zložitosti signálu vieme dáta rozdeliť na jednoduché a zložité, viacsignálové.

Konkrétne príklady pre dáta s jednoduchými signálmi sú

- 1 dimenzionálny signál napr. zvuk,
- 2 dimenzionálne dáta, napr. čierno-biely obraz,
- 2 dimenzionálne dáta, ktoré vznikli transformáciou zvuku do spektrálnej oblasti čiže do dvojdimenzionálneho vektora: Prvá dimenzia predstavuje frekvenčnú škálu a druhá dimenzia je čas,
- 3 dimenzionálne volumetrické dáta, sú zväčša získané pomocou CT alebo magnetickou rezonanciu, ide o 3D obrazy orgánov alebo iných častí živých organizmov.

Zložitejšie, viackanálové dáta:

- dáta farebného obrázka, kde jednotlivé signály predstavujú finálnu farbu zloženú z RGB (červená, zelená, modrá) modelu,
- farebné videodáta, kde jedna os predstavuje čas, x-ová a y-ová súradnica predstavuje pozíciu v priestore.

2.3.3.2 Pooling operácia (združovacia)

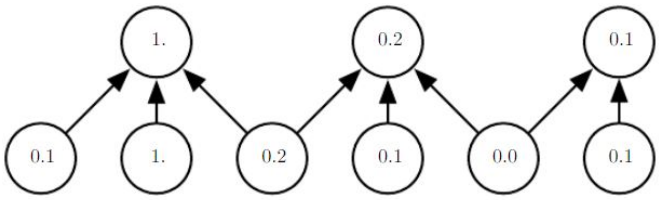
Po vykonaní konvolučnej operácie na vstupný vektor, na mapu príznakov, sa vykoná nelineárna aktivačná funkcia. Na tento výstup sa aplikuje združovacia funkcia, ktorá má za úlohu zmenšiť pôvodnú reprezentáciu a to aproximovateľnou nemennosťou. Nemennosť v tomto prípade znamená, že aj keď zmenšíme veľkosť vstupu, hodnota združovaného výstupu sa nezmení. Ako príklad môžeme uviesť úlohu, kde potrebujeme zistiť, či sa na obrázku nachádza človek. Nepotrebujeme vedieť, kde presne sa človek nachádza, ale len to, či tam je, alebo nie je.

Združovanie (pooling) nám pomáha zefektívňovať konvolučnú vrstvu aj časovo, aj pamäťovo, keďže zmenšuje veľkosť príznakovej mapy z k pixelov na 1 (Obrázok č. 2.8). Vďaka tomuto ďalšia vrstva nemusí prechádzať spomínaných k

pixelov, ale len 1 z danej oblasti. [5]

Medzi najznámejšie združovacie funkcie patrí:

- Max združenie (the max pooling), ktoré zo štvorca susedov vyberie najväčší prvok.
- Priemer z obdĺžnika susedov (the average of a rectangular neighborhood), ktorý zo štvorca susedov vypočíta priemer a ten vráti.
- Vážený priemer (weighted average), založený na vzdialenosti od centrálneho pixela.

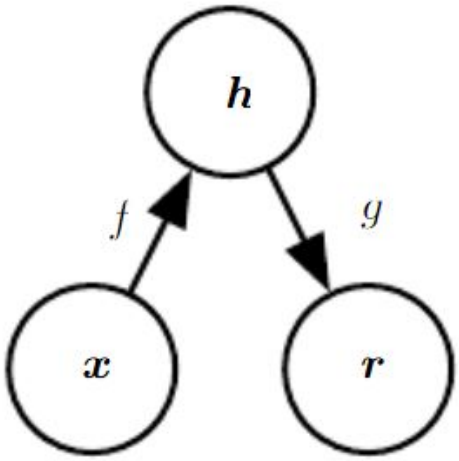


Obr. 2.8: Ukážka aplikovania združovacej funkcie max pooling [5]

2.3.4 Autoenkóder

Autoenkóder je špeciálna verzia doprednej neurónovej siete s typickým mini-batch gradientom klesania počítaný so spätným šírením. Špeciálna funkcia spočíva v tom, že neurónová sieť sa učí skopírovať vstup a následne ho prekopírovať do výstupu (Obrázok č. 2.9). Obsahuje skrytú vrstvu h, ktorá je tvorená kódom, ktorý reprezentuje vstup. Sieť môžeme rozdeliť na dve časti, prvá enkódovacia funkcia $h = f(x)$ a dekódovacia funkcia $r = g(h)$. Pokiaľ by sa mal autoenkóder dokonale naučiť kopírovať, potom táto sieť nemá zmysel a zároveň sa tomuto snažíme vyhnúť preto, že je to nežiadúce. Kvôli tomuto vznikli rôzne varianty autoenkóderu, ktoré obmedzujú túto schopnosť.

Pri kopírovaní vstupu autoenkóderu nás zväčša nezaujíma jeho výstup, ale



Obr. 2.9: Schéma autoencodera, mapujúca vstup x na výstup r pomocou skrytej vrstvy h. [5]

namiesto toho dúfame, že pri učení kopírovania funkcia h získa dôležité parametre. Jednou z možností, ako získať užitočné parametre pre enkódovaciu funkciu h je, že bude mať menšiu veľkosť ako vstupné dáta. Takto upravený autoenkóder sa nazýva nedokončený (undercomplete). Učiaci proces, ktorý dovoľuje nedokončenému autoenkóderu zachytiť najdôležitejšie časti trénovacích dát. Tento proces môžeme zjednodušiť na minimalizovanie stratovej funkcie $L(x, g(f(x)))$, kde L je stratová funkcia penalizovaná $g(f(x))$. Pokiaľ dekóder je lineárny a L je druhá mocnina strednej chyby. Nedokončený autoenkóder sa naučí preklopiť rovnaký podpriestor ako PCA. V tomto prípade sa autoenkóder, cvičený na kopírovanie vstupu, naučil hlavný podpriestor ako vedľajší efekt. Autoenkóder s nelineárnym enkódovaním a nelineárnym dekódovaním sa môže naučiť silnejšiu nelineárnu generalizáciu PCA. No pokiaľ nemá enkóder a dekóder určenú kapacitu, do ktorej sa môže učiť, naučí sa úlohu kopírovania bez extrahovania užitočných informácií o dátach. Podobný problém nastáva, pokiaľ skrytá vrstva a jej funkcia h má väčšiu alebo rovnakú

veľkosť ako vstup. Preto sa odporúča limitovať veľkosť enkódera a dekódera na menšiu veľkosť, ako má vstup. Vďaka tomuto sa bude môcť sieť viacej naučiť pri kopírovaní dát a stane sa robustnejšou. (Obrázok 2.10)

Riedky autoenkóder (sparse) je jednoduchšou verziou autoenkóderu, ktorá trénovacie kritériá penalizuje $\Omega(h)$ funkciou na skrytej vrstve h funkcie. Tento typ autoenkóderu, ako aj iné, je typicky používaný pre inú úlohu, ako je napríklad klasifikácia. Pričom je upravený tak, aby bol citlivý na jedinečné štatistické črty dát. O penalizačnej funkcií $\Omega(h)$ uvažuje ako o regularizačnom výraze pridaného do doprednej siete, ktorej primárnou úlohou je kopírovať vstup na výstup.

$$-logpmodel(h) = \ i\lambda|h_i| - log\lambda 2 = \omega(h) \tag{2.7}$$

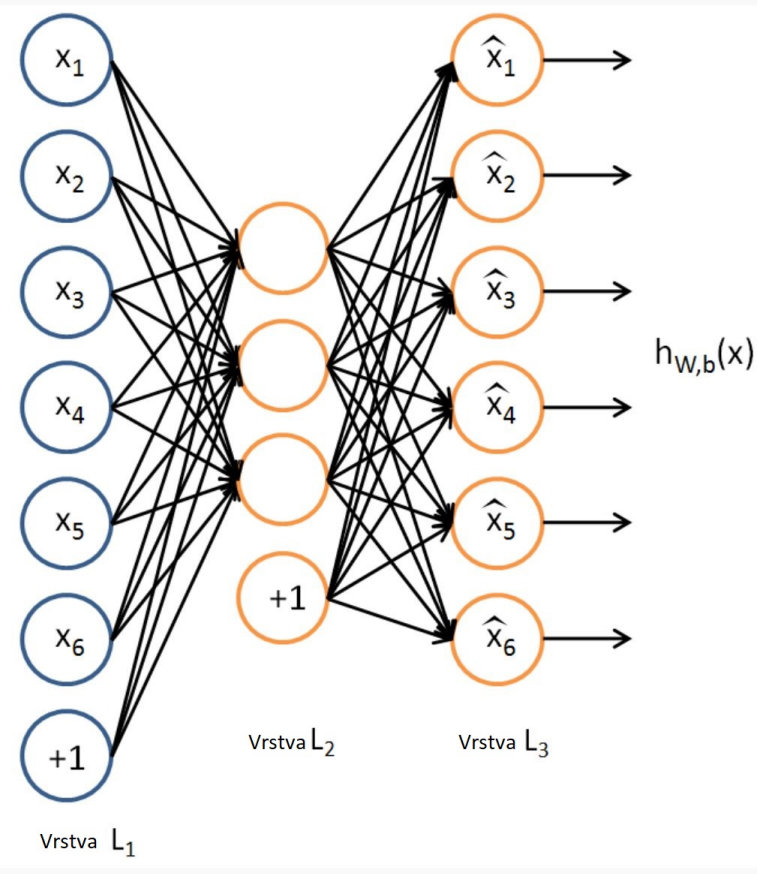
Denizujúci Autoencoders (denoising) na miesto pridania penalizačnej funkcie zmení výraz pre rekonštrukčnú chybu hodnotovej funkcie. Tradične minimalizuje niektoré funkcie, ako $L(x, g(f(x)))$, kde L je stratová funkcia penalizujúca g(f(x)) pre rozdielne x. Na miesto toho upraví funkciu $L(x, g(f(x^{-})))$, kde x^{-} je kópiou x, ktorá nebola poškodená pri predošlom spracovávaní.

Ďalšími možnými prístupmi sú napríklad zmena penalizačnej funkcie $\omega(h, x)$ na $\omega(h, x) = \sum_i ||\Delta xhi||^2$. Takýto autoenkóder sa nazýva kontraktívny(contractive). [5]

2.3.5 Transfer learning

Trénovanie neurónovej siete je veľmi náročný proces,či už sa to týka času alebo množstva dát, ktoré sú potrebné na trénovanie siete. Tento problém nám rieši transfer learning.

Medzi základné dva spôsoby transfer learning-u patria:



Obr. 2.10: Ukážka autoenkódovej neurónovej siete [10]

- vytvorenie modelu od začiatku,
- využitie už natrénovaného modelu.

2.3.5.1 Vytvorenie modelu od začiatku

Vytvorenie modelu od začiatku spočíva v tom, že si vytvoríme architektúru modelu neurónovej siete tak, že bude schopný extrahovať vzory a váhy, ktoré sa naučí. Následne potom použijeme tento model ako východiskový bod pre model s podobnou úlohou, teda tou, ktorá bola pôvodne definovaná. Prvá časť tohto spôsobu je v podstate rovnaká ako u všetkých, že trénujeme dáta, len s tým rozdielom,

že cieľom nie je ho ďalej použiť, ale len extrahovať váhy a natrénované vzory dát. [12, 9]

2.3.5.2 Využitie natrénovaného modulu

Pri druhom spôsobe sa používa už zhotovený model neurónovej siete. Tento model by mal robiť podobnú úlohu ako pôvodné riešenie. Teda zväčša sa použije nejaká verejná neurónová sieť. Podľa veľkosti našich dát a iných vlastností sa následne bude táto sieť upravovať, zväčša sa vynechajú niektoré vrstvy alebo odstránia. Ďalšie rozhodovanie o úprave neurónovej siete sa vyvíja od veľkosti vstupného datasetu, ktorý je daný pôvodnou úlohou. Môžeme odstrániť posledné vrstvy alebo vymažeme stratovú funkciu, jej váhy a natrénované vzory. Dataset z pôvodnej úlohy preženie sieťou, aby sa posledné vrstvy, tie, ktorým sme vymazali váhy a vzory, znova vytvorili a natrénovali. Pokiaľ dataset z pôvodnej úlohy obsahuje veľký počet dát, prvé vrstvy sa preskočia, aby sa zbytočne nepretrénovali, teda nevznikli nežiadúce efekty, ale trénovať sa budú len posledné vrstvy. Pokiaľ by veľkosť datasetu z pôvodnej úlohy nebola veľká, môžeme trénovať na všetkých vrstvách. [12, 9]

2.3.5.3 Príklad transfer learning

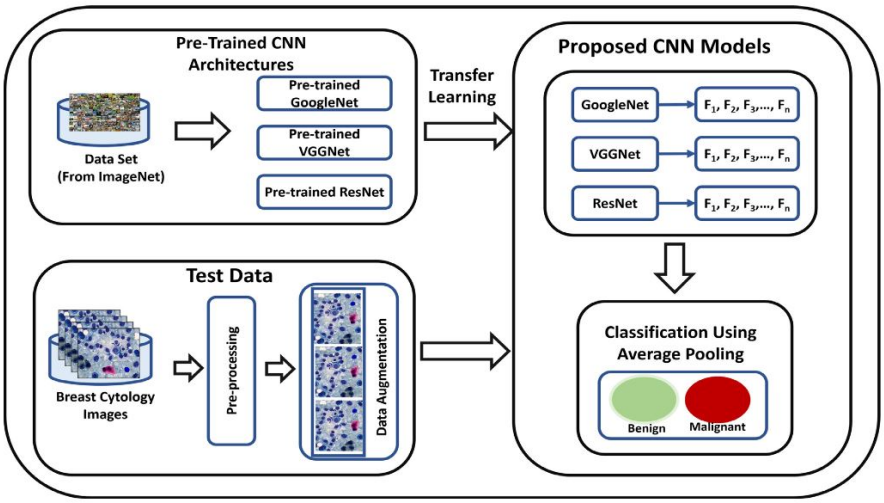
Jednou z najbežnejších aplikácii transfer learningu sú úlohy týkajúce sa klasifikácie alebo predikcie. Používajú sa hlavne konvolučné neurónové siete, kde nižšie vrstvy spracovávajú dáta, obrázky alebo zvukové stopy na všeobecnejšej úrovni, teda len základné vlastnosti, ako sú hrany, farby a iné. Odstraňujú sa teda len tie vrstvy, ktoré presne klasifikujú dáta, t. j. do ktorej skupiny dáta patria. Neurónová sieť sa znova ďalej trénuje, aby rozoznávala nové skupiny. Zväčša sa používajú najznámejšie modely s veľmi dobrou presnosťou klasifikácie ako ResNet od Microsoftu, Inception Model od Googlu alebo VGG model z Oxfordu. [8]

2.4 Publikované práce

2.4.1 Publikácia s názvom A novel deep learning based framework for the detection and classification of breast cancer using transfer learning

V roku 2019 autori článku A novel deep learning based framework for the detection and classification of breast cancer using transfer learning zlepšili aplikáciu konvolučných neurónových sietí a to využitím transfer learningu a skombinovaním viacerých architektúr konvolučných neurónových sietí namiesto jednej architektúry. Autori použili 3 architektúry a to GoogLeNet, VGGNet a ResNet. Pričom kombinovaná sieť obsahuje plne prepojenú vrstvu s klasifikačnou úlohou. Celý tento proces nazvali „proposed framework" (Obrázok č. 2.11). Následná metóda obsahuje 2 hlavné časti a to:

- pre procesovanie dát a zväčšovací proces,
- pretrénovanie konvolučných neurónových sietí pre úlohu extrakcie.



Obr. 2.11: Diagram Proposed Frameworku. [15]

Preprocesovanie dát a zväčšovací proces

Preprocesovanie dát (Data pre-Processing) je jedným z esenciálnych krokov pre odstránenie chybných a poškodených dát. Zväčšovací proces (Augmentation processing) je proces, v ktorom sa zväčšuje veľkosť datasetu a redukujú sa „overfitting" problémy. Veľkosť, na ktorú sa bude meniť dataset, je určená geometrickou transformáciou či už zväčšením farieb, alebo transformáciou obrazov, ako sú rotácia, orezávanie a otáčanie.

Pretrénovanie konvolučných neurónových sietí pre úlohu extrakcie

Pretrénovanie architektúr konvolučných neurónových sietí pre úlohu extrakcie, kde sa na začiatku separovali architektury a následne sa spojili do plne prepojenej vrstvy určenej na klasifikáciu. Jednotlivé architektúry sú:

- GoogleNet je malá sieť, ktorá obsahuje 3 konvolučné vrstvy, združovaciu vrstvu a 2 plne prepojené vrstvy. V tomto sa pomocou rôznych kombinácií konvolučných filtrov rôznych veľkostí vytvára jeden filter, ktorý nielen redukuje množstvo parametrov, ale aj minimalizuje výpočtovú náročnosť.

- VGGNet je podobná sieť ako AlexNet s pridanými konvolučnými vrstvami. Táto sieť obsahuje 13 konvolučných rektifikovaných združovacích vrstiev a 3 plne prepojené vrstvy. Konvolučná sieť využíva 3x3 veľkosť filtru a združovacia vrstva zmenšuje výstup na 2x2.

- ResNet je veľmi hlboká neurónová sieť, ktorá dosahuje veľmi dobré výsledky pri klasifikácii na úlohách v ImageNet. Táto sieť kombinuje konvolučné filtre s rôznymi veľkosťami, ktoré riešia problém degradácie a redukuje čas na trénovanie.

CNN Architectures	Lens 100X	Magnification 140X 200X 500X	Average Classification Accuracies
GoogleNet	90.4	93.7 95.3 94.6	94.5%
VGGNet	90.8	94.8 96.7 94.2	94.15%
ResNet	91.5	93.3 95.4 97.2	94.35%
Proposed Framework	96.8	96.9 97.8 98.6	97.525%

Tabuľka 2.1: Porovnanie výsledkov jednotlivých architektúr a výsledného frameworku. [15]

Dataset

Dataset obsahoval 2 druhy dát. Prvý bol zo štandardného benchmarku a druhý bol získaný lokálne z nemocnice Peshawar v Pakistane. Obsahoval okolo 8000 obrázkov, kde prvých 6000 bolo použitých na trénovanie a 2000 na testovanie.

Výsledok

Tento framework (práca) dosiahol veľmi dobré výsledky, pričom kombinovanie 3 CNN architektúr pomocou transfer learningu a hlavne zväčšovaciemu procesu malo za následok vyššiu presnosť a to 97%, pričom jednotlivé boli v priemere o 3% – 4% horšie. Presné výsledky a porovnania môžeme vidieť v tabuľke č. 2.1. Proposed Framework dosiahol excelentné výsledky aj napriek tomu, že sa sieť nevytvorila od začiatku . V budúcnosti obe vytvorené techniky môžu zvýšiť presnosť klasifikácie konvolučných neurónových sietí. [15]

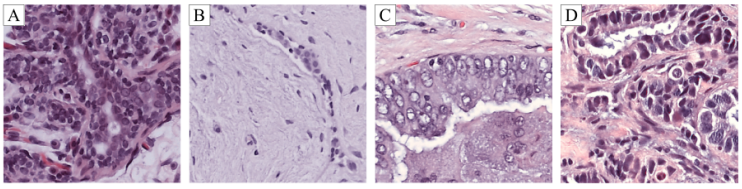
2.4.2 Publikácia s názvom Classification of breast cancer histology images using Convolutional Neural Networks

Publikácia bola zverejnená v roku 2017 talianskou univerzitou. V tejto práci sa využívajú konvolučné neurónové siete pre klasifikáciu rakoviny prsníka. Obrazy sa rozdeľujú do 4 kategórií a to normálne tkanivo, počiatočné lézie, in situ karcinóm

a invazívny karcinóm. V práci sa používajú konvolučné neurónové siete samostatne a aj s pomocou Support Vector machine klasifikácie. Konvolučná neurónová sieť je navrhnutá pre analýzu histologických obrázkov rakoviny prsníka H&E. Okrem toho je architektúra konvolučnej neurónovej siete navrhnutá tak, aby dokázala získať informácie z viacerých histologických stupňov vrátane nuklidu, usporiadania nuklidu a celkového usporiadania štruktúry. Zohľadnením konvolučnej neurónovej siete sa môže použiť aj na klasickú klasifikačnú úlohu celoobrazových histologických obrazov.

Dataset

Dataset pozostáva z obrazov s vysokým rozlíšením (2040 x 1536 pixelov), ktoré sú anotované H&E označeniami z Bioimaging 2015 breast histology súťaže. Každý obraz je označený jednou zo štyroch skupín. Toto označenie bolo vykonané dvoma doktormi, bez vyznačenia špecifickej oblasti. Dataset pozostáva z 249 trénovacích dát a oddelených 20 testovacích dát. Príklad týchto dát môžeme vidieť na obrázku č. 2.12. Zastúpenie jednotlivých tried je rovnomerné. Pre zväčšenie datasetu sa nad dátami vykonala operácia zväčšenia (augmentation).

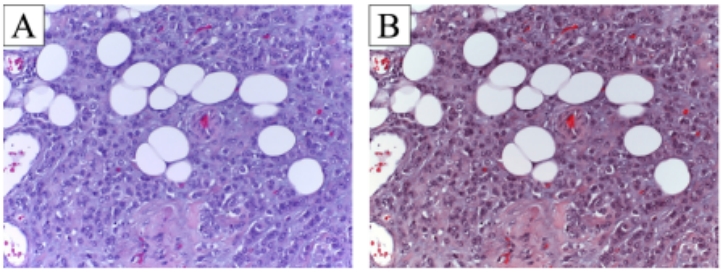


Obr. 2.12: Ukážka histologických dát v datasete. [2]

Preprocesovanie dát

Preprocesovanie dát spočíva v 3 krokoch. V prvom kroku sa farba obrázkov prekonvertuje do optickej hustoty (optical density) použitím logaritmickej transfor-

mácie. Potom sa použije rozklad singulárnych hodnôt (singular value decomposition) na optickú hustotu, aby sa našli 2D projekcie s vysokou odchýlkou. Následne sa použije transformácia farebného priestoru (color space transform) na originálny obraz. Nakoniec sa histogram obrazu zväčší, aby sa pokryli dáta s menšími veľkosťami, približne 90% dát, a to dynamicky. (Obrázok č. 2.13)



Obr. 2.13: Ukážka preprocesovania obrazu, kde obraz A je pôvodný a B je po prepocesovaní. [2]

Klasifikácia obrazov

Procedúra pre klasifikáciu obrázku je nasledovná. Najprv sa originálny obrázok rozdelí do susedných neprekrývajúcich sa súborov (patch). Kategória jedného súboru je počítaná v patch-wise algoritmoch, kde trénujeme konvolučnú neurónovú sieť a konvolučnú neurónovú sieť s využitím support vector machine algoritmov. Určenie kategórie sú získané pomocou jednej z troch metód.

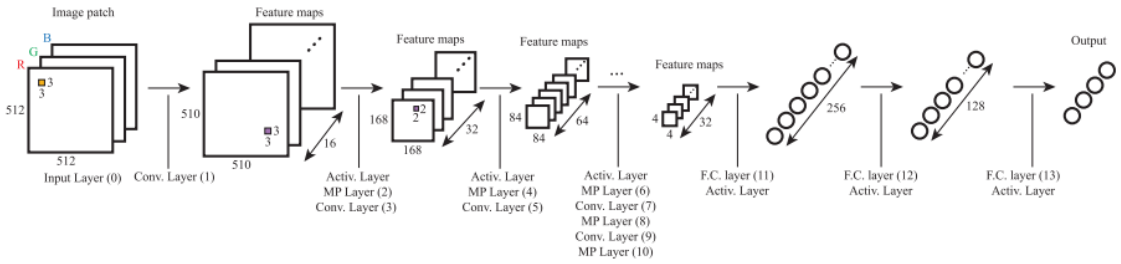
- Primárne hlasovanie (Majority voting), kde značenie obrazu je vybrané pomocou najčastejšieho značenia súboru (patch).
- Maximálnej pravdepodobnosti (Maximum probability), kde značenie súboru (patch) s najväčším zastúpením v obraze je vybrané pre označenie obrazu.
- Suma pravdepodobnosti (Sum of probability), kde značenie obrazu je definované sumou značení súborov s najvyšším zastúpením.

Konvolučné neurónové siete pre patch-wise klasifikáciu

V tomto článku sa používa konvolučná neurónová sieť s dopredným šírením (Obrázok č. 2.15), pričom je špecializovaná na rozpoznávanie vizuálnych vzorov. Neuróny sú prepojené s prekrývajúcimi lokálnymi súbormi obrazu a usporiadané v konvolučných mapách, kde všetky zdieľajú rovnaké váhy. To umožňuje konvolučným mapám pracovať s nimi ako lokálnymi filtrami, ktoré detekujú rovnaké vzorce pre všetky obrazy. Redukujú tiež celkový počet parametrov pre trénovanie. Vstupná vrstva obsahuje 3 signály 512 x 512 pixelov, ktoré zodpovedajú štandartnému RGB modelu. HĺbkA a počet máp je definovaná v tabuľke 2.14, v ktorej sa nachádzajú konvolučné vrstvy, max-pooling vrstvy (združovacie), ktoré znižujú veľkosť výstupu bez zvýšenia parametrov v sieti. Tieto vrstvy max-pooling a konvolučná obsahujú nelineárnu aktivačnú funkciu ReLu. V konvolučnej neurónovej sieti sa nachádzajú aj plne prepojené vrstvy, ktoré produkujú finálnu klasifikáciu. Výstupná vrstva je zložená zo 4 neurónov, pričom každý prislúcha jednej kategórii a sú normalizované softmax funkciou.

	Layer type	Maps & Neurons	Filter size	Effective receptive field	Effective receptive field (μm)
0	Input	3M × 512 × 512N		1 × 1	0.4 × 0.4
1	Convolutional	16M × 510 × 510N	3 × 3	3 × 3	1 × 1
2	Max-pooling	16M × 170 × 170N	3 × 3	5 × 5	2 × 2
3	Convolutional	32M × 168 × 168N	3 × 3	11 × 11	4.6 × 4.6
4	Max-pooling	32M × 84 × 84N	2 × 2	14 × 14	5.9 × 5.9
5	Convolutional	64M × 84 × 84N	3 × 3	26 × 26	11 × 11
6	Max-pooling	64M × 42 × 42N	2 × 2	32 × 32	13 × 13
7	Convolutional	64M × 42 × 42N	3 × 3	56 × 56	24 × 24
8	Max-pooling	64M × 14 × 14N	3 × 3	80 × 80	34 × 34
9	Convolutional	32M × 12 × 12N	3 × 3	152 × 152	63.8 × 63.8
10	Max-pooling	32M × 12 × 12N	3 × 3	224 × 224	94.1 × 94.1
11	Fully-connected	256N		512 × 512	215 × 215
12	Fully-connected	128N		512 × 512	215 × 215
13	Fully-connected	4N		512 × 512	215 × 215

Obr. 2.14: Zloženie konvolučnej neurónovej siete. [2]



Obr. 2.15: Štruktúra konvolučnej neurónovej siete. [2]

Classifier	No classes	Initial	Extended	Overall
CNN	4	72.5	59.4	66.7
	2	80.4	74.0	77.6
CNN + SVM	4	72.9	55.2	65.0
	2	82.9	69.3	76.9

Tabuľka 2.2: Výsledky patch-wise techniky konvolučnej neurónovej siete. [2]

Výsledky

Patch-wise technika dosiahla presnosť a citlivosť zobrazenú v tabuľke 2.2. Celková presnosť je 66,7% pre konvolučnú neurónovú sieť a pre konvolučnú neurónovú sieť s podporou support vector machine algoritmu to je 65%. Presnosť systému je nízka, pre rozšírený dataset kvôli zníženej časovej náročnosti. Celková presnosť sa zvýši, pokiaľ sa zmení počet kategórií zo štyroch na dve. Teda ak triedy normálne tkanivo a začiatočné lézie, in situ a invazívny karcinóm spojíme, dosiahneme až 80% presnosť. [2]

2.4.3 Publikácia Transfer learning based deep CNN for segmentation and detection of mitoses in breast cancer histopathological images

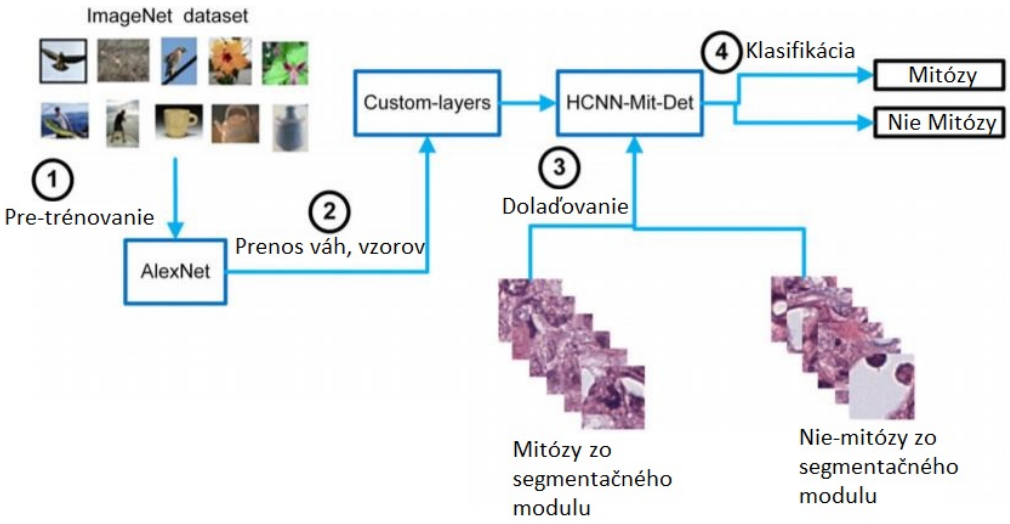
V práci o segmentácii a detekovaní mitózy v rakovine prsníka sa používajú dve konvolučné neurónové siete. Tieto siete sú navzájom prepojené. Segmentačná

používa Transfer learning, kde model, od ktorého preberá váhy, je ImageNet sieť, ktorá obsahuje približne 1000 kategórií na rozpoznávanie prirodzených farebných obrázkov.

Prvá konvolučná neurónová sieť sa používa na segmentáciu mitóz. Súbory (patch), ktoré sú extrahované, slúžia ako vstup do druhej, ktorá je hybridno konvolučná a to tak, že kombinuje Weight Transfer a custom vrstvu pre finálnu klasifikáciu. Vysledky klasifikácie sú rozdelené do dvoch kategórií, na „mitózy" a „nie mitózy". Vďaka týmto dvom fázam, učením 2 neurónových sietí, sa zmenšuje efekt nerovnováhy obsadenosti jednotlivých kategórií v histologických dátach.

WorkFlow celého modelu

Nápad použitia pretrénovanej konvolučnej neurónovej siete pre segmentáciu mitózy nám poskytuje 3 hlavné výhody a to (i) produkuje pomerne rovnocenný dataset pre trénovaciu a validačnú klasifikáciu; (ii) povoľuje použitie architektúry pre hlbokú konvolučnú neurónovú sieť bez pretrénovaných váh z modela alebo predošlého trénovania na veľkom datasete a (iii) redukuje čas pre trénovanie. Úlohou segmentačného modelu, teda transfer learning (TL-Mit-seg) založeného na segmentácií mitózy je vytvoriť vstup pre hybridnú konvolučnú neurónovú sieť, ktorá detekuje kategóriu daného vstupu. Celkový „workflow" tejto siete je teda taký, že najprv dostaneme nepredspracované farebné obrázky, ktoré pomocou preprocessing-u upravíme, najprv normalizáciou škvŕn, ďalej anotáciou mitóz následným orezávaním na súbor (patch) 512x512 pixelov s 80 pixelmi pre prekrytie. Potom sa vykoná úloha segmentácie s TL-Mit-seg modelom, ktorá označí obrázok za „true positive" alebo „false positive" a extrahuje 80x80 pixelové súbory. Súbory, ktoré sú výstupom segmentačnej siete, budú vstupmi pre hybridnú konvolučnú neurónovú sieť (HCNN-Mit-Det), ktorá vykoná klasifikáciu daného súboru. Výstupom tejto siete je už výsledná klasifikácia. (Obrázok 2.16)



Obr. 2.16: WorkFlow pre neurónovú sieť (architektura celeho modelu). [19]

Dataset

Trénovacie dáta obsahujú obrázky od 73 pacientov, kde 34 pacienti boli oddelení pre evaluačný účel. Dataset pochádza z TUPAC16 úlohy a MITOS12 a MITOS14. Vo všetkých datasetoch sú obrazy získavané skenovaním s 40% zoom a približná oblasť 2mm štvorcových. Testovacie dáta obsahujú 34 prípadov z TUPAC16 súťaže. Tieto dáta sú z podobného zdroja ako trénovacie dáta.

Výsledky

Dáta z datasetu pre TUPAC16 získali presnosť 66,7%, pričom pokiaľ sme použili pri validácií dáta aj z MITOS12 a MITOS14, získali sme presnosť 65,1%. No pokiaľ bol model trénovaný od začiatku na všetkých dátach, jednotlivé presnosti sa zmenšili, teda pokiaľ validačný dataset bol TUPAC16 a presnosť bola 65%. Validačný dataset pozostával z oboch zdrojov, presnosť bola 63,6% a pri MITOS12 a MITOS14 bola 59,7%. Tieto výsledky sú zhrnuté v tabuľke 2.3

[19]

Train	Validate	F-measure
Source-A	Source-A	0.667
	Source-A and B	0.651
	Source-B	0.599
Source-A and B	Source-A	0.651
	Source-A and B	0.638
	Source-B	0.597

Tabuľka 2.3: Výsledky pre segmentáčnú a detekčnú neurónovú sieť (zdroj A je TUPAC16, zdroj B je MITOS12 a MITOS14). [19]

2.4.4 Publikácia Rotation Equivariant CNNs for Digital Pathology

V tejto práci sa ne miesto klasickej konvolučnej neurónovej siete, používa upravená skupinová konvolučná neurónová sieť, ktorá klasickú generalizuje. Pričom sú ekvivariantné v rámci skupín symetrie, ako je skupina G, skupina p4 s 90 stupňovou rotáciou alebo G, skupina p4 m.

P4 a p4 m, sú charakteristické kanály prichádzajúce do skupín po 4 alebo 8, čo zodpovedá 4 čistým otáčkam v p4 alebo 8 roto-reflekcia v p4m. V prvej vrstve sa vyrábajú pomocou vzorca:

$$[f * \psi](g) = \sum_{y \in Z^2} \sum_{k=1}^{K} f_K(y)\psi_k(g^{-1}y) \tag{2.8}$$

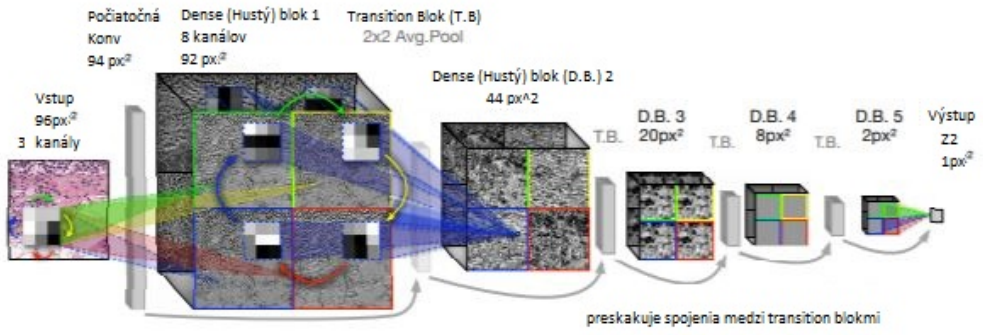
kde $g = (r, t)$ je roto-translacia (v prípade $G = p4$ alebo roto-reflekcio-translacia (v prípade $G = p4m$)

V ďalších vrstváchj su funkčné mapy aj filtre funkciami na G, a tieto sú kombinovane pomocou $G- > G$ konvolúcie:

$$[f * \psi](g) = \sum_{h \in Z^2} \sum_{k=1}^{K} f_K(h)\psi_k(g^{-1}h) \tag{2.9}$$

V poslednej vrstve sa používa skupinová vrstva, aby sa zabezpečilo, že výstup je buď invariantný (pre klasifikačné úlohy) alebo ekvivariantný ako funkcia v rovine (pre segmentačné úlohy, kde sa má výstup transformovať spolu so vstupom)

Následne pomocou týchto vrstiev G-CNN sa vytvárajú husté vrstvy (Dense), a pomocou nich vytvorili špeciálnu architektúru tzv. DenseNet, model tejto architektúry môžte vidieť na obrázku č. 2.17



Obr. 2.17: Architektúra DenseNet siete. [16]

Dataset

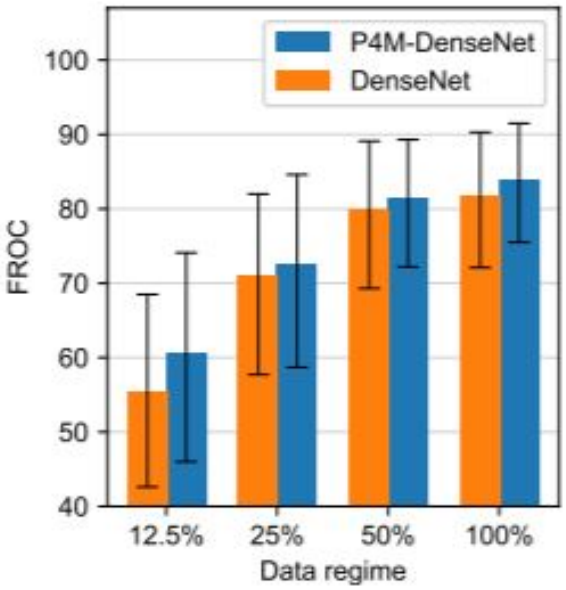
Na overenie a trénovanie sa používal dataset PCAM a Camelyon16 (PCAM) [17]. Camelyon16 obsahguje 400 H&E zafarbených WSI rezov, sentinelových lymfatických uzlín rozdelených do 270 podložných sklíčok s anotáciami na úrovni pixelov pre trénovanie a 130 neoznačených snímok na testovanie. Výrezy boli získané a digitalizované v 2 rôznych centrách s použitím objektívu 40x. V datasete Camelyon16 sa úspešnosť meria pomocou FROC krivky na lokalizáciu nádoru. PCAM, ktorý obsahuje histologické výrezy extrahované z Camelyon16, pričom tieto obrázky sú v tvare 96x96x3. Trénovacií časť datasetu obsahuje 262 144 obrázkov, testovacia časť obsahuje 32 7678 a validačná časť taktiež 32 7678 obrázkov.

Network	K	#W	NLL	Acc	AUC
P4M-DenseNet	64	119K	0.260	89.8	96.3
P4M-DenseNet M	24	19K	0.273	89.3	95.8
P4-DenseNet	48	125K	0.329	89.0	94.5
DenseNet+	24	128K	0.306	88.1	95.1
DenseNet+ M	64	902K	0.365	87.2	94.6
DenseNet	24	128K	0.315	87.6	95.5

Tabuľka 2.4: Výsledky pre PCAM dataset. [16]

Výsledky

Data z datasetu PCAM, dosiahli pre DenseNet so skupinou P4M 89.8 presnosť a pre DenseNet so skupinou P4 89 presnosť. Rozsiahlejšie výsledky môzme vidieť v tabuľke číslo 2.4. V datasete Camelyon16 môžme vidieť aj porovnanie vysledkov DenseNet a DenseNet so skupinou P4M na obrazku č. 2.18, pričom výsledky na FROG krivke sú nasledovné pre P4M DenseNet je to 84.0 a pre DenseNet 81.7. Rozsiahlejšie výsledky môžme vidieť v tabuľke č. 2.5. [16]



Obr. 2.18: Porovnanie výsledkov pre DenseNet so skupinou P4M a DenseNet. [16]

Model	Data	FROC
P4M-DenseNet 123k params	100%	84.0 (75.5, 91.5)
	50%	81.5 (72.2, 89.3)
	25%	72.6 (58.7, 84.6)
	12.5%	60.7 (46.0, 74.1)
DenseNet 126k params	100%	81.7 (72.1, 90.3)
	50%	80.0 (69.3, 89.1)
	25%	71.0 (57.7, 82.0)
	12.5%	55.4 (42.6, 68.5)

Tabuľka 2.5: Výsledky pre Camelyon16 dataset, FROC skóre. [16]

Kapitola 3

Návrh

3.1 Koncept riešenia

V návrhu nášho riešenia sa zameriame na metódy učenia sa s učiteľom pomocou hlbokých neurónových sietí. Použijeme konvolučné neurónové siete, ktoré najlepšie dokážu rozpoznávať vzory v obrazových dátach. Naším hlavným cieľom bude porovnať viaceré metódy tvorenia filtrov v prvých vrstvách neurónových sietí. Na skúmanie a porovnávanie sme si vybrali viaceré prístupy: backpropagation, autoencoder, transfer learning a filtre generované pomocou Gáborovej funkcie, teda Gáborové filtre. Jednotlivé princípy dokážeme rozdeliť do dvoch kategórií, a to podľa toho, ako tvoria filtre – t. j. automaticky (ktoré sieť sama inicializuje) a manuálne (mi dodáme sieti filtre, ktoré už ďalej upravovať nebude). Hlavnou myšlienkou porovnania bude pripojiť na začiatok spomínané prístupy tvorenia filtrov do existujúcej architektúry a následne trénovať na prístupných datasetoch.

Cieľom jednotlivých prístupov a celkovo neurónovej siete bude dosiahnuť čo najlepšie výsledky, ktoré budeme merať vo viacerých metrikách. Jednotlivé prístupy budeme modifikovať a zisťovať, ktoré kombinácie parametrov vykazujú

najlepšie výsledky. Parametre, ktoré budeme meniť v prístupoch, budú závisieť od prístupu, ale hlavne chceme meniť veľkosť a počet daných filtrov (kernelov). Ostatné vrstvy, teda iné ako tie v jednotlivých prístupoch, nebudeme už počas trénovania a porovnávania prístupov meniť. Úloha, ktorú budú mať prístupy, je jednotná a to klasifikovať do dvoch tried, čo sa týka histologických dát 5.1.1, ktoré použijeme a budeme klasifikovať na malígne a nemalígne. Dáta štrukturovaných povrchov 5.1.2 budeme klasifikovať na pozitívny výskyt objektu a negatívny výskyt objektu.

3.1.1 Architektúra

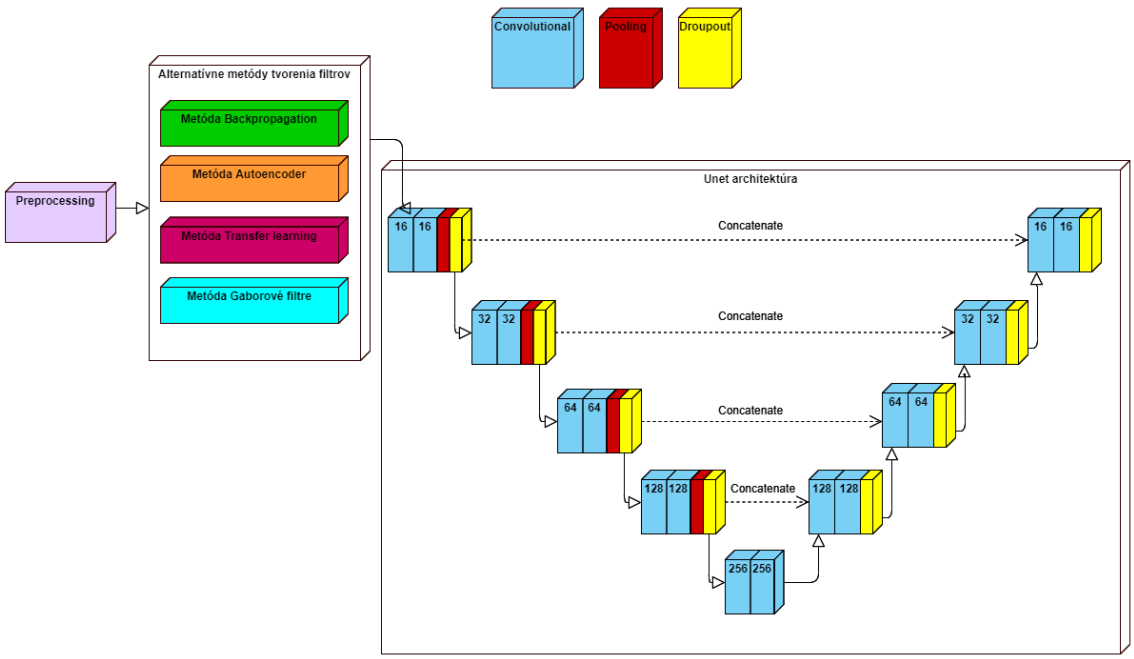
Ako existujúcu architektúru použijeme Unet neurónovú sieť s hĺbkou 9, ktorá sa skladá zo špeciálnych konvolučných vrstiev, ktoré obsahujú 2 konvolučné vrstvy, pričom jedna je skrytá vrstva. Túto architektúru sme vybrali kvôli jej najlepším výsledkom, pričom v prvotných testoch dosiahla veľmi dobrú presnosť a to 80%. Iné architektúry, ktoré sme skúšali, nedosahovali tak dobré výsledky, pričom skúšané boli ResNet s hĺbkou 8, VGG16 a vlastné neurónové siete s maximálnou hĺbkou 5 a minimálnou 2. Celkový návrh spomínaného modelu môžeme vidieť na obrázku č. 3.1.

3.1.1.1 Implementácia architektúry

Unet architektúra využíva špecifickú konvolučnú vrstvu, ktorá obsahuje viaceré konvolučné operácie spolu s dávkovo normalizačnou (batchnormalization) vrstvou a aktivačnou funkciou ReLu. Takéto konvolučné vrstvy sa v architektúre nachádzajú konkrétne 2 s ďalšími vrstvami ako združovacou (pooling) vrstvou a vynechávacou (dropout) vrstvou . Toto je štandartná implementácia každej konvolučnej vrstvy v Unet architektúre, ktorú používame. Jej implementáciu môžeme vidieť na bloku implementácie č. 3.1

```
Function UnetLayer
    ConvolutionalLayer(KernelSize, NumberOfFilters)(PreviousLayer)
    BatchNormalization
    Activation(ReLu)

    ConvolutionalLayer(KernelSize, NumberOfFilters)(PreviousLayer)
    BatchNormalization
    Activation(ReLu)

    MaxPooling
    Dropout
```

Listing 3.1: Ukážka implementácie Unet vrstvy



Obr. 3.1: Návrh celkového modelu.

3.1.2 Metóda generovania filtrov s využitím backpropagation

Prvý prístup, ktorý budeme používať a testovať, je backpropagation. Tento prístup môžeme zaradiť do skupiny automatickej inicializácie filtrov. Vybrali sme ho kvôli tomu, že tu nie je nutná špeciálna implementácia a ide o najviac používaný prístup v doméne konvolučných neurónových sietí. Spomínaný prístup je tiež veľmi efektívny a dáva sieti skoro úplnú kontrolu nad tým, ako budú filtre vyzerať. Čo sa týka nášho zadania, aby sme dodržali architektúru Unet neurónovej siete, použijeme presne takú istú vrstvu pre tento prístup ako v celej architektúre.

Pre tento prístup sme zvolili ako vhodné kombinácie veľkosť filtrov 3x3, 5x5 a 7x7 a počet filtrov 16, 32 a 64.

3.1.2.1 Implementácia backpropagation

Táto vrstva bude zložená z dvoch konvolučných vrstiev, kde jedna konvolučná vrstva bude skrytá. Ide o dávkovo normalizačnú (batchnormalization) vrstvu, aktivačnú funkciu ReLu, združovaciu (pooling) vrstvu a vynechávaciu (dropout) vrstvu, kvôli predídeniu pretrénovania. Výstupom funkcie BackpropagationLayer je vrstva, ktorá predstavuje túto metódu generovania filtrov. Implementáciu tejto metódy môžeme vidieť na bloku implementácie č. 3.2

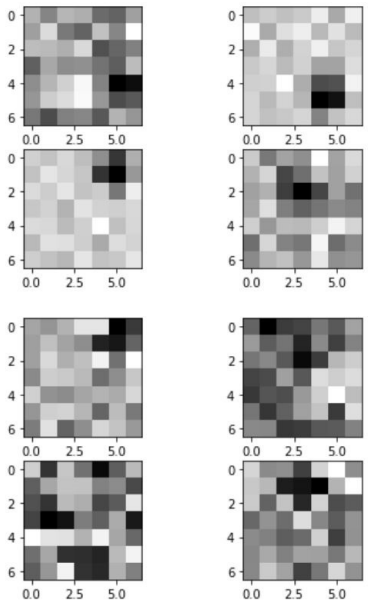
```
1 Function BackpropagationLayer
2     ConvolutionalLayer(KernelSize, NumberOfFilters)(PreviousLayer)
3     BatchNormalization
4     Activation(ReLu)
5
6     ConvolutionalLayer(KernelSize, NumberOfFilters)(PreviousLayer)
7     BatchNormalization
8     Activation(ReLu)
9
```

```
10      MaxPooling
11      Dropout
```

Listing 3.2: Ukážka prístupu backpropagation

3.1.2.2 Vizualizácia filtrov

Na obrázku č. 3.2 môžeme vidieť vizualizáciu niekoľkých filtrov tejto metódy. Vyberali sme také, ktoré najlepšie ukázali jednotlivé rozdiely.



Obr. 3.2: Vizualizácia filtrov pre prístup backpropagation.

3.1.3 Metóda generovania filtrov s využitím autoenkóder

Druhý prístup, ktorý budeme používať a testovať, je autoencoder. Je opísaný v časti 2.3.4. Keďže tento prístup, metóda, má za úlohu skopírovať a následne prekopírovať dáta, potrebuje viac ako jednu konvolučnú vrstvu a pre efektívnejšie vykonanie uvedenej metódy používame robustnejší model tejto vrstvy, ktorý je opísaný v časti 3.3.

Dôvod výberu tohto prístupu, metódy, je ten, že si myslíme, že pri jeho vykonávaní, teda skopírovaní a následnom prekopírovaní, by sa mohol naučiť veľmi dôležité informácie, vzory o dátach, ktoré automaticky aplikuje na dáta, a svoj výstup pošle ďalej. Teda dáta, ktoré budú obsahovať pozitívnu klasifikáciu, by po výstupe z tejto vrstvy mali vyzerať podobne, čo pomôže ďalej v klasifikácii. Tento prístup sa zaraďuje do triedy automatickej inicializácie filtrov.

Pre tento prístup sme zvolili ako vhodné kombinácie veľkosť filtrov 3x3, 5x5 a 7x7 a počet filtrov 16, 32 a 64.

3.1.3.1 Implementácia autoencoderu

Tento prístup obsahuje 4 konvolučné operácie, kde na konci siete spájame prvú vrstvu s poslednou vrstvou. Túto činnosť robíme pomocou spojovacej (Concatenate) vrstvy a nad-vzorkovacej (UpSampling) vrstvy. Výstupom je, ako u všetkých funkcií pre definovanie jednotlivých metód na generovanie filtrov, celá vrstva, ktorú môžeme ďalej použiť. Implementáciu tohto prístupu vidíme v bloku implementácie č. 3.3

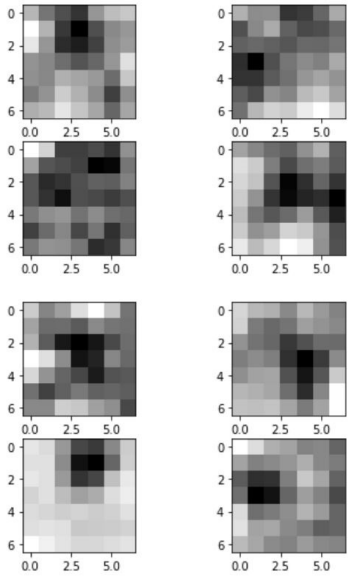
```
Function AutoencoderLayer
    ConvolutionalLayer(KernelSize, NumberOfFilters)
    Activation(ReLu)
    Dropout
    ConvolutionalLayer(KernelSize, NumberOfFilters)
    Activation(ReLu)
    Pooling

    ConvolutionalLayer(KernelSize, NumberOfFilters)
    Activation(ReLu)
    Dropout
    ConvolutionalLayer(KernelSize, NumberOfFilters)

```

```
14    Concatenate
15    UpSampling
```

Listing 3.3: Ukážka prístupu autoencoder

3.1.3.2 Vizualizácia filtrov

Na obrázku č. 3.3 môžeme vidieť vizualizáciu niekoľkých filtrov tejto metódy. Vyberali sme také, ktoré najlepšie ukázali jednotlivé rozdiely.



Obr. 3.3: Vizualizácia filtrov pre prístup autoencoder.

3.1.4 Metóda generovania filtrov s využitím Transfer learning

Ďalší prístup, ktorý používame a otestujeme, je transfer learning. Opis tohto prístupu sa nachádza v časti 2.3.5. Dôvod výberu tohto prístupu plynie z prác, ktoré sme opisovali v časti 2.4. Veľká časť novodobých riešení je založená práve na tomto princípe. Preto veríme, že tento prístup by mohol mať veľmi dobré

výsledky vďaka povahe, fungovaniu. Očakávame aj väčší nárast trénovacieho času pre veľkosť výslednej architektúry po pripojení modelu k našej architektúre (Unet). Tento prístup môžeme zaradiť znova do skupiny automatická inicializácia filtrov, či už kvôli tomu, že filtre už boli vytvorené podobným štýlom, alebo tomu, že určitú časť filtrov, ktoré používame v modeli, znova preučíme.

Prevzatým modelom, ktorý použijeme, budú už dobre známe modely, ktoré sú voľne dostupné aj s naučenými váhami, teda nebudeme trénovať pre tieto účely vlastné modely.

Vhodnými kombináciami pre tento prístup nie sú podobné varianty ako u predošlých dvoch prístupoch, veľkosti filtrov a počet filtrov, keďže tieto kombinácie a parametre sú presne definované v modeloch, ktoré použijeme. Jedinou vhodnou kombináciou bude počet filtrov, ktorým umožníme znova sa učiť, a typ modelu, teda model, ktorý používame. My sme sa rozhodli použiť modely, ktoré majú veľmi dobré úspešnosti čo sa týka klasifikácie v obrazových dátach. Použijeme modely ResNet152 a VGG16.

3.1.4.1 Implementácia transfer learningu

Pri tomto modeli, implementácia nebola nejako obzvlášť komplikovaná, pretože ako spomíname v predošlom bloku **??**, budeme používať modely, ktoré sú voľne dostupné a teda jedinou implementáciou bude vložiť ich. Následne definujeme ešte počet vrstiev, pri ktorých môže sieť upraviť naučené váhy, filtre. Pseudokód tejto implementácie môžeme vidieť v bloku implementácie č. 3.4

```
1 Function TransferLearningLayer
2   import ResNet152, VGG16
3     base model = ResNet152 or VGG16
4     base model.weights(trainable = 15)
```

Listing 3.4: Ukážka prístupu transfer learning

3.1.4.2 Vizualizácia filtrov

Čo sa týka vizualizácie filtrov, je zbytočná a tiež by nebola presná, lebo všetky vizualizácie, ktoré sme mohli doteraz vidieť, boli brané z prvých vrstiev. A keďže tento prístup v prvej vrstve má metódu tvorenia filtrov klasickým spôsobom, teda backpropagation, boli by filtre totožné s filtrami z metódy generovania filtrov pomocou backpropagation.

3.1.5 Metóda generovania filtrov s využitím Gáborových filtrov

Posledný prístup, ktorý chceme použiť a otestovať, je vytvorenie filtrov pomocou Gáborovej funkcie, pričom tie vložíme ako jediné filtre, ktoré táto vrstva bude mať a nebude ich môcť meniť.

Tento prístup sme vybrali kvôli tomu, že veríme, že Gáborové filtre budú mať dobré výsledky a budú vedieť lepšie nájsť a detegovať hrany, poprípade objekty v dátach, keďže sú generované prirodzenou funkciou a vyjadrujú určitú prirodzenosť používanú v prírode, ako spomíname aj v odstavci 2.2.4, kde taktiež opisujeme, ako sa tvoria a ako vyzerá Gáborová funkcia.

Kvôli skutočnosti, že tieto filtre sa budú musieť vytvoriť manuálne, očakávame, že aj používanie bude oveľa zdĺhavejšie a teda čas, počas ktorého sa bude trénovať sieť, bude rovnako ako v prístupe transfer learning oveľa väčší. Tento prístup sa zaraďuje do skupiny manuálnej inicializácie filtrov. Tieto filtre budeme pri každom modeli vytvárať nanovo a to pomocou dostupných funkcií. Očakávame, že implementácia tohto prístupu bude najkomplikovanejšia v porovnaní s ostatnými. Budeme musieť ešte otestovať aj vhodné vybratie parametrov pre Gáborovú funkciu. To urobíme tak, aby rôznorodosť týchto filtrov bola čo najväčšia.

V tomto prístupe budeme používať pre lepšie využitie väčšie filtre ako v iných prístupoch, pričom počet filtrov necháme rovnaký ako v iných kombináciách. Použijeme veľkosť filtrov 5x5, 7x7 a 9x9.

3.1.5.1 Implementácia Gáborových filtrov

Implementáciu tohto prístupu sme museli rozdeliť do dvoch krokov a to takých, že prvý krok definuje len konvolučnú operáciu, ktorej bolo zabránené učenie a upravovanie filtrov, a aktivačnej funkcii ReLu, túto implementáciu môžeme vidieť v bloku implementácie č. 3.5

```
Function GaborKernelsLayer
  ConvolutionalLayer(KernelSize, NumberOfFilters, Trainable =
   False)
    Activation(ReLu)
```

Listing 3.5: Ukážka prístupu gáborové filtre

Ďalším krokom je vytvorenie jednotlivých gáborových filtrov. Dôvodom tohto rozdelenia bolo, že keby sme chceli vložiť vygenerované filtre pomocou knižnice OpenCv, tak by sme museli vytvoriť taký tvar týchto filtrov, ktorý bolo nesmierne náročne vytvoriť. Zistili sme, že jednoduchšie bude upraviť váhy pre určitú vrstvu. Následne sa vyskytol problém s tým, že sieť si vytvorí váhy až potom, ako sa definuje model. A teda v druhom kroku sme si museli najprv získať daný tvar váh pre konvolučnú vrstvu a následne podľa tohto tvaru vytvoriť Gáborové filtre, teda dostačujúci počet týchto filtrov. Popis tohto algoritmu a celkovú implementáciu môžeme vidieť na bloku implementácie č. 3.6

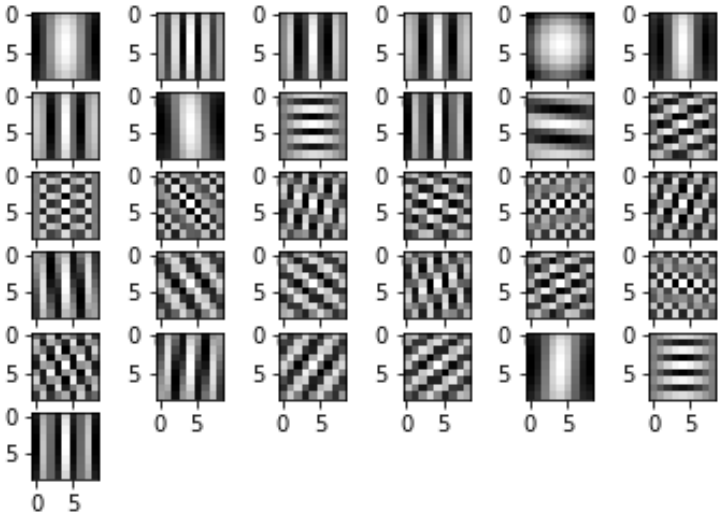
```
Function GaborKernelsInitialization
  shape = Model.Layer.weights
  params = theta, lambda, sigma, gamma, psi
  gabor kernels = cv2.getGaborKernel( params, kernel size  = shape
   .x shape.y )
```

```
5  Model.Layer.weights = gabor kernels
```

Listing 3.6: Ukážka prístupu gáborové filtre

3.1.5.2 Vizualizácia filtrov

Na obrázku č. 3.4 môžeme vidieť vizualizáciu všetkých filtrov vygenerovaných pomocou metódy GaborKernelsInitialization, pričom počet týchto fitlrov je 32 a jednotlivé parametre vstupujúce do funkcie sú theta je od 0 do pi po pi / 16, lambda je od 0 do pi po pi / 4, gamma je 5, sigma je 1, psi je 0.



Obr. 3.4: Vizualizácia filtrov pre prístup gáborové filtre.

Kapitola 4

Implementácia

4.1 Použitý hardvér a softvér

Túto prácu sme trénovali a testovali na stavanom počítači, ktorý obsahoval grafickú kartu NVIDIA GTX 1660 Ti, procesor AMD Ryzen 7 3700X, ktorý disponuje 8 jadrami a 32 GB RAM pamäťou. Minimálne sme používali Google Collab pre trénovanie, ktoré taktiež disponovalo grafickou kartou, ale s obmedzeným časom.

Celá implementačná časť bola naprogramovaná v programovacom jazyku Python, pričom ako distribúciu sme použili anacondu, ktorá ponúka predinštalované knižnice pre vedeckú a výskumnú prácu. Ako prostredie sme používali Jupyter notebook. Toto prostredie nám umožnilo vidieť okamžité výsledky, čo nám pomohlo nielen pri vývoji, ale aj testovaní. Ako hlavnú knižnicu a framework sme použili Keras a Tensorflow, kde podstatná časť je implementovaná pomocou knižnice Keras. Funkcie, ktoré knižnica neimplementovala, sme použili z frameworku Tensorflow. Definíciu Unet architektúry sme taktiež použili z Keras dokumentácie, jednotlivé prístupy sme implementovali do externého python script súboru, z

ktorého ich aj používame. Ako ďalšie knižnice, ktoré nám pomohli implementovať jednotlivé prístupy a preprocessing dát, sme použili:

1. OpenCv, ktorú sme použili hlavne pri implementácii Gáborových filtrov,
2. Numpy, ktorú sme použili na premenu vstupných dát do správnych rozmerov a typov,
3. Pylabna vizualizáciu grafov, pri histórii trénovania a pri vizualizácii filtrov (kernelov) jednotlivých prístupov,
4. H5py sme použili na načítanie a uloženie datasetov, pričom táto funkcia oproti klasickému načítaniu niekoľkonásobne znížila časovú a pamäťovú náročnosť,
5. Pillow sme použili pri vizualizácii dát,
6. využili sme aj tradičné knižnice na prácu so systémom OS, prácu s textom RE a iné.

4.2 Implementácia navrhnutého riešenia

4.2.1 Opis implementácie vytvárania modelov

Najdôležitejšie funkcie definované v skripte adaptive_model.py boli už vysvetlené v návrhovej časti. V tomto skripte implementujeme jednotlivé metódy na generovanie filtrov, tiež je tam implementovaná unet architektura spolu s jej špeciálnou konvolučnou vrstvou. Ostatné funkcie slúžia len pre sprehľadnenie finálneho kódu, ktorý bude opísaný v nasledujúcej časti. Tieto funkcie len vytvárajú model (funkcia define_model), definujú vstup (funkcia input_layer), definujú výstup (funkcia output_layer) a prechodný výstup z obrázka do finálnej vrstvy pre určite prístupy, teda pre transfer learning. Je špeciálny výstup po sieti a to Glo-

balAveragePooling, funkcia transfer_layer a pre ostatné prístupy to je vrstva s Flatten, ktorá premieňa obrázok na vektor, funkcia flatten_layer.

4.2.2 Opis implementácie hlavnej časti

V prvej časti tohto notebooku len importujeme knižnice, ktoré sú potrebné pre ďalšiu prácu. Následne umožňujeme, aby sa výpočty uskutočňovali na grafickej karte.

Ďalšie časti ako Load DAGM data a Load PCAM data slúžia len na preprocessing a načítanie trénovacích i testovacích dát a ich rozdelenie.

V časti Build model si definujeme základné globálne premenné pre trénovanie ako tvar vstupu (shape), počet epoch, teda počet iterácii, koľkokrát chceme model trénovať (epochs), počet predikovaných tried (num_classes), teda pozitívne a negatívne. A veľkosť trénovanej jednotky (batch_size).

V ďalších častiach definujeme jednotlivé prístupy, kde najprv vytvoríme vstup pre model, ďalej definujeme meno, podľa ktorého chceme, aby sa natrénovaný model uložil, potom definujeme špeciálnu vrstvu prístupu a jej parametre, pre jednoduchšie zvolenie parametrov máme tieto možnosti už napísané a sú len zakomentované. Následne vložíme Unet architektúru a definujeme výstupnú vrstvu. Všetky tieto kroky využívajú ako vstupný parameter predošlú vrstvu, čo nám umožňuje jednoducho ich spájať. Ku koncu použijeme funkciu define_model na vytvorenie modelu. Tieto kroky sú všeobecné pre každý prístup s tým rozdielom, že používame zakaždým inú špeciálnu vrstvu a to podľa prístupu.

Po všetkých týchto krokoch máme už model, ktorý môžeme trénovať a ďalej evaluovať a testovať. Nasleduje časť, v ktorej definujeme vlastné metriky pre overenie riešenia. Kompilácia modelu s callbacks, optimalizátor, používame opti-

malizátor Adam, loss funkciu categorical crossentropy a metriky úspešnosť, citlivosť, presnosť a iné, pričom tie sú využívané z frameworku Tensorflow. Po tomto kroku začneme trénovať s príslušnými parametrami a validačnými dátami. Potom uložíme model a vykonávame evaluáciu aj s reálnou predikciou a vizualizáciou grafov pomocou knižnice Pylab. Po týchto krokoch máme možnosť načítania špecifického modelu, vizualizáciu filtrov v prvej vrstve tohto modela a evaluácie na tomto modeli.

Kapitola 5

Popis experimentov a evaluácia

5.1 Použité dáta

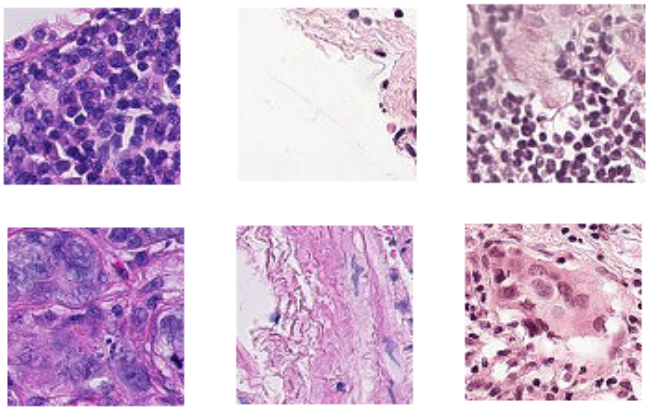
5.1.1 Dataset "PatchCamelyon" (PCAM)

[17]

Hlavným zdrojom dát pre naše riešenie bude dataset s menom PatchCamelyon, ktorý obsahuje 327 680 obrázkov histologických dát. Tieto obrázky sú vyrezy z pôvodného datasetu Camelyon16, ten obsahuje 400 H&E zafarbených WSI úsekov sentinelových lymfatických uzlín rozdelených na 270 podložných sklíčok s anotáciami pre trénovanie a 130 neoznačených podložných sklíčok na testovanie. Sklíčka sa získali a digitalizovali v 2 rôznych centrách. Jednotlivé obrazy sú vo veľkosti 96x96 pixelov. [16] (Obrázok č. 5.1)

5.1.1.1 Preprocessing

Načítanie a používanie datasetu PCAM nebolo veľmi náročné, pretože spolu s dátami bol poskytnutý aj skript pre načítanie tohto datasetu. Celý datasete je



Obr. 5.1: Ukážka datasetu PatchCamelyon16

rozdelený na trénovaciu, validačnú a testovaciu časť, každá časť bola rozdelená do viacerých súborov. Xové dáta, teda obrázky, sú uložené do HDF5 matice, čo uľahčuje použitie, šetrí pamäťové miesto a celkový čas potrebný pre načítanie týchto dát. Čo sa týka Yových dát, tie sú uložené taktiež v HDF5 súbore, pričom jedinou potrebnou úpravou bolo zmeniť tvar. To sme vykonali pomocou knižnice Numpy. Každá časť obsahuje tiež aj meta súbor, ktorý je vo formáte csv. V tomto súbore sa nachádzajú x a y koordináty, či sa tumor na týchto dátach nachádza, či je obrázok centrom tumoru a WSI.

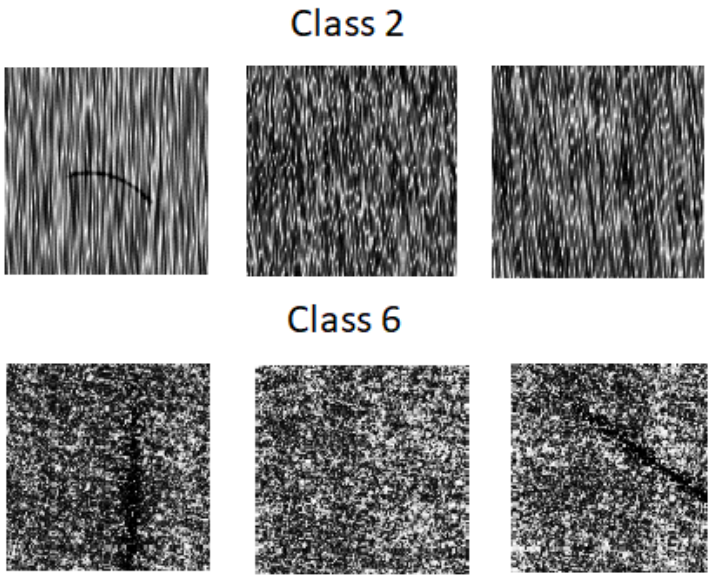
5.1.2 Dataset "Deutsche Arbeitsgemeinschaft für Mustererkennung" (DAGM)

[14]

Ako druhý dataset použijeme dataset menom Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM). Tento dataset obsahuje syntetický súbor údajov na detekciu defektov na textúrovaných povrchoch. Pôvodne bol vytvorený pre súťaž na sympóziu DAGM 2007.

My tento dataset použijeme pre kvalitnejšie porovnanie jednotlivých metód generovania filtrov. Keďže histologické dáta sú komplikované a sieť veľmi ťažko extrahováva vzory z nich. Pri týchto textúrových povrchov dokáže sieť oveľa jednoduchšie extrahovať vzory, ktoré neskôr použije na predikciu. Jednotlivé súbory sú rozdelené do 10 tried, pričom každá obsahuje rôzne frekvencie textúr. Pre naše účely budú stačiť 2 triedy. Ukážky týchto tried môžeme vidieť na obrázku č. 5.2 ako druhú triedu a ako šiestu triedu. [14]

Tieto obrázky sú vo veľkosti 512x512 pixelov.



Obr. 5.2: Ukážka datasetu DAGM class 2 a class 6.

5.1.2.1 Preprocessing

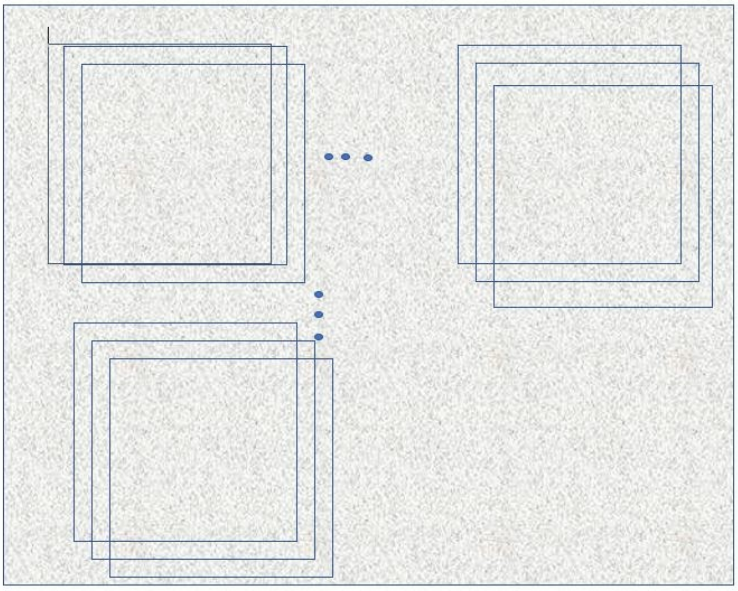
Práca s týmto datasetom bola náročnejšia než s predošlým, pretože v tomto datasete boli dáta rozdelené na holé obrázky, oanotované obrázky, kde boli len obrázky s pozitívnou triedou, pričom celý obrázok bol čiernou farbou a výskyt pozitívnej triedy bol označený bielou farbou. V textovom dokumente boli označené pozitívne a negatívne triedy s názvom obrázka.

Jednotlivé triedy datasetu, ako som spomínal v časti 5.1.2, boli rozdelené do trénovacej a testovacej časti. Keďže tento dataset obsahoval obrázky vo veľkosti 512x512 a náš pôvodný dataset PCAM obsahoval obrázky 96x96, museli sme upraviť obrázky, aby dosiahli veľkosť 96x96. Celý proces spracovania týchto obrázkov fungoval tak, že sme vytiahli len tie pozitívne obrázky a následne len s tými sme pracovali. Na predspracovanie, zmenšenie veľkosti obrázkov a zväčšenie počtu sme použili techniku s názvom kĺzavé okno.

5.1.2.2 Kĺzavé okno

Táto technika spočíva v tom, že máme obdĺžnik pevnej šírky a dĺžky. Následne len posúvame tento obdĺžnik doprava, ako štandardnú hodnotu sme použili 16 pixelov, teda veľkosť posunu bola 16 pixelov. Túto aktivitu sme robili dovtedy, pokiaľ sme neprišli na koniec riadku. Potom sme posúvali obdĺžnik dole, znova štandardná hodnota posúvania je 16 pixelov, a znova, až pokiaľ sme neprišli na koniec obrázku. Túto operáciu sme robili aj pre oanotované obrázky a obrázky s reálnymi dátami, pričom celý dataset sme ukladali do h5py matice, kam sme ukladali aj Yovú hodnotu a teda triedu, či je obrázok pozitívny alebo negatívny, či sa na obrázku objekt nachádza alebo nenachádza. Na obrázku č. 5.3 môžeme vidieť náčrt tohto algoritmu.

Vďaka skutočnosti, že vytváranie tohto datasetu sme zabezpečovali sami, vedeli sme ho jednoducho načítať a používať. Jedinou potrebnou úpravou bolo len následné rozdelenie testovacej časti na validačnú a testovaciu časť podľa vlastného výberu veľkosti.



Obr. 5.3: Náčrt kĺzavého okienka na pôvodnom obrázku.

5.2 Vyhodnocovanie jednotlivých metód generovania filtrov

5.2.1 Použité metriky

Ako hlavnú metriku sme vybrali účinnosť (accuracy), ktorá vyjadruje počet správnych výsledkov klasifikácie v pomere k počtu všetkých skúmaných vzoriek.

$$Eff = \frac{TP + TN}{TP + FP + TN + FN}, \tag{5.1}$$

Kde TP je skutočne pozitívne, FN je falošne negatívne, TF skutočne negatívne, FP je falošne pozitívne.

Ďalšou metrikou je presnosť (precision), ktorá vyjadruje pravdepodobnosť,

že vzorka je skutočne pozitívna, keď test vyšiel pozitívne.

$$PPH = \frac{TP}{TP + FP}, \tag{5.2}$$

Kde TP je skutočne pozitívne, FP je falošne pozitívne.

Ako tretiu metriku sme vybrali citlivosť (recall), ktorá je definovaná pomerom správne pozitívnych výsledkov k počtu hľadaných objektov skutočne sa vyskytujúcich v dátach [18].

$$TPR = \frac{TP}{TP + FN}, \tag{5.3}$$

Kde TP je skutočne pozitívne, FN je falošne negatívne.

Poslednou metrikou, ktorú používame, je f1 skóre, ktoré je kombináciou presnosti a citlivosti.

$$F1 = \frac{2PPH * TPR}{PPH + TPR}. \tag{5.4}$$

5.2.2 Výsledky na datasete PCAM

		Autoencoder			Backpropagation			GaborFilters		
		16	32	64	16	32	64	16	32	64
3x3	Úplnosť	0.7232	0.8212	0.8138	0.7891	0.8166	0.8596			
	F1 skóre	0.8055	0.8641	0.8594	0.8523	0.8683	0.8941			
5x5	Úplnosť	0.6474	0.7151	0.7843	0.7879	0.8055	0.8313	0.7979	0.7091	0.8227
	F1 skóre	0.7385	0.7812	0.8321	0.8497	0.8626	0.8772	0.8506	0.79329	0.8673
7x7	Úplnosť	0.7678	0.7045	0.7471	0.8167	0.8198	0.8070	0.5002	0.8160	0.5002
	F1 skóre	0.8213	0.7771	0.8062	0.8662	0.8684	0.8591	0.4990	0.8618	0.5001
9x9	Úplnosť							0.8140	0.4997	0.5002
	F1 skóre							0.8541	0.4997	0.4995

Obr. 5.4: Výsledky pre testovanie na PCAM datasete.

Toto testovanie bolo vykonané na testovacích dátach, ktoré sieť pri trénovaní vôbec nevidela. Trénovali sme v 10 cykloch a testovacie dáta boli v pomere 1 : 1, pričom veľkosť testovacej množiny bola okolo 320 tisíc obrázkov. Testovanie, ktoré sme vykonali pre jednotlivé metódy generovania filtrov, dávalo veľmi rozdielne výsledky, pričom v priemere dosahovali okolo 80%, samozrejme, boli aj také, ktoré dosiahli lepšie, ale aj horšie, a také, ktoré úlohu klasifikácie vôbec nezvládli.

Ako môžeme vidieť v tabuľke č. 5.4 , najlepšie výsledky dosiahla metóda generovania filtrov. pomocou Backpropagation a to úplnosť 85.96%, v kombinácií počet filtrov 64 a veľkosť filtrov 3x3. Dôvodom tohto výsledku bude asi, prispôsobivosť tohto prístupu, teda, že backpropagation, dáva sieti úplnú moc ako budu vzory, ktoré extrahuje a bude používať, vyzerať.

Čo sa týka najhoršieho výsledku z celého testovania na PCAM datasete, je to metóda generovania filtrov pomocou Gáborových filtrov, čo hovorí skôr o náhodnosti tejto metódy, pretože hodnoty z tohto prístupu sú viac menej náhodne, ak nie náhodne tak majú medzi sebou jednotlivé kombinácie najväčšie rozdiely. Poprípade to značí o tom, že sieť s týmto prístupom je dobrá pokiaľ sa jej dobre vygenerovali filtre, a teda táto metóda nedáva sieti dobré možnosti pre zlepšenie. A teda najhorší výsledok bol úplnosť 49,97%.

5.2.2.1 Transfer learning

Čo sa týka metódy na generovania filtrov pomocou Transfer learningu, táto metóda úplne zlyhala a všetky dáta označila ako negatívne, teda obrovská architektúra nedokázala vôbec klasifikovať nad takýmito dátami. Či už to bolo zapríčinené, že model ktorý sme použili nebol presne trénovaný nad takýmito dátami alebo že takáto veľkosť siete je nevhodná pre takúto klasifikaciu.

5.2.2.2 Backpropagation

Táto metóda dosahovala vo všetkých kombináciách dosť podobné výsledky, aj keď tieto výsledky boli rôzdielne na úrovni jednotiek. Môžme povedať, že táto metóda je najlepšia zo všetkých porovnávaných. To bude spôsobené vďaka tomu, že tento prístup, ako sme už spomínali, umožňuje sieti mať úplnú kontrolu. Čo sa týka jednotlivých kombinácií môžeme si všimnúť, že v tomto prístupe na výsledok jemne vplývajú nami zvolené parametre, pričom v tomto prípade sú to oba. Teda pokiaľ počet filtrov, ktoré sieť si mohla vytvoriť a upravovať, je väčší sieť dosahuje lepšie výsledky ako ked je menší. Pričom u veľkosti filtrov to je naopak aj ked to nie je nutné, čím je veľkosť menšia tým je úplnosť väčšia, čo môže byť spôsobené aj tým, že menšie filtre sú menej konkretnejšie a pomáhajú v tejto vrstve k lepšiemu spracovaniu. Aj keď na prekvapenie podľa tohto tvrdenia najhorší výsledok v tomto prístupe je s kombináciou počet filtrov 16 a veľkosť filtrov 5x5. V priemere tento prístup dosahuje okolo 82% čo je veľmi dobrý výsledok aj napriek zložitosti dát.

5.2.2.3 Autoenkodér

Metóda autoenkóder v tomto prípade dosiahla najhoršie výsledky, čo sa týka maximálnej úplnosti predikcie. Preto si myslíme, že táto metóda sa svojou podstatou pri rozdielnych dátach veľmi ťažko naučí klasifikovať triedy správne, hlavne pozitivne triedy. Pričom jednotlivé parametre nemajú skoro žiaden viditeľný vplyv na výsledky predikcie. Nehovoríme, že táto metóda na generovanie filtrov je zlá no žiadna zo zvolených kombinácií nedosiahla najlepšie výsledky no ani najhoršie. Najlepšia kombinácia dosiahla, v parametrov veľkosť filtrov 3x3 a počet filtrov 32 úplnosť 82,12% a najhoršia v parametrov veľkosť filtrov 5x5 a počet filtrov 16 úplnosť 64,74%, pričom tento výsledok je stále dobrý a sieť celkom zvládla úlohu klasifikacie.

5.2.2.4 Gáborové filtre

Čo sa týka tejto metódy, sieť v niektorých prípadoch vôbec nezvládla úlohu klasifikácie a všetky dáta označila ako negatívne. Toto sa stalo hlavne v prípadoch, keď kombinácia obsahovala veľkosť filtrov 7x7 a 9x9. V celkovej metóde na počte filtrov ani tak veľmi nezáležalo no na veľkosti filtrov výsledky siete boli veľmi viditeľné, čomu nasvedčujú aj prípady, kedy sieť zlyhala v úlohe klasifikácie. Najlepšie výsledky dosahovala metóda spolu s veľkosťou filtrov 5x5, tam dosiahlo najlepší výsledok čo sa týka tejto metódy, počtu filtrov 64 a to úplnosť 82.27%.

5.2.3 Výsledky na datasete DAGM

		Autoencoder			Backpropagation			GaborFilters		
		16	32	64	16	32	64	16	32	64
3x3	Úplnosť	0.9513	0.9568	0.9580	0.9149	0.9353	0.9486			
	F1 skóre	0.9565	0.9581	0.9607	0.9335	0.9468	0.9549			
5x5	Úplnosť	0.9417	0.9250	0.9581	0.9479	0.9353	0.9476	0.9514	0.9466	0.9492
	F1 skóre	0.9494	0.9344	0.9546	0.9494	0.9468	0.9525	0.9515	0.9511	0.9540
7x7	Úplnosť	0.9499	0.9588	0.9587	0.9344	0.9298	0.9520	0.899	0.9317	0.9339
	F1 skóre	0.9497	0.9580	0.9486	0.9433	0.9419	0.9545	0.9147	0.9351	0.9386
9x9	Úplnosť							0.9125	0.9267	0.9051
	F1 skóre							0.9245	0.9341	0.9254

Obr. 5.5: Výsledky pre testovanie na DAGM datasete.

Testovanie, ktoré sme vykonali na datasete DAGM, nám dalo oveľa lepšie výsledky ako na predošlom testovaní, pričom toto testovanie bolo vykonané na testovacích dátach, ktoré sieť pri trénovaní vôbec nevidela. Znova sme trénovali v 10 cykloch a testovacie dáta boli v pomere 2 : 1, kde negatívnych bolo okolo 18 tisíc, a pozitívnych bolo 9 tisíc.

Ako môžeme vidieť v tabuľke č. 5.5 ,najlepšie výsledky na týchto dátach dosiahla sieť s použitím metódy na generovanie filtrov pomocou Autoenkóderom a to úplnosť 95,88%, pričom od ostatných metód sa líši len v desatinných číslach. Dôvod

tohto výsledku je podľa nášho názoru ten, že vďaka charakteru tohto princípu a teda skopírovania a následného prekopírovania sieť dokázala obrázok tak upraviť, že následné pozitívne obrázky boli viac podobné a teda ďalšie spracovanie bolo jednoduchšie a jednotnejšie ukazovalo vzory.

Najhoršie výsledky dosiahla sieť s použitím metódy na generovanie filtrov pomocou Gáborových filtrov a to úplnosť 89,90%. Dôvod tohto výsledku môže byť ten, že vytvorené filtre boli buď príliš jednotné alebo sa vyskytovali mimo frekvenciu, ktorá sa nachádza na dátach a teda horšie rozoznáva pozitívne dáta.

5.2.3.1 Transfer learning

Táto metóda znova ako v predošlých dátach dosiahla nulové výsledky, teda všetky testovacie dáta označila ako negatívne.

5.2.3.2 Backpropagation

Pri tejto metóde výsledky jednotlivých kombinácií boli dosť podobné, no líšili sa v len v jednotkách alebo desatinných číslach. Úspešnosť tejto metódy je vynikajúca, kde najhoršie výsledky sú úplnosť 91,49%. Z uvedeného vyplýva, že klasická metóda, ktorá sa používa v konvolučných neurónových sieťach, je výborná. No napriek jej výbornej úspešnosti nie je najlepšia pre túto úlohu. Najlepším riešením je najväčší počet filtrov – 64 a najväčšia veľkosť filtrov, 7x7, kde úplnosť dosiahla 95,20%.

5.2.3.3 Autoenkodér

Táto metóda sa osvedčila ako najlepšia zo všetkým, pričom jej hodnoty sú skoro vo všetkých kombináciách totožné, len v desatinných miestach rozdielne a minimálne je tých, ktoré sa líšia v jednotkách. To, že je táto metóda najlepšia, hovorí aj o jej najhoršom výsledku z kombinácií a variácií, ktoré sme testovali, a to

92,5% . Najlepšia kombinácia je veľkosť filtrov 7x7 a počet filtrov 32. V porovnaní s predošlým datasetom, táto metóda predbehla všetky výsledky a ukázala sa ako najlepšia, čo pripisujeme hlavne jednote týchto dát, teda po vykonaní základnej funkcie autoenkóder vrstvy pri totožných dátach je táto úloha najlepšia. No pokiaľ máme dáta výrazne rozdielne, nedosahuje veľmi dobré výsledky oproti iným metódam.

5.2.3.4 Gáborové filtre

V tejto metóde medzi testovanými kombináciami môžeme vidieť už oveľa väčšie rozdiely a to hlavne v jednotkách, kde rozdiel medzi najlepšími a najhoršími výsledkami kombinácie je približne 6%. Najlepšie výsledky dosahuje kombinácia v zložení 7x7 veľkosť filtrov a 16 počet filtrov. Môžeme tiež vidieť, že veľkosť filtrov dosť vplýva na výsledky, keďže výsledky, ktoré majú rovnaké veľkosti filtrov, sú veľmi podobné a odlišujú sa len v desatinných číslach.

Kapitola 6

Záver

Spracovanie obrazových dát metódami umelej inteligencie je v dnešnej dobe jednou z najčastejších oblastí počítačovej vedy, ktorá ide dopredu a ktorá sa skúma. Hľadajú sa nové riešenia, objavujú sa nové poznatky. Našou úlohou bolo skúmať hlboké neurónové siete, konkrétne konvolučné neurónové siete a ich základné časti, teda metódy generovania filtrov. Skúmali sme rôzne typy metód generovania filtrov, pričom tieto typy vieme jednoducho rozdeliť do dvoch hlavných skupín a to na automatické generovanie filtrov a manuálne generovanie filtrov. Za automatické sme vybrali metódu generovania filtrov pomocou backpropagation, ktoré je klasickým spôsobom generovania filtrov. Čo sa týka použiteľnosti, používa sa skoro všade a je to štandardný spôsob. Ďalej sme skúmali generovanie filtrov pomocou autoenkóderu, ktorý svojou podstatou pomáha zjednodušovať vstupy a následne je jednoduchšie nachádzať v jeho výstupe vzory, poprípade určité objekty. Transfer learning sme vyhodnotili ako metódu, ktorá sa stáva v dnešnej dobe viac modernejšou a používanejšou. A poslednou metódou na generovanie filtrov sú filtre vytvorené pomocou Gáborovej funkcie. Táto metóda sa radí do kategórie manuálne generovanie filtrov.

Jednotlivé prístupy, metódy, sme sa rozhodli testovať a porovnávať na dátach, ktoré budú pre sieť náročné na rozpoznanie a taktiež na dátach, ktoré budú pre sieť jednoduché na rozpoznanie. Vďaka tomu sme chceli zistiť, či sú tieto prístupy rozdielne, či majú podobné výsledky, ktorá z týchto metód je lepšia a ktorá je horšia, a to aj na akom type dát. Ako dáta, ktoré budú pre sieť náročné, sme použili histologické dáta, konkrétnejšie v časti 5.1.1, ako dáta, ktoré budú pre sieť jednoduchšie, sme využili dáta rôznych textúrových povrchov, konkrétnejšie v časti 5.1.2.

Jednotlivé prístupy sme sa rozhodli otestovať viacerými parametrami, ktoré by mohli ovplyvniť výsledky. Pre metódy: Gáborové filtre, backpropagation, autoencoder, sme sa rozhodli meniť parametre, ktoré mohli ovplyvňovať výsledky jednotlivých neurónových sietí. Tieto parametre sú veľkosť filtrov a počet filtrov. Pri metóde transfer learning sme vybrali rôzne typy modelov a množstvo filtrov, ktoré mohla sieť upraviť.

Najlepším princípom pre klasifikáciu v histologických dátach sa ukázala metóda generovania filtrov pomocou Gáborových filtrov. Táto metóda je ale veľmi nestabilná, čo znamená, že pri zmene spomínaných parametrov veľmi kolíše úspešnosť tejto klasifikácie a to až do takej miery, že sieť v niektorých prípadoch zlyhávala, teda úspešnosť klasifikácie bola skoro nulová. Najstabilnejšie výsledky a zároveň najlepšie dosahuje metóda s využitím backpropagation, tá síce reagovala na zmenu parametrov, ale nie tak drasticky ako Gáborové filtre. Táto metóda dosahovala veľmi dobré výsledky aj v druhom datasete, ale nikdy nedosiahla najlepšie, čo odzrkadľuje jej prispôsobiteľnosť. Čo sa týka metódy generovania filtrov pomocou autoenkóder, táto metóda dosahovala tiež vhodné výsledky, no v porovnaní s ostatnými boli až 3. v poradí v oblasti histológie. Najlepšie výsledky podávala vtedy, pokiaľ dáta boli veľmi podobného charakteru, čo sme mohli vidieť v da-

taset DAGM 5.1.2. Tam dosiahla zo všetkých skúmaných najlepšie výsledky, a to pomerne stabilné, teda na úspešnosť zmena jednotlivých parametrov až tak nevplývala. Čo sa týka metódy s použitím transfer learningu, táto metóda v našich doménach, histológii a textúrových povrchov dosiahla katastrofálne výsledky a vôbec nedokázala klasifikovať a učiť sa. Preto si myslíme, že táto metóda je nepoužiteľná, pokiaľ model chceme používať už naučený, ale nie na podobnej doméne.

Literatúra

[1] A. Alekseev a A. Bobe. "GaborNet: Gabor filters with learnable parameters in deep convolutional url = http://arxiv.org/abs/1904.13204, neural networks". In: *CoRR* abs/1904.13204 (2019). arXiv: 1904.13204.

[2] T. Araújo et al. "Classification of breast cancer histology images using convolutional neural networks". In: *PloS one* 12.6 (2017), e0177544.

[3] A. L. Chandra. "Perceptron: The Artificial Neuron". In: (2018). URL: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d.

[4] G. Chartrand. "Deep Learning: A Primer for Radiologists". In: (2017). URL: https://pubs.rsna.org/doi/full/10.1148/rg.2017170077.

[5] I. Goodfellow, Y. Bengio a A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016, s. 167–523.

[6] A. K Jain a F. Farrokhnia. "Unsupervised texture segmentation using Gabor filters". In: *Pattern recognition* 24.12 (1991), s. 1167–1186.

[7] Ch. Kumar. "Artificial Intelligence: Definition, Types, Examples, Technologies". In: (2018). URL: https://medium.com/@chethankumargn/artificial-intelligence-definition-types-examples-technologies-962ea75c7b9b.

[8] S. Martin. "What is Transfer Learning?" In: (2019). URL: https://blogs.nvidia.com/blog/2019/02/07/what-is-transfer-learning/.

[9] D. Nelson. "What is Transfer Learning?" In: (2019). URL: https://www.unite.ai/what-is-transfer-learning/?gclid=Cj0KCQiAtrnuBRDXARIsABiN-7DFomxmBM3W6DPlFnHD5Oa_WBwx9iAbZEAMg1wbVLBXsGZI5Vz0qGQaAmRvEALw_wcB.

[10] A. Y. Ng. "Autoencoders". URL: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/.

[11] P. Návrat a et al. *Umelá inteligencia*. http://www2.fiit.stuba.sk/~kapustik/KnihaUI.html. STU Bratislava, 2002, 2006, 2015. (Malé centrum), 2015, s. 1–27. ISBN: 80-227-1645-6.

[12] S. J. Pan et al. "Transfer learning via dimensionality reduction." In: *AAAI*. Zv. 8. 2008, s. 677–682.

[13] A. Shah. "Through The Eyes of Gabor Filter". In: (2018). URL: https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97.

[14] Martin H. Skjelvareid. "DAGM 2007 competition dataset". In: (2007). URL: https://www.kaggle.com/mhskjelvareid/dagm-2007-competition-dataset-optical-inspection.

[15] Khan. S. U. et al. "A novel deep learning based framework for the detection and classification of breast cancer using transfer learning". In: *Pattern Recognition Letters* 125 (2019), s. 1 –6. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2019.03.022. URL: http://www.sciencedirect.com/science/article/pii/S0167865519301059.

[16] Bastiaan S. Veeling et al. *Rotation Equivariant CNNs for Digital Pathology*. 2018. arXiv: 1806.03962 [cs.CV].

[17] Bastiaan S Veeling et al. "Rotation Equivariant CNNs for Digital Pathology". In: (jún 2018). arXiv: 1806.03962 [cs.CV].

[18] E. Šikudová et al. *Počítačové videnie Detekcia a rozpoznávanie objektov*. Praha: Wikina,Praha, 2013, s. 38–252. ISBN: 978-80-87925-06-5. URL: http://vgg.fiit.stuba.sk/kniha/.

[19] N. Wahab, A. Khan a Y. S. Lee. "Transfer learning based deep CNN for segmentation and detection of mitoses in breast cancer histopathological images". In: *Microscopy* 68.3 (feb. 2019), s. 216–233. ISSN: 2050-5698. DOI: 10.1093/jmicro/dfz002. eprint: http://oup.prod.sis.lan/jmicro/article-pdf/68/3/216/28762457/dfz002.pdf. URL: https://doi.org/10.1093/jmicro/dfz002.