VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE A KLASIFIKACE POŠKOZENÍ OTISKU PRSTU S VYUŽITÍM NEURONOVÝCH SÍTÍ

DETECTION AND CLASSIFICATION OF DAMAGE IN FINGERPRINT IMAGES USING NEURAL NETS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. PETER VICAN**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ONDŘEJ KANICH, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Vican Peter, Bc.**
Program: Informační technologie
Obor: Bezpečnost informačních technologií
Název: **Detekce a klasifikace poškození otisku prstu s využitím neuronových sítí**
Detection and Classification of Damage in Fingerprint Images Using Neural Nets
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte literaturu týkající se rozpoznávání podle otisků prstů s důrazem na poškozené či nekvalitní snímky z důvodu onemocnění na prstu či dalších poškození. Seznamte se z možnostmi detekce a klasifikace s využitím neuronových sítí.
2. Vylepšete stávající řešení či navrhněte nový přístup, který posoudí, jaká část otisku je poškozená. V místech s poškozením pak algoritmus klasifikuje dané poškození (např. onemocnění, vliv přítlaku, nečistota senzoru, apod.) do minimálně tří tříd.
3. Implementujte navržený algoritmus z předchozího bodu.
4. Otestujte algoritmus na dostupných databázích otisků prstů (poškozených i nepoškozených). V případě vylepšování stávajícího řešení porovnejte oba přístupy.
5. Dosažené výsledky shrňte a diskutujte. Uveďte možná vylepšení a rozšíření Vašeho řešení.

Literatura:

- Drahanský, M.: *Hand-Based Biometrics: Methods and technology*, IET 2018, p. 430, ISBN 978-1-78561-224-4.
- Kanich, O.: *Research in Fingerprint Damage Simulations*, Doctoral thesis, FIT BUT in Brno, Brno, 2018, p. 148.
- Maltoni, D., Maio, D., Jain, A.K. and Prabhakar, S.: *Handbook of Fingerprint Recognition*. Springer, 2009, p. 512. ISBN 978-1-8488-2254-2.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Kanich Ondřej, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Cieľom diplomovej práce je naštudovať a navrhnúť experimentálne vylepšenie konvolučnej neurónovej siete pre detekciu ochorenia. Ďalším cieľom je rozšíriť klasifikátor o nový typ detekcie. Jedná sa o poškodenie pri snímaní odtlačku prsta tlakom. Experimentálne vylepšená konvolučná sieť je implementovaná pomocou PyTorch. Sieť detekuje aká časť odtlačku prsta je poškodená a túto časť vykreslí do odtlačku prsta. Pri trénovaní siete sú použité syntetické odtlačky prstov. K syntetickým odtlačkom prstom sú doplnené aj reálne odtlačky prstov.

Abstract

The aim of this diploma thesis is to study and design experimental improvement of the convolutional neural network for disease detection. Another goal is to extend the classifier with a new type of detection. he new type of detection is damage fingerprint by pressure. The experimentally improved convolutional network is implemented by PyTorch. The network detects which part of the fingerprint is damaged and draws this part into the fingerprint. Synthetic fingerprints are used when training the net. Real fingerprints are added to the synthetic fingerprints.

Kľúčové slová

odtlačky prstov, poškodenie a detekcia, konvolučné neurónové siete, dyshidróza, bradavice, PyTorch, Keras, python, metacentrum, tlak, psoriáza

Keywords

fingerprint, damage and detection, convolutional neural networks, dyshidrosis, warts, PyTorch, Keras, python, metacentrum, pressure, psoriasis

Citácia

VICAN, Peter. *Detekce a klasifikace poškození otisku prstu s využitím neuronových sítí.* Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ondřej Kanich, Ph.D.

Detekce a klasifikace poškození otisku prstu s využitím neuronových sítí

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Ondřeja Kanicha PhD. Ďalšie informácie mi poskytol Bc. Milan Šalko Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Peter Vican
19. mája 2021

Poďakovanie

Týmto by som sa chcel poďakovať pánovi Ing. Ondřejovi Kanichovi PhD, za odbornú pomoc pri písaní diplomovej práci. Zároveň chcem poďakovať kolegovi Bc. Milanovi Šalkovi za konzultácie a uvedenie do problematiky v jeho bakalárskej práci, na ktorú diplomová práca nadväzuje. Taktiež chcem poďakovať metacentru za poskytnutie výpočtového výkonu, kde sa konvolučné neurónové siete trénovali a testovali.

Obsah

Kapitola 1

Úvod

Slovné spojenie, odtlačky prstov, sa spája s rôznymi seriálmi a filmami z kriminalistiky. Avšak odtlačky prstov sú už dnes súčasťou našich životov. Odtlačky prstov sa využívajú na overenie totožnosti používateľa v mobilných zariadeniach, prenosných počítačoch, vstupov do budovy a mnoho ďalšieho. Vďaka tomuto overeniu sa človek môže prihlásiť do svojho zariadenia a aplikácií pre správu citlivých údajov ako je napríklad bankovníctvo alebo informácie o zdravotnom poistení. Ďalší príklad využitia odtlačku prsta sa nachádza v osobných dokladoch, ktorými sú napríklad občianske preukazy a pasy. V dokladoch je uložená šablóna, ktorá slúži na overenie totožnosti pri hraničných kontrolách alebo kontrole cestujúcich na letiskách.

Problém pri overení totožnosti človeka nastáva v prípade, ak odtlačok prsta je nejakým spôsobom znečistený, vlhký alebo človek trpí kožným ochorením. Vďaka týmto poškodeniam môže dôjst k falošnej identifikácii. Falošná identifikácia nesprávne určí osobu ako neoprávnenú alebo v najhoršom prípade priradí inú identitu danej osobe. Falošnú identifikáciu by mohlo zmierniť varovanie systému, že odtlačok je nejakým spôsobom poškodený.

Diplomová práca je zameraná na trénovanie konvolučnej neurónovej siete pre detekciu a klasifikáciu poškodenia odtlačkov prstov, ktoré by mohlo riešiť vyššie zmienené varovanie systému. Poškodeniami odtlačku prsta sú nielen kožné ochorenia ako napríklad bradavice a dyshidrózy, ale aj poškodenie odtlačku prstu pri snímaní. Táto práca nadväzuje na bakalársku prácu [66], ktorá vznikla vo výskumnej skupine STRaDe, kde sa detekciou a klasifikáciou zaoberajú dlhší čas.

Neurónové siete sa v dnešnej dobe viac a viac integrujú do každodenného života, kde nám pomáhajú pri riadení dopravnej situácii na križovatke, detekcii rôznych predmetov v obraze, spracovanie prirodzeného jazyka a v neposlednom rade na pomoc pri hľadaní vakcíny na rôzne ochorenia. Diplomová práca sa skladá z hlavného cieľa, ktorý má za úlohu experimentálne vylepšiť stávajúcu konvolučnú neurónovú sieť. Druhým cieľom ak sa podarí je rozšírený klasifikátor.

Diplomová práca je rozdelená do siedmich kapitol. V kapitole 2 sú vysvetlené pojmy zo spracovania odtlačku prsta, ktorými sú napríklad odtlačok prsta a jeho syntetická varianta, choroby a iné. Pojmy ako konvolučná neurónová sieť, stratová funkcia, aktivačná funkcia a ostatné základné fundamenty neurónových sietí sa nachádzajú v kapitole 3. Kapitola 4 popisuje návrh experimetálneho vylepšenia a návrh nového klasifikátora. Kapitola 5 obsahuje implementáciu riešenia. Posledná kapitola 6 sa zameriava na dosiahnuté výsledky a testovanie. V tejto kapitole sa taktiež nachádza porovnanie implementácie z bakalárskej práce a implementácie z kapitoly 5.

Kapitola 2

Spracovanie odtlačkov prstov

V priebehu dňa prebehne niekoľko miliónov spracovaní odtlačku prsta na rôznych zariadeniach ako sú smartfóny, prenosné počítače, zariadenia pre povolenie vstupu do budovy a mnoho ďalšieho. Niekedy sa môže stať, že spracovanie odtlačku prsta neprebehne korektne, čo môže mať za následok mnoho faktorov.

Kapitola sa bude práve zaoberať týmito vplyvmi neúspešného spracovania odtlačkov prstov, ktorými sú napríklad rôzne choroby alebo poškodenia pri snímaní. Ale skôr než to bude rozobraté v kapitole dopodrobna, je potrebné si vysvetliť základný pojem biometria v kapitole 2.1, koža a jej funkcia v kapitole 2.2, odtlačok prsta v 2.3, získavanie odtlačku prsta v 2.4, spracovanie odtlačku prsta v kapitole 2.5, syntetický odtlačok v kapitole 2.6, poškodenie odtlačku prsta pri snímaní v kapitole 2.7 a nakoniec choroby v kapitole 2.8.

2.1 Biometria

Digitalizácia je v súčasnej dobe na veľkom vzostupe a v najbližších rokoch bude exponenciálne rásť [46]. Z tohto dôvodu bude potreba lepšieho zabezpečenia dát, prístupu do chytrého bankovníctva, prístupu do budovy, prístup do auta a mnoho ďalšieho. Dnešné zabezpečenie **heslom** (to, čo človek pozná) môže byť zabudnuté alebo stratené na poznámkovom bloku. Ďalším riešením je **prístupová karta alebo usb kľúč** (niečo, čo človek má fyzicky), ale aj tie môžu byť odcudzené alebo stratené. Heslá, karty a usb kľúče sa dajú ľahko zdieľať, takže neposkytujú nespochybniteľnosť. [19] [45]

Problémy s nespochybniteľnosťou rieši **biometrické rozpoznávanie** (niečim, čím človek je) alebo jednoducho **biometria** *(z gréckeho slova bios = život a metron = merať)*, ktorá využíva anatomické (*odtlačky prstov, tvár, dúhovka*) a behaviorálne (*chôdza, reč, písmo*) charakteristiky. Tieto charakteristiky sa nazývajú biometrické charakteristiky na automatické rozpoznávanie jednotlivcov. Hlavnou výhodou týchto vlastností je, že človek ich nemôže zabudnúť, stratiť alebo byť jednoducho zneužité inou osobou. Mnoho týchto vlastností môže veľa napovedať o zdravotnom stave jedinca a tým pádom, treba k tomu pristupovať zodpovedne a bezpečne. Pred tým, než hociktorú vlastnosť zoberieme do úvahy je nutné overiť, či daná vlastnosť spĺňa nasledujúcich deväť vlastností: [19] [41] [66]

- *Univerzálnosť* - každý by mal mať túto vlastnosť.
- *Jedinečnosť* - žiadne dve osoby by nemali mať rovnakú vlastnosť.
- *Trvalosť* - vlastnosť by sa nemala meniť počas života.

- *Merateľnosť* - vlastnosť by sa mala dať ľahko získať.
- *Výkon* - vlastnosť by nemala byť menená ani zmenená.
- *Prijateľnosť* - ochota spoločnosti prijať zachytenie tejto vlastnosti.
- *Falšovateľnosť* - ako náročné je sfalšovať danú vlastnosť.
- *Cena* - koľko stojí zavedenie takéhoto biometrického systému s danou vlastnosťou.
- *Prevádzka* - koľko stojí prevádzka takéhoto biometrického systému s danou vlastnosťou.

Pre biometriu existuje pár dôležitých konceptov, medzi ktoré patrí inter a intratriedna premenlivosť. Intertriedna premenlivosť zobrazuje veľkosť rozdielu vlastností medzi rozdielnymi triedami (osobami). Intratriedna premenlivosť udáva veľkosť rozdielu medzi vlastnosťami v jednej triede (jedinec). [40] [66]

Vďaka tomuto konceptu môžeme použiť ako biometrickú charakteristiku napríklad dúhovku, tvár alebo odtlačky prstov. Posledná zmienená charakteristika je vďaka intertriednej premenlivosti schopná identifikovať aj jednovaječné dvojičky. Pri intratriednej premenlivosti sa dokáže určiť, z ktorej ruky a z ktorého prsta bol odtlačok nasnímaný alebo odobratý.

2.2 Popis kože a jej funkcia

Koža pokrýva povrch tela a je zároveň najväčším orgánom v tele. Jej plocha je okolo 2 m^2 a váži v priemere 4 kg. Koža plní viacero funkcií ako napríklad vylučovanie odpadových látok alebo termoregulácia. Najvýznamnejšia funkcia kože je ochrana pred vonkajšími vplyvmi. Koža obsahuje receptory tepla, tlaku a bolesti, pomocou ktorých sme v kontakte s vonkajším svetom. Koža sa skladá z troch častí: [20] [68]

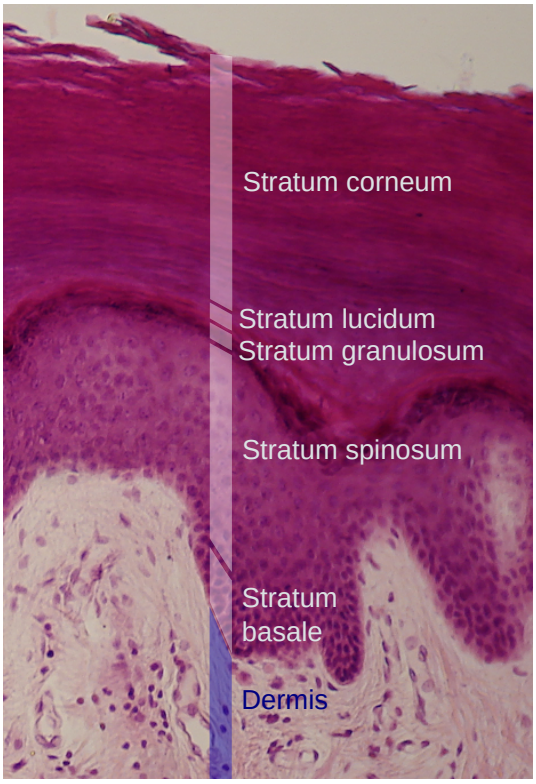
- *Pokožka* (epidermis) - vrchná časť pokožky.
- *Zamša* (dermis) - stredná vrstva, ktorá podopiera pokožku a zároveň však s pokožkou pevne spojená.
- *Podkožie* (hypodermis) - najspodnejšia vrstva, voľné spojivové tkanivo, ktoré obsahuje dostatok tuku.

Vedný obor, ktorý sa zaoberá a popisuje štruktúru kože, choroby a jej príčiny, sa volá *dermatológia*.

2.2.1 Epidermis

Najvrchnejšia časť kože môže mať hrúbku od 0,5 mm až po 1,5 mm. Najtenšia je v oblasti očných viečok a najhrubšia na dlaniach a chodidlách [4]. Skladá sa z mnoho vrstiev, tesne zabalených buniek v dlaždicovom usporiadaní. Pri detailnejšom pozorovaní je vidieť päť hlavných vrstiev, ktoré sú zobrazené na obrázku 2.1. V najspodnejšej vrstve si je možné všimnúť výbežky dermy. [4] [20] [68]

Najspodnejšia vrstva je **stratum basale**. Obsahuje vrstvu kubicko-cylindrických buniek, ktoré sú pripojené k bazálnej membráne. Takéto spojenie oddeľuje epidermu od dermy, a kvôli tomu je membrána zvlnená. Jej úlohou je, tlačiť staršie bunky smerom nahor, ktoré sa pri posunoch splošťujú až nakoniec odumrú a vylúčia sa.



Obrázok 2.1: Štruktúra epidermy. Zdroj: [36].

Nad vrstvou **stratum basale** sa nachádza **stratum spinosum**, ktorá vďaka keratínovým filamentom zaisťuje mechanickú odolnosť kože. Obsahuje tiež Langerhansove bunky, ktoré nás ochraňujú pred infekciou.

Stredná vrstva, ktorá má názov **stratum granulosum**, skladajúca sa zo živých buniek keratinocytov. Tieto bunky sú mierne sploštené. Bunky produkujú veľké množstvo keratínu, ktorý sa ukladá do zŕn, podľa čoho bola táto vrstva pomenovaná. Hrubne bunková membrána a bunky vylučujú epiderálne lipidy, ktoré sú zodpovedné za bariérové vlastnosti pokožky.

Stratum lucidum je tenká vrstva epidermis obsahujúca dve až tri vrstvy buniek. Tvorí dôležitú bariéru, ale existuje len na dlaniach a chodidlách.

Najvrchnejšou vrstvou je **stratum corneum**. Skladá sa z niekoľkých vrstviev sploštených buniek, ktoré sa nazývajú korneocyty. Rozlišujeme tenký a hrubý typ podľa hrúbky tejto vrstvy. Najhrubší a najsilnejší typ je na miestach s vysokým tlakom, napríklad na dlaniach a chodidlách. Rozdeľuje sa na dve časti:

- *Stratum conjunctum* - spodná kompaktná vrstva,
- *Stratum disjunctum* - horná odlupujúca vrstva.

2.2.2 Dermis

Je stredná vrstva medzi *epidermou* a *hyperdermou*. Skláda sa z elastického tkaniva, retikulárneho vlákna a hlavnou zložkou je kolagén. Kolagén tvorí 70 % obsahu a slúži na odolnosť voči napätiu a ťahu. Vo vrstve sa nachádzajú nervové zakončenia (Ruffiniho, Meissnerove

telieska), krvné vlásočnice. Pre potreby daktyloskopie nás zaujímajú potné žľazy, ktoré vyúsťujú na vyvýšeninách odtlačkov prstov, (papilárne línie). Dermu rozdeľujeme: [72]

- *Stratum papillare* - vystupuje z epidermy v podobe papíl. Tieto papily sa na konci kože prstov a dlaní prejavujú ako papilárne línie. Papilárne línie slúžia k identifikácii osoby. Vrstva obsahuje riedke kolágenové väzivo a množstvo rôznych nervových zakončení.
- *Stratum disjunctum* - skladá sa zo silnej vrstvy elastínového a kolagénového väziva a nachádza sa v nej menej buniek.

2.2.3 Hyperdermis

Najvnútornejšia vrstva. Väčšinou je tvorená tukom a spojivovým tkanivom. Hrúbka vrstvy sa líši, kde sa nachádza. Najhrubšia je v oblasti zadku, chodidlách a dlaniach. Plní dôležitú funkciu, termoreguláciu.

2.3 Odtlačok prsta

Odtlačky prstov sú podľa dochovaných čínskych textov používané viac než 3000 rokov [69]. Veľké množstvo ľudí verí, že odtlačky prstov sú unikátne, ale v skutočnosti to nie je matematicky potvrdené. Rozhodnutie o unikátnosti je založené na empirickom pozorovaní po niekoľko storočí. Prvý, ktorý popísal empirické pozorovanie, bol sir Francis Galton. Popísal, že dva odtlačky prstov sú rovnaké s pravdepodobnosťou 1 ku 64 miliardám. [37] [42]

Odtlačky prstov sa uplatnili v praxi kvôli stálosti, cene a unikátnosti. Najvýznamnejším uplatnením je v kriminalistike, odomykanie mobilných zariadení a iné. Rastie akceptovateľnosť medzi ľudmi, ale na druhú stranu stále zostáva nedôvera z uloženia biometrických údajov a potencionálnym zneužitím pomocou falzifikátu odtlačku prstov.

V nasledujúcej podkapitole bude vysvetlené, čo sú papilárne línie, a prečo sú dôležité. Následne bude popísaná klasifikácia odtlačkov prstov podľa tried a markanty.

2.3.1 Papilárne línie

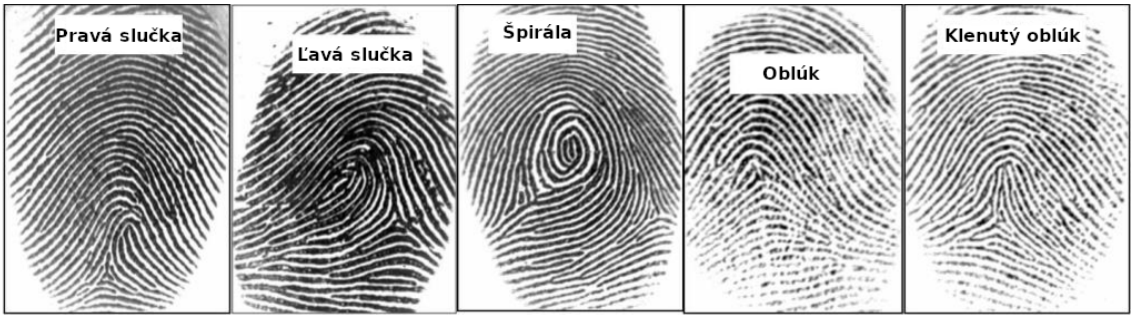
Papilárne línie, ako bolo spomenuté v podkapitole 2.2.2, sú výstupky z epidermy, ktoré sú tvorené z papíl a vrásnia kožu. Ich výskyt je na rukách a dlaniach. Ich výška môže byť 0,1 až 0,4 mm a dosahujú hrúbky 0,2 mm až 0,5 mm. Vyvinuli sa pre lepšie uchopenie predmetov, aby sa nekĺzali v rukách. Na ich vyústení sú potné žľazy, ktoré je vidieť pri kvalitnejšom obrázku ako malé bodky. Papilárne línie na konci brušiek prstov tvoria odtlačky. [40] [41] [66]

Smer a tvar papilárnych línií čiastočne vychádza z genetických línií, vďaka čomu sa nejedná o náhodné vzory. Podobnosť papilárnych línií je medzi rodičmi a deťmi ako napríklad počet papilárnych línií, tvar alebo hĺbka. Najväčšia podobnosť papilárnych línií je u identických dvojčiat, vyplýva to podľa [45], ale aj pri nich nie sú odtlačky prstov identické. Tieto malé zmeny tvoria jedinečnosť odtlačkov prstov. Papilárne línie sa formujú približne vo štvrtom mesiaci tehotenstva. Plne vyvinuté sú v siedmom mesiaci tehotenstva a počas života sa nemenia. Jediným možným spôsobom ako zmeniť papilárne línie je odstránenie zárodočnej vrstvy. [40] [41] [45] [66]

2.3.2 Klasifikácia odtlačkov

Porovnať dva odtlačky nie je jednoduchá úloha, ako to môže vyzerať na prvý pohľad. Najväčším priekopníkom v klasifikácii bol Francis Galton, ktorý v diele [29], popísal tri hlavné triedy pre rozpoznanie odtlačku prsta. Konkrétne špirála, oblúk a slučka.

S podobným trojtriednym rozdelením prišiel aj Edward Henry. Henryho klasifikácia bola použitá pri tvorbe IAFIS *(Integrated Automated Fingerprint Identification System)* FBI (*Federal Bureau of Investigation*) v roku 1999 [24]. V roku 2017 FBI vydal ďalší systém NGI *(Next Generation Identification)*, ktorý sa nelimituje iba na odtlačky prstov [25]. Dnes sa väčšinou používa rozdelenie do 5 tried. Z obrázku 2.2 je vidieť, že táto klasifikácia sa skladá z pravej slučky, ľavej slučky, špirály, oblúku a klenutého oblúku.



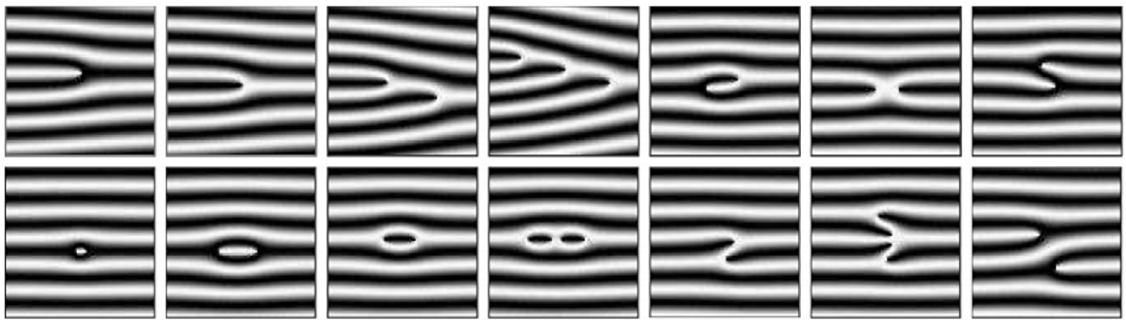
Obrázok 2.2: Rozšírené triedy klasifikácie. Upravené zo zdroja: [22].

Najfrekventovanejším výskytom sú slučky. Je ich možné nájsť v (65,5 %) všetkých odtlačkov prstov. Druhým najrozšírenejším sú špirály, ktoré tvoria (27,9 %). Oblúky sú najunikátnejšie, tvoria výskyt len okolo (6,6 %). [19] [40] [42]

2.3.3 Markanty

Pre jednoznačné určenie odtlačku prsta je rozdelenie do tried nedostatočné. Charakteristiky, ktoré sú dostatočné a jednoznačné pre určenie každého prsta sú **markanty** (*minutiae*). Markant [19] je špeciálny druh informácie, tvorený lokálnymi útvarmi, ktoré tvoria papilárne línie. Markanty sa nazývajú aj Galtonove detaily [45] pomenované na počesť sira Francisa Galtona. Galton popísal viac ako 100 markantov [45].

Na obrázku 2.3 je vidieť niekoľko základných typov markantov. Zľava doprava je to *ukončenie, jednoduchá vidlička/rozdvojenie, dvojitá vidlička, trojitá vidlička, hák, kríženie,*



Obrázok 2.3: Základné typy markantov. Zdroj: [42].

bočný kontakt, bod, interval, jednoduchá slučka, dvojitá slučka, jednoduchý most, dvojitý most a priesečná línia.

Pri automatickom/počítačovom rozpoznávaní odtlačku prsta sa prevažne využívajú dva typy markantov a to ukončenie a vidlička. Tieto markanty sa používajú hlavne z dvoch dôvodov. Prvým dôvodom je menšia náročnosť na rozpoznanie ako u ostatných markantov. Druhým dôvodom je, že sa s nimi dajú poskladať ostatné typy markantov [19] [38]

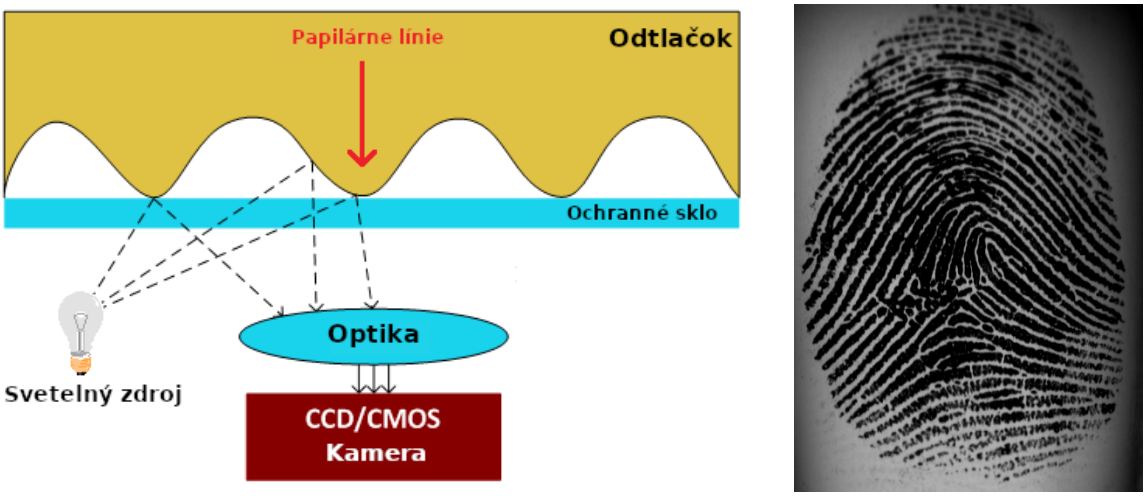
2.4 Získavanie odtlačkov prstov

Pre potreby verifikácie a validácie je potrebné nejakým spôsobom získať odtlačky prstov do digitálnej podoby [40]. Existujú rôzne spôsoby pri získavaní, napríklad oskenovaný, latentný odtlačok prsta. Avšak existuje pohodlnejší spôsob, ako získať odtlačok prsta do počítača, ktorým je priame nasnímanie odtlačku prsta pomocou senzora pripojeného USB (*universal serial bus*) portom [40].

2.4.1 Optická technológia

Je jednou z najstarších technológií snímačov odtlačkov prstov. Je založená na pomerne jednoduchej technológii, ktorú je možné vidieť na obrázku 2.4.

Prst je položený na ochranné sklo, kde sa výstupky papilárnych línií dotýkajú skla a údolia sú v diaľke. Zo svetelného zdroja, najčastejšie LED, dopadá paprsok na povrch prstu a je odrazený od výstupkov a absorbovaný údoliami. Odrazené paprsky potom sníma CCD/CMOS kamera. Najväčšou výhodou snímania pomocou tejto technológie je odolnosť voči teplotným výkyvom a je možné fungovanie v 3D. Nevýhodou tejto technológie je vysoká citlivosť na špinavé prsty. [19] [21] [40]

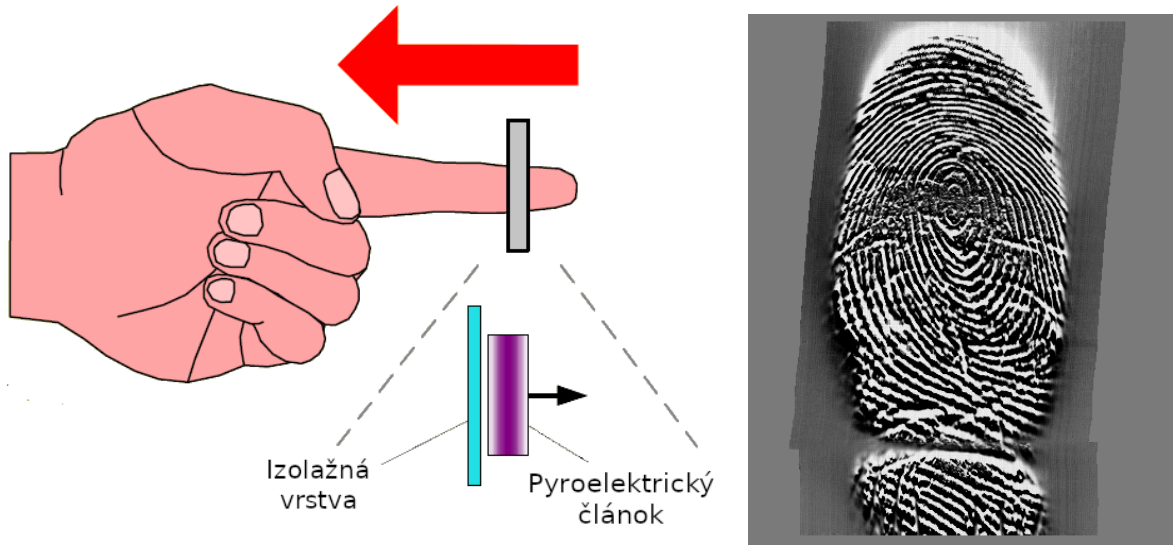


Obrázok 2.4: Optická technológia, upravené ([40]) a odtlačok prsta z nej ([21]).

2.4.2 Tepelná technológia

Technológia je založená na princípe tepelného žiarenia. Výstupky majú vyššie tepelné žiarenie než údolia. Snímanie odtlačku prsta sa deje prejdením prsta po pyroelektrickom článku. Článok generuje prúd podľa teploty, ktorú je možné zmerať. [21] [40] Kvôli rýchlo sa vyrovnávajúcej teplote sa používa senzor na princípe potiahnutia prsta. Volajú sa prieťahové

(sweep) senzory a nájdeme ich napríklad v prenosných počítačoch alebo mobiloch. Princíp technológií je zobrazený na obrázku 2.5. Výhodou týchto senzorov je vysoká odolnosť voči elektrostatickému náboju.



Obrázok 2.5: Tepelná technológia, upravené ([40]) a odtlačok prsta z nej ([21]).

2.5 Spracovanie odtlačku

Po získaní odtlačku prstu metódami, ktoré sme si popísali vyššie, je potrebné vysvetliť proces rozpoznávania odtlačku prstu. Proces rozpoznávania je zobrazený na obrázku 2.6. Z obrázka je vidieť päť fáz. Prvou fázou je digitálny odtačok prsta. Digitálny odtlačok sa vytvorí snímaním odtlačku prsta. Väčšina novodobých senzorov využíva pri snímaní odtlačku prsta detekciu živosti (*prezentation attack detection*). Druhou fázou je vylepšenie obrazu. Pre vylepšenie obrazu sa využíva napríklad Gáborov filter alebo filtrovanie vo frekvečnej doméne (FFT $\longrightarrow$ aplikácia filtra $\longrightarrow$ IFFT). Filtre, ktoré sa používajú pri filtrovaní vo frekvečnej doméne sú napríklad dolná priepust, Butterworth filter alebo Ikonomopoulos filter. Treťou fázou je binarizácia. Binarizácia sa obvykle prevádza pomocou nejakej prahovej metódy. Príkladom môže byť adaptívne prahovanie alebo priemerné prahové hodnoty. Na konci tohto procesu sú výstupky čierne a údolia biele. Následne prebieha proces detekcie markantov, kde sa pre tento účel používajú výstupky. V tomto kroku sú výstupky zmenšené na šírku jedného pixelu. Pri stenšení musí platiť, že papilárne línie neubudnú žiadnym smerom. Ak by ubudli, tak by mohol byť problém pri určení polohy markantov. Posledná fáza je detekcia a extrakcia markantov. [19] [41]

Obrázok 2.6: Proces spracovania odtlačkov prstov. Upravené, zdroj: [41].

2.6 Syntetický odtlačok

Pri vývoji a testovaní algoritmov pre detekciu a klasifikáciu poškodenia odtlačku prsta sú potrebné trénovacie dáta. Pri získavaní dát sa naráža na problém, že databázy s odtlačkami sú veľmi malé, a tým sa znižuje kvalita trénovania a testovania. Pri takto malých databázach vzniká problém, že natrénovaný algoritmus je priamo závislý na trénovacích dátach a znižuje sa miera zistenia ochorenia pri rôznych snímaných prostrediach alebo techník. Pre kvalitné natrénovanie je odhadom potreba tisíc až desaťtisíc odtlačkov prsta pre jeden typ ochorenia. Získavanie takto veľkých databáz je náročné na čas a peniaze.

Tento problém by vyriešila veľká zdieľaná databáza s rôznymi poškodeniami a chorobami. Taktiež by obsahovala rôzne snímané prostredia a techniky. Pri zdieľanej databáze vzniká ďalší problém a to je, že v praxi sa tieto databázy nemôžu zdieľať. Jednou z príčin, prečo to nie je možné, je ochrana osôb a problém pri zneužití. Druhou príčinou je, že odtlačok prsta naznačuje mnoho o zdravotnom stave človeka. Tieto informácie môžu byť tiež istým spôsobom zneužité.

Pre problém zdĺhavého zbierania vlastnej databázy a problém zdieľania databázy sa môže využiť alternatíva vo forme syntetického odtlačku prstu. Jedná sa o vytvorenie odtlačku prsta podobného tomu ľudskému zo šablóny. Zo šablóny sa dajú vytvoriť rôzne variácie a rôzne odtlačky prstov simulujúce napríklad ochorenie. [45]

Pre vytvorenie syntetického odtlačku prsta sa využívajú rôzne nástroje ako napríklad SFinGe, Anguli, SyFDaS. Všetky zmienené nástroje majú grafické používateľské rozhranie. V nasledujúcich podkapitolách budú nástroje v skratke predstavené.

2.6.1 Generátor SFinGe

Generátor SFinGe *(Synthetic Fingerprint Generator)* je nástroj pre vytvorenie väčšej databázy odtlačkov prsta. Nástroj bol vyvíjaný v roku 2004 a pri vývoji sa podieľal Raffaele Cappelli z Boloňskej univerzity. [9] Jedná sa o najznámejší a zároveň najstarší nástroj pre generovanie syntetického odtlačku [40].

Najnovšia verzia nástroju SFinGe je 5.0. Vo verzii sú vylepšené algoritmy pre vytvorenie databáz a modelov. V tejto verzii pribudol aj nový parameter, pomocou ktorého sa dokáže kontrolovať pravdepodobnosť vytvorenia odtlačku prsta pri nízkej kvalite. [65]

Nástroj obsahuje až 10 krokov, ktoré pomáhajú k vytvoreniu čo najautentickejšieho odtlačku prstu. Prvé štyri kroky sa musia vykonať spoločne, aby sa vytvoril hlavný odtlačok prsta *(master fingerprint)*. Prvým krokom je vytvorene vzhľadu odtlačku prstu. Predvoleným nastavením je elipsoidný tvar. Druhým krokom sa určuje rozmiestnenie jadier a delt a ich počet. Vďaka tomuto sa určuje smerovanie papilárnych línií. V tomto kroku sa určuje do ktorej triedy odtlačok prsta patrí. Triedy boli popísané v podkapitole 2.3.2. Tretím krokom sa vytvorí mapa hustoty. Vo štvrtom kroku sa vytvorí hlavný odtlačok prsta, z ktorého sa v ďalších krokoch pomocou faktorov vytvoria rôzne odtlačky prsta s rôznymi tvarmi, ryhmi v odtlačku, simulovanie rôznych snímaných prostredí, pridanie šumu. [8] [9]

2.6.2 Generátor Anguli

Generátor Anguli je ďalší nástroj pre generovanie syntetického odtlačku prsta. Vytvorený bol v Indickom inštitúte vied. Generátor je inšpirovaný generátorom SFinGe a zároveň využíva aj jeho algoritmy. Jedná sa o voľne dostupný generátor, ktorý je napísaný v jazyku C++. Názov Anguli je z hidnského slova *Anguli*, čo v preklade znamená *Prst*. [16]

Pre jeho voľnú dostupnosť sa často používa vo vedeckých štúdiách. Jeho obrovskou výhodou je, že dokáže za necelé štyri dni vygenerovať až 1 milión odtlačkov prstov. Pri generovaní sa vie zadať presný počet vygenerovaných odtlačkov, pridanie šumu, uhol otočenia a do akej triedy má patriť. Umožňuje paralerné generovanie a pred generovaním nastaviť počet jadier. [16]

2.6.3 Generátor SyFDaS

Nástroj SyFDaS bol vyvinutý Ondřejom Kanichom v spolupráci s výskumnou skupinou STRaDe na FIT VUT v rámci dizertačnej práce [41]. Nástroj sa skladá z generátora odtlačku prsta a simulácie poškodenia odtlačku prsta. Nástroj vychádza z diplomovej práce [10].

Používateľ si môže nastaviť rôzne parametre ako napríklad typ poškodenia, typ senzora. Aktuálna aplikácia má na výber prieťahový, dotykový senzor alebo bezdotykový senzor. V nástroji sa dajú simulovať poškodenia ako napríklad tlak a vlhkosť, deformácia pokožky, úzky senzor, poškodený a špinavý senzor [41]

2.7 Poškodenie odtlačkov pri snímaní

Poškodenie odtlačku prsta pri snímaní môže spôsobiť problémy pri identifikácii osôb. Poškodenie môže byť spôsobené technickým vybavením ako aj rôznymi fyzikálnymi a biologickými zmenami. Týmito zmenami môžu byť napríklad suchý prst, vlhký prst, prst od oleja, vlas medzi prstom a senzorom alebo rôzne prítlaky na senzor.

2.7.1 Poškodenie a nečistoty senzora

Poškodenie pri snímaní odtlačku prsta vplyvom poškodenia senzora môže mať viacero technických príčin. Technické príčiny ako napríklad odchádzajúce podsvietenie pod snímačom senzora, prasknuté sklo znižujú kvalitu odtlačku prsta alebo pridávajú šum do snímku. Ďalšia technická príčina môže byť porucha pri posielaní signálu zo senzora do počítača. Táto príčina môže spôsobiť menšiu snímaciu plochu alebo sa odtlačok nenasníma vôbec.

Ďalším poškodením môžu byť rôzne znečistenia senzoru. Týmito nečistotami môžu byť napríklad mastné sklo, prach, piesok alebo padnutý vlas medzi prstom a snímačom.

2.7.2 Tlak a vlhkosť

Možným poškodením pri snímaní je tlak. Tlak sa môže považovať za úmyselné poškodenie odtlačku prsta pri snímaní. Pokiaľ sa na senzor pritlačí, tak to spôsobuje zosilenie papilárnych línií. Papilárne línie pri veľmi veľkom tlaku zosilia natoľko, že ich nebude možné rozpoznať. [41]

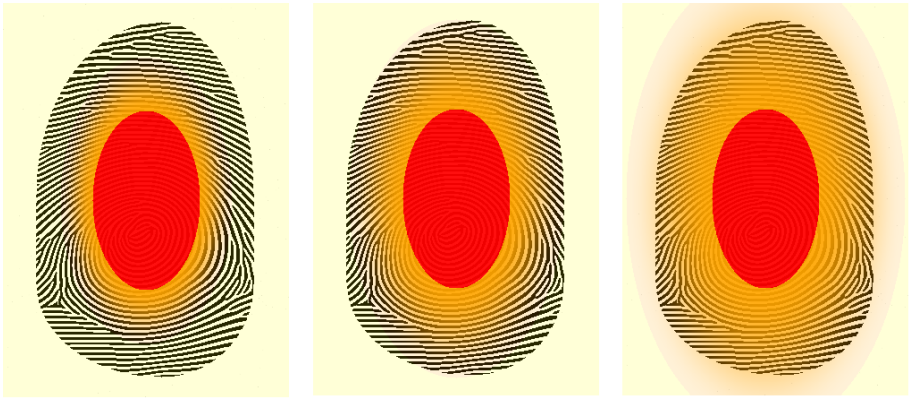
Vlhkosť podobne ako tlak upravuje rôznu hrúbku a kontrast papilárnych línií [41]. Na obrázku 2.7 je vidieť syntetický odtlačok prsta, na ktorom boli prevedené tieto poškodenia.

2.7.3 Skreslenie odtlačku

Poškodenie odtlačku prsta pomocou skreslenia obvykle nie je úmyselné. Tento typ poškodenia sa vyskytuje tak často, že je problém nasnímať odtlačok bez tohto poškodenia. Príčinou poškodenia je deformácia kôže a neortogonálneho tlaku prsta na snímač. Je to zapríčinené veľkou elasticitou kože. Problémom nie je samotné poškodenie, ktoré zvyčajne nenarobí

Obrázok 2.7: Rôzny tlak a vlhkosť. Zdroj: Vygenerovaný pomocou SFinGe.

veľké škody pri zmene polohy markantov alebo pri zmene vzdialenosti markantov. Problémom sú algoritmy na rozpoznávanie odtlačkov prstov. Algoritmy rozpoznávajú odtlačky prstov na základne svojich hlavných rozpoznávacích prvkov. Ak jeden z hlavných rozpoznávacích prvkov je poloha markantov, tak toto poškodenie môže spôsobit problémy pri identifikácii osôb. [41] Prípady takýchto skreslení popisuje obrázok 2.8, na ktorom sú vyznačené rôzne oblasti skreslenia.



Obrázok 2.8: Rôzne oblasti skreslenia. Zdroj: [41].

2.8 Choroby ovplyvňujúce odtlačky prstov

Choroby kože ovplyvňujú zdravotný stav daného človeka, ale zároveň aj obmedzuje pracovať s niektorými technológiami. V tejto podkapitole budú popísané niektoré choroby, ktoré znemožňujú krátkodobo alebo dlhodobo bezproblémové použitie odtlačku prsta ako biometriu.

2.8.1 Bradavice

Táto podkapitola hodne vecí preberá z bakalárskej práce [66]. Bradavice *(verrucae vulgaris)* sú najrozšírenejším kožným ochorením. Ochorenie sa šíri prenosom z človeka na človeka, ale aj autoinokuláciou. Ochorenie je spôsobené ľudskými papilomavírusmi (skratka *HPV* z anglického *human papillomavirus*) typu 1, 21 4 a 7. Do organizmu sa dostanú najčastejšie

cez kožné poranenia. Inkubačná doba je niekoľko týždňov alebo mesiacov. Príznakom tohto chorenia je zhrubnutie vrchnej vrstvy epidermy (stratum corneum) [39] [73].

Ochorenie sa najčastejšie vyskytuje u detí, ale vyskytuje sa aj u dospelých. Vyskytuje sa predovšetkým na rukách a nohách, pozdĺž nechtov a pod nimi, na ústach. Môžu sa objaviť kdekoľvek na celom tele. Obvykle sú to zrohovatené tvrdé výrastky šedohnedej farby. Ich tvar je kruh a môžu byť veľké od 5 mm až po 1 cm. Niekedy sa môže stať, že okolo najstaršej bradavice „materskej“ sa vyskytne druhotný výsev menších „dcérskych“ bradavíc. Na bradaviciach sa môže vyskytnúť aj iný znak, ktorým sú tmavé bodky. Jedná sa o trombotizované kapiláry. [39]

Liečba bradavíc nie je nijak zvlášť jednoduchá. Na odstránenie bradavice sa používa sneh CO_2, tekutý dusík, kyselina salicová a iné leptadlá. Často po odstránení materskej bradavice zmiznú aj tie dcérske. [39]

Obrázok 2.9: Bradavice na odtlačkoch prstov. Zdroj: Interná databáza skupiny STRaDe.

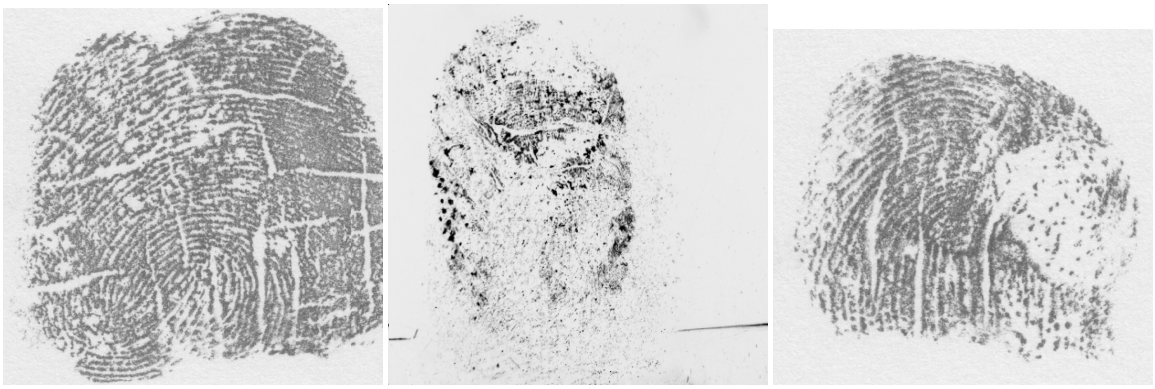
Na obrázku 2.9 je možné vidieť prejavy bradavice na odtlačkoch prstov. Prejavujú sa ako biele, skoro okrúhle plochy s čiernymi bodkami. Bodky vo vnútri bradavici sa nemusia vôbec prejaviť pri menších bradaviciach. Poškodenie touto chorobou nie je až tak závažné. Na zvyšku odtlačku prsta sú zreteľne vidieť papilárne línie. [35]

2.8.2 Dyshidróza

Pompholyx je občasným pomenovaním pre ochorenie dyshidróza (*dyshidrotic dermatitis*) [20]. Vo všeobecnosti nie je zhoda, čo zapríčiňuje ochorenie, ale udáva sa, že to pravdepodobne môže byť s kontaktom nejakého alergénu. Ochorenie nie je prenosné z človeka na človeka.

Ochorenie postihuje ruky a nohy, boky prstov a dlaní, kde sa objavujú svrbiace pľuzgieriky. Pľuzgieriky sa môžu spájať do väčších búl a následne prasknúť. Prasknuté miesta vlhnú a vysychajú. Pri vyschnutí kože tieto miesta praskajú a objavujú sa hlboké, bolestivé praskliny. Jeden zo sprievodných javov môže byť zápal týchto pľuzgierikov. U niektorých ľudí sa vyskytuje ochorenie permanentne alebo sa vracia každé leto po dobu troch až štyroch týždňov. Problém sa môže riešiť nepoužívaním kozmetiky, na ktorú je človek alergický. [20]

Z obrázka 2.10 je vidieť prejavy dyshidrózy na odtlačku prsta. Typickým poškodením sú hrubé čiary, ktoré pretínajú papilárne línie. Čiary sa môžu zobrazovať cez celý odtlačok

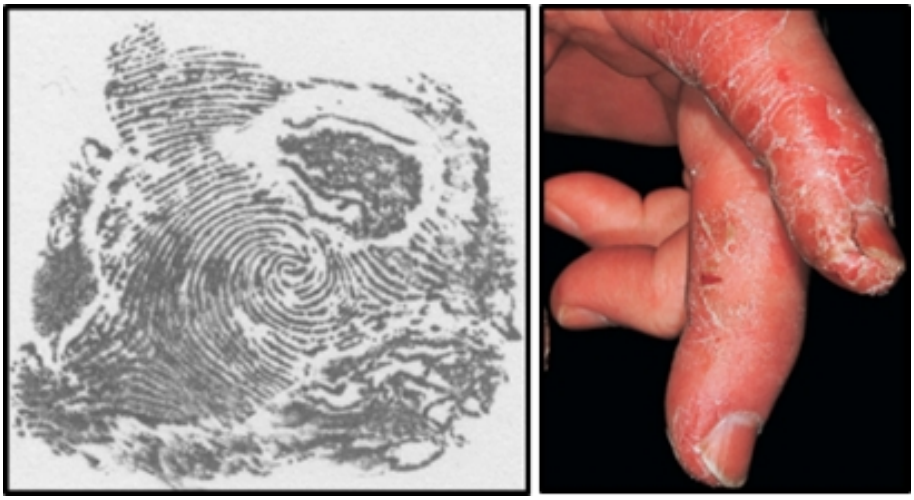


Obrázok 2.10: Dyshidróza na odtlačkoch prstov. Zdroj: Interná databáza skupiny STRaDe.

prsta a majú rôzne smery a dĺžku. Ľahší priebeh tohto ochorenia nám umožňuje v celku dobre vidieť papilárne línie, kde pri stredných až ťažkých prípadoch zaberajú cez celú oblasť odtlačku prstu a sú nepoužiteľné. [35]

2.8.3 Psoriáza

Psoriáza je ochorenie, ktoré postihuje hlavne kožu, ale 10-20 % pacientov trpí aj kĺbovým postihnutím. Toto podružné ochorenie sa nazýva psoriatická artritída. Jedná sa o chronické



Obrázok 2.11: Ochorenie psoriáza. Zdroj: [2].

ochorenie kože, ktoré je často nerozoznateľné od závažnej formy ekzému ruky. Na koži sa psoriáza prejavuje ako ložiská hnedočervenej kože so striebornými olupujúcimi sa šupinami. Príčina psoriázy nie je doposiaľ známa, ale radí sa do autoimunitných ochorení. Nemôže dôjsť prenosu z človeka na človeka. Obrázok 2.11 ilustruje psoriázu na končekoch prstoch a jej prejavy na odtlačkoch prstov. [48]

Kapitola 3

Neurónové siete

Umelé neurónové siete (skratka *ANNs* z anglického *Artificial Neural Networks*) obsahujú súbor algoritmov, ktoré fungujú na základe princípu mozgu savcov. Základnou výpočtovou jednotkou je neurón. V ľudskom mozgu je približne 10^{11} neurónov, ktoré tvoria zhruba 10^{15} prepojení. Neuronóvá sieť je považovaná za základný zdroj inteligencie, ktorá zahrňuje vnímanie, poznanie a učenie pre človeka ako aj ostatné živé tvory. [57]

Prepojenie medzi neurónmi môže byť tvorené dvomi spôsobmi. Prvý spôsob je pomocou prepojenia [33] a druhý je pomocou synapsií. Synapsie tvoria zložitejšie štruktúry a tie vytvarajú sieť pre spracovanie dát. [15]

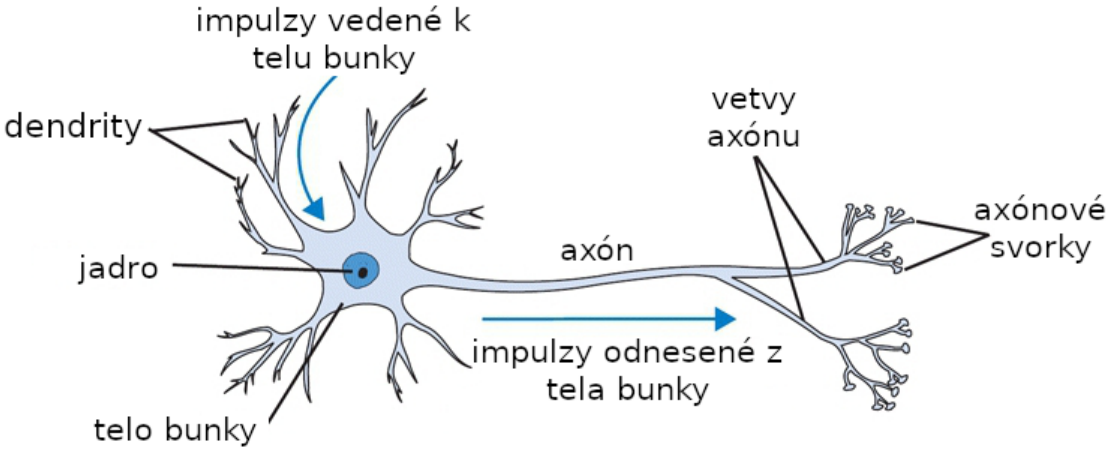
Na podobnom princípe ako bolo zmienené vyššie fungujú umelé neurónové siete. Umelé neurónové siete je možné využiť pre rôzne úlohy spracovania dát. Môžeme využiť buď učenie s učiteľom, kde sa môže učiť rozpoznávať štruktúry v súboroch tréningových dát a zovšeobecniť, čo sa naučil. Druhá možnosť je využiť učenie bez učiteľa, ktoré môžeme použiť na analýzu veľkých súborov dimenzionálnych dát. Učenie bez učiteľa nám pomáha riešiť problém dôležitosti funkcií. V praxi to znamená, že nám pomáha určiť, ktoré funkcie sú dôležité pre riešenie daného problému. [49]

Neurónové siete sa často využívaju v praxi ako napríklad detekovanie a vymazanie spamového emailu, komunikácia s chatbotom, detekcia objektu v obraze alebo na predpoveď prosperity firmy na trhu [27].

V nasledujúcich podkapitolách bude rozobraný detailnejšie rozdiel medzi biologickým a umelým neurónom. Budú vysvetlené pojmy ako aktivačná funkcia, loss funkcia, viacvrstvé neurónové siete, konvolučné siete a učenie neurónových sietí. V poslednej podkapitole budú spomenuté predtrénované modely.

3.1 Biologický neurón

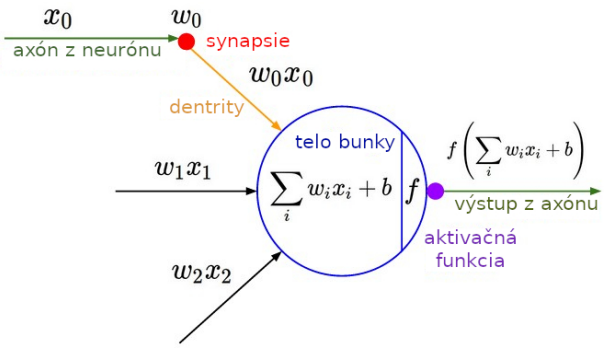
Biologický neurón sa skladá z bunkového tela, dendritov a axónov, ktoré sú vidieť na obrázku 3.1. Telo bunky sa stará o látkové premeny. Pomocou dendrít prijímajú vstupné vzruchy a pomocou axónu prenášajú vzruchy neurónov do ďalších. Axón neurónu je dlhý a tenký, vyznačujúci sa vysokým odporom a kapacitou. Preto axón môže byť modelovaný ako odporovo kapacitné prenosové vedenie. [33] Jednotlivé neuróny komunikujú medzi sebou pomocou synapsií. Najbežnejší typ neurónu obsahuje jeden axón a zopár rozvetvených dendritov, ktoré končia synapsiami. Funkciou neurónu je získavanie, prenos, spracovanie a ukladanie informácií. [50]



Obrázok 3.1: Biologický neurón. Upravené, zdroj: [13].

3.2 Umelý neurón

Na obrázku 3.2 je zobrazený matematický model neurónu, ktorý sa používa v ANN. Z obrázkov 3.1 a 3.2 je vidieť, že algoritmy neurónovej siete používajú výrazne zjednodušené modely neurónu. Avšak základný princíp zostáva rovnaký. Signály, vo výpočtovom modeli, ktoré sa pohybujú pozdĺž axónu sa nazývajú vstupy (napríklad x_0, x_1). Tieto signály interagujú s dendritami z druhého neurónu na základe synaptickej sily v danej synapsii. Spojenie s danou pevnosťou sa nazýva váha (napríklad w_0, w_1) a operácia spôsobená interakciou so signálom má charakter násobenia (napríklad $w_0 x_0$). Myšlienka opätovného pripojenia uvedená vyššie je predstavovaná učiacimi váhami, ktoré riadia silu vplyvu jedného neurónu na iný. Následne sa signály dostanú do tela bunky, kde sa všetky sčítajú spoločne s bias b. Aktivačná funkcia f sa použije na konečný výsledok, ktorý rozhodne, či neurón by mal byť ďalej poslaný alebo nie. [15] [21]



Obrázok 3.2: Matematický model. Upravené, zdroj: [14].

3.3 Aktivačná funkcia

Aktivačná funkcia je matematická funkcia, ktorá umožňuje rozhodnúť, či je neurón aktivovaný alebo nie. Na obrázku 3.2 je v tele bunky popísaná formula aktivačnej funkcie. Táto formula je prepísaná v nasledujúcej rovnici 3.1.

$$a = \sum_{i=1} w_i x_i + b \tag{3.1}$$

Z formuly vyplýva $a \in \mathbb{R}$, čo značí neexistujúcu informáciu o hraničných hodnotách. Kvôli tejto neexistujúcej informácii, nie je možné rozhodnúť či daný neurón má byť aktivovaný alebo nie. Z tohto dôvodu aktivačná funkcia *f(a)* slúži na transformáciu do rozsahu, ktorý určuje, či sa daný neurón aktivuje a propaguje sa ďalej alebo nie. V tejto podkapitole sú všetky informácie čerpané z [21] [33] [66].

Prahová funkcia nazývaná ako Heavisidová funkcia. Zo vzťahu 3.2 je vidieť, že pre kladné hodnoty vrátane nuly vracia funkcia 1 a pre záporné hodnoty vracia 0.

$$\phi(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases} \tag{3.2}$$

Niektorí používajú inú definíciu pre prahovú funkciu ako je vidieť na vzťahu 3.3. Táto definícia funkcie hovorí, že pre kladné hodnoty vracia 1, pre $x = 0$ vráti $1/2$ a pre zápornú hodnotu vracia 0. [71]

$$\phi(n) = \begin{cases} 1 & n > 0 \\ \frac{1}{2} & \text{n} = 0 \\ 0 & n < 0 \end{cases} \tag{3.3}$$

Kroková funkcia je znázornená v rovnici 3.4. Táto funkcia vráti 1 v prípade, že vstupná hodnota n je väčšia ako určený prah vo funkcii. V ostatných prípadov funkcia vráti 0.

$$\phi(n) = \begin{cases} 1 & n > prah \\ 0 & inak \end{cases} \tag{3.4}$$

Problém krokovej funkcie rieši **lineárna funkcia**. Rozdiel medzi krokovou a lineárnou funkciou je, že funkcia násobí vstup konštantou c. Vďaka tomuto vie produkovať viac hodnôt. Zároveň je to problém, pretože nie je možné použiť metódu spätného šírenia pri učení. Tomuto typu aktivačnej funkcii sa treba vyhnúť aby neboli problémy s učením.

$$\phi(n) = cn \tag{3.5}$$

Veľmi známou funkciou je **sigmoid** funkcia. Definuje sa ako neklesajúca funkcia, ktorá transformuje vstupné hodnoty do rozsahu (0;1). Vďaka tomuto vykazuje dobrú rovnováhu medzi lineárnym a nelineárnym správaním. Využíva sa pri binárnych problémoch. Sigmoid funkciu môžeme definovať následovne:

$$\phi(n) = \frac{1}{1 + e^{-n}} \tag{3.6}$$

Podobnou funkciou k sigmoid funkcii je **tanh** funkcia. U tejto funkcii sa vstupné hodnoty transformujú do hodnôt (-1;1). Funkcia má strmejší sklon než sigmoid funkcia.

$$\phi(n) = \frac{2}{1 + e^{-2n}} - 1 \tag{3.7}$$

Pri klasifikácií do viacerých tried sa často používa **softmax** funkcia. Funkcia, ktorá určí každej triede na výstup pravdepodobnosť, do ktorej triedy patrí. Súčet týchto pravdepodobností sa rovná jednej. Triedu určí na základe najvyššej pravdepodobnosti. Funkcia môže byť zapísaná nasledujúcim matematickým vzťahom, kde J je počet pravdepodobností:

$$\phi(n)_i = \frac{e^{n_i}}{\sum_{j=1}^{J} e^{n_j}} \tag{3.8}$$

ReLu funkcia je jedna z najčastejších funkcií, ktorá sa používa. Výstupom z funkcie je vstup, ak vstupná hodnota je väčšia nanajvýš rovná 1. Pre záporné hodnoty vracia 0. Jej zápis je:

$$\phi(n) = max(0, n) \tag{3.9}$$

Jej hlavnou výhodou je jednoduchosť. Nevýhoda funkcie je, že ak je problém v záporných hodnotách, tak vracia 0. Pre tieto problémy sa využíva **leaky ReLu** verzia. U tejto funkcii je dôležité, že sa pridáva faktor c, ktorý sa znižuje pri zápornom rozsahu. Upravenú funkciu je možné zapísať nasledujúcim vzťahom:

$$\phi(n) = \begin{cases} n & pre \geq 0 \\ cn & pre < 0 \end{cases} \tag{3.10}$$

3.4 Loss funkcia

Stratová funkcia (anglicky *loss function*) jedná sa o objektívnu funkciu, ktorou je možné ohodnotiť riešenie. Takouto objektívnou funkciou môže byť napríklad vyhodnotenie množiny váh. Snahou je maximalizovať alebo minimalizovať objektívnu funkciu. Znamená to, že hľadáme také riešenie, ktoré má minimálne alebo maximálne skóre. [5]. Funkcia popisuje vzdialenosť medzi aktuálnym odhadom siete a skutočným riešením. Pri strojovom učení je snaha dosiahnuť minimum stratovej funkcie. Jej dôležitou úlohou je, čo najpresnejšie interpretovať všetky vstupné parametre tak, aby na výstupe z funkcie sa odrazilo reálne zlepšenie modelu. Výsledná hodnota stratovej funkcie je číslo, ktoré zohľadní všetky vstupné parametre siete. [66] Pri výbere zlej funkcie, môže byť aj skvelý model nepoužiteľný. V nasledujúcich podkapitolách bude rozobraný problém klasifikácie a stručne popísané jednotlivé stratové funkcie, ktoré sa dajú použiť.

3.4.1 Problém klasifikácie

Problém klasifikácie pri stratovej funkcii rozdeľujeme na tri druhy. [7] [66]

- **Binárna klasifikácia** je problém, kde úlohou je zaradiť vzorku do jednej z dvoch tried. Koncepcia predikcie pravdepodobnosti je, že vzorka patrí do triedy jedna. Príkladom je trieda, ktorej sa priradí celočíselná hodnota jedna. Zatiaľ čo druhej triede je priradená nula. Pre výstupný uzol je vhodné použiť uzol so sigmoid funkciou a ako stratovú funkciu použiť krížovú entropiu. [5] [7] [66]
- **Multi-triedna klasifikácia** je problém, kde úlohou je zaradiť vzorku do jednej z viacerých než dvoch tried. Koncepcia predikcie pravdepodobnosti je, že vzor patrí do každej triedy. Pre tento problém sa nezaobídeme len s nulou a jednotkou, a preto je vhodné použiť aktivačnú funkciu softmax. Pre stratovú funkciu použijeme kategorickú krížovú entropiu. [5] [7] [66]

- **Regresívne predikovanie modelu** je problém, ktorý sa snaží o predikciu skutočnej hodnoty. Pre tento problém je vhodné použiť uzol s lineárnou aktivačnou funkciou a stratovú funkciu strednej štvorcovej chyby. [5]

Na nasledujúcej podkapitole, sú detailnejšie popísané možnosti pre binárnu klasifikáciu.

3.4.2 Funkcie straty binárnej klasifikácie

- **Binárna krížová entropia** (anglicky *binary cross-entropy*) je určená pre binárnu klasifikáciu, kde cieľové hodnoty sú {0, 1}. Vypočítava skóre, ktoré zahrňuje priemerný rozdiel medzi predpovedanými a skutočnými pravdepodobnosťami pre predikciu triedy 1. Dokonalá hodnota krížovej entropie je nula. [6]

- **Hinge loss** je alternatívou k binárnej krížovej entropii. Vyvinutá je primárne pre modely *SVM* (z anglického *Support Vector Machine*). Určená je pre binárnu klasifikáciu, kde sú hodnoty {-1, 1}. Funkcia sa snaží, aby výsledok mal správne znamienko a priradil viac chýb, ak existuje rozdiel v znamienku medzi predpovedanými a skutočnými hodnotami. Niekedy vedie k lepšiemu výsledku než krížová entropia. [6]

- **Squared hinge loss** je rozšírená funkcia hinge loss, ktorá jednoducho vypočíta druhú mocninu hinge loss. Jej účinkom je vyhladenie funkcie aby sa dalo s ňou ľahšie numericky pracovať. Cieľové hodnoty musia byť tiež v hodnotách {-1, 1}. [6]

Nasledujúca podkapitola bude rozoberať detailnejšie možnosti použitia rôznych stratových funkcií pre multi-triednu klasifikáciu.

3.4.3 Funkcie straty pri klasifikácii viacero tried

- **Viactriedna krížová entropia** (anglicky *multi-class cross-entropy*) je základná stratová funkcia pre problémy viacerých tried. Je vhodná pre cieľové hodnoty {0, 1, 3, ..., n }, kde je priradená každej triede unikátna hodnota. Krížová entropia vypočíta skóre, ktoré zahrňuje priemerný rozdiel medzi skutočným a predpokladaným rozdelením pravdepodobnosti pre všetky triedy. Skóre je minimalizované a dokonalá hodnota krížovej entropie je nula. [6]

- **Riedka strata krížovej entropie pre viac tried** (anglicky *sparse multi-class cross-entropy loss*) je funkcia, ktorá rieši problém pri klasifikácií krížovou entropiou. Problém krížovej entropie je, že môže spôsobiť problémy pri učení s veľkým počtom tried [6]. Takýto problém môže vzniknuť pri predikcii, o aký druh machu sa jedná. Na zemi je približne 15000 [28] rôznych druhov machov. Pri tomto vznikne 15000 rôznych kategórií pre každú triedu. Následkom tohto môže byť, že prvok môže vyžadovať významne veľkú pamäť. Riedka strata krížovej entropie to rieši tak, že prevedie rovnaký výpočet chyby krížovej entropie tak, aby nebolo nutné cieľovú hodnotu pred tréningom kódovať za behu [6].

- **Strata divergencie Kullback Leibler** (skrátene *KL divergence* anglicky *Kullback Leibler divergence*) jedná sa o funkciu, ktorá meria odlišnosť rozdelenia pravdepodobností od základného rozdelenia. V praxi sa táto funkcia správa podobne ako krížová entropia. Základným princípom tejto funkcie je, že počíta koľko informácií vo forme bitov je stratených. Ak rozdiel medzi základným rozdelením a odlišným rozdelením pravdepodobnosti je nula, tak sa jedná o identické rozdelenie. Praktické využitie tejto

funkcie je pri náročnejších klasifikáciach ako len jednoduchá viactriedna klasifikácia. Využitie nájde pri rôznych úlohách, kde je potrebné zrekonštruovať vstup. Avšak, ak sa použije pre klasifikáciu jednoduchej viactriednej klasifikácie, tak správanie tejto funkcie je ekvivaletné ku krížovej entropii viac tried. [6]

- **Kategorická krížová entropia** (anglicky *cross-entropy loss*) v princípe ide o porovnávanie vektoru predpovedí s vektorom pre správny výstup. Vektor predpovedí je výstup aktivačných funkcií, kde jedna pravdepodobnosť je pre každú triedu. Pravdepodobnosti sú uložené v správnom výstupe a jednotka je len v prípade tej triedy, ktorá je správna, ostatné triedy sú nastavené na nulu. Stratová funkcia je nižšia, ak vektor predpovedí je čo najbližšie k správnemu vektoru výstupov. [7] [66]

Posledná podkapitola sa bude zaoberať rôznymi možnosťami pre riešenie problému straty regresie.

3.4.4 Funkcie straty regresie

- **Stredná kvadratická chyba** (skratka *MSE* z anglického *Mean Squared Error*) je základná funkcia, ktorá sa môže použit pre regresné problémy. Táto funkcia je preferovaná, ak jej distribúcia cieľovej premennej je Gaussovo rozdelenie. Ako z názvu vyplýva, tak chyba sa vypočíta ako priemer kvadratických rozdielov medzi predpovedanými a skutočnými hodnotami. Výsledok tejto funkcie je vždy kladný. Pre výslednú hodnotu z funkcie sa udáva dokonalá hodnota *0,0*. Ak väčšie chyby majú za následok viac chýb ako menšie chyby, tak celý model trpí. [6] Z tohto vyplýva, že pri učení ak väčšie chyby vygenerujú menej chýb ako menšie chyby, tak model sa učí lepšie.

- **Stredná logaritmická chyba** (skratka *MSLE* z anglického *Mean Squared Logarithmic Error*) sa použije v prípade, ak cieľová hodnota má rozpätia hodnôt. Funkcia vylepšuje strednú kvadratickú chybu. Jej hlavným vylepšením je, že sa snaží znížiť dopadok na učenie siete, ak sa vyskytnú väčšie chyby, ktoré generujú viac chýb. Z tohto dôvodu sa najskôr vypočíta prirodzený logaritmus každej z predpovedanej hodnoty, a následne sa vypočíta strata strednej kvadratickej chyby. [6]

- **Priemerná absolútna chyba** (skratka *MAE* z anglického *Mean Absolute Error*) je funkcia, ktorá sa využije v prípade, ak distribúcia cieľovej hodnoty je Gaussova rozloženie s odľahlými hodnotami. Vypočítava sa ako priemer absolutného rozdielu medzi skutočnými a predpokladanými hodnotami. [6]

3.5 Optimalizátory

Pre učenie máme koncept straty, ktorý hovorí ako zle si model v danom okamžiku vedie. Pre účely zlepšenia učenia sa tieto straty musia použiť, aby sieť fungovala lepšie. V podstate treba spraviť stratu a pokúsiť sa ju minimalizovať. Proces minimalizácie alebo maximalizácie akéhokoľvek matematického výrazu sa nazýva optimalizácia.

Optimalizátory sú metódy alebo algoritmy, ktoré určujú rýchlosť učenia a presnosť výsledkov. Podkapitola predstaví jednotlivé metódy.

3.5.1 Prechodový zostup

Prechodový zostup (anglicky *gradient descent*) je najzákladnejší a najpoužívanejší algoritmus. Používa sa v lineárnych, regresívnych a klasifikačných algoritmoch. Jedná sa o optimalizačný algoritmus prvého radu, kde závisí na derivácii prvého radu. [18] Vypočítava, akým spôsobom by sa mohli váhy zmeniť, aby funkcia mohla dosiahnuť minima. Funkcia *F(x)* klesá rýchlejšie smerom k najnižšiemu zostupu *w*. Potom sa môže napísať vzťah ako: [66]

$$w_{n+1} = w_n - \gamma \nabla F(w_n) \tag{3.11}$$

Zo vzťahu 3.11 je vidieť, že w_n je aktuálna pozícia, γ je veľkosť kroku a $\nabla F(w_n)$ je smer s najnižším zostupom. Opakovaním vznikne sekvencia, ktorá bude klesať. Postupnými krokmi sa dostane k lokálnemu minimu. Výhodou je jednoduchý výpočet a dá sa ľahko pochopiť. Nevýhodou je jej pomalý postup, pretože pre každú aktualizáciu sa musí vypočítať gradient pre celý súbor dát. [66]

3.5.2 Stochastic Gradient Descent

Stochastický prechodový zostup (skratka *SGD* z anglického *Stochastic Gradient Descent*) je varianta prechodového zostupu. SGD varianta sa snaží častejšie aktualizovať parametre modelu. Patrí medzi najpoužívanejšie. [18] Výhodou algoritmu je lepšia konvergencia k minimu. Nepočíta reálny zostup, ktorý je počítaný pre celý súbor dát, ale stochasticky odhadne ďalšiu optimalizáciu tak, že počíta zostup pre aktuálny prvok. To má význam pri veľkých súboroch dát. Nespornou výhodou je samotná rýchlosť učenia. Nevýhodou je veľká fluktuácia pri konvergencii. [66]

3.5.3 Mini-Batch Gradient Descent

Podľa článku [18] sa jedná o najlepšiu variantu prechodového zostupu. Algoritmus vylepšuje svojich predchodcov SGD a základný prechodový zostup. Princíp tohto algoritmu je, že po každej dávke *(anglicky batch)* aktualizuje parametry modelu. Znamená to, že súbor trenovacích dát, je rozdelený na menšie časti. Po každej časti sa aktualizujú parametre modelu.

3.5.4 Momentum

Momentum bolo objavené kvôli zníženiu vysokého rozptylu v SGD a zmierňujúcej konvergencii [18]. V momente je snaha zachytiť niektoré informácie. Informácie sa týkajú predchadzajúcich aktualízácií, ktorými váha prešla pred prevedením aktualizácie. Váha sa pohybuje určitým smerom a to pomaly môže akumulovať určitú hybnosť. Týmto postupom sa algoritmus snaží obísť lokálne minima a nájsť globálne minimum. Znázornené v nasledujúcich vzťahoch. [17] Vzťah 3.12 popisuje aktualizáciu algoritmu a 3.13 popisuje zmenu váh.

$$v_{new} = \eta v_{old} - \alpha \frac{\partial(Loss)}{\partial(W_{old})} \tag{3.12}$$

$$W_{new} = v_{new} + W_{old} \tag{3.13}$$

3.6 Modely umelého neurónu

Model umelého neurónu musel prejsť vývojom, aby sa mohol používať pre rôzne účely. Podľa článku [53] existujú tri vývojové stupne modelu neurónu. Prvým je McCulloch-Pitts, nasleduje percentrón a posledný je sigmoid. Ďalej v tomto článku je spomenutý moderný umelý model neurónu. Tento model vychádza z pohľadu autora článku na súčasný stav modelu neurónu. Na nasledujúcich podkapitolách budú všetky tieto modely popísané.

3.6.1 McCulloch-Pitts model neurónu

V roku 1943 Warren McCulloch a Walter Pitts napísali dielo [47]. V tomto diele prvýkrát popísali matematický model biologického neurónu. Model dostal podľa nich pomenovanie McCullich-Pitts. Skladá sa zo štyroch častí. [54]

Neurón je výpočtová jednotka, do ktorej vstupujú vstupné signály. Vypočítajú sa vstupné signály a výstup sa aktivuje. Neurón sa skladá z ďalších dvoch častí: [54]

- *Sumačná funkcia* - výpočet súčtu prichádzajúcich vstupov.
- *Aktivačná funkcia* - využíva krokovú funkciu. Popis krokovej funkcie je v podkapitole 3.3 rovnica 3.4.

Excitačný vstup je príchádzajúci vstupný signál do neurónu, ktorý nadobúda dve hodnoty 0 a 1. Vstup je zapnutý, ak príde 1. V opačnom prípade príde 0 a značí vypnutý vstup. [54]

Inhibičný vstup je ďalší typ vstupného signálu do neurónu. Ak je vstup zapnutý, tak neurón sa nepustí ďalej. [54]

Výstup neurónu, ktorý môže mať iba binárne hodnoty 0 alebo 1. Pri hodnote 1 indikuje, že neurón môže byť vyslaný ďalej. Ak je 0, tak je neurón stopnutý. [54]

Fungovanie modelu

Model funguje tak, že sa najprv musia zapnúť vstupy na jedna. Týmto sa aktivuje neurón. Prvým krokom je zistenie, či je zapnutý inhibičný vstup. Ak je tento vstup zapnutý, tak na výstup sa pošle nula a tento neurón sa nespustí. V opačnom prípade sa vypočíta súčet excitačných vstupov, ktoré sú zapnuté. Po súčte priebeha kontrola, či súčet je väčší ako prahová hodnota. V prípade, že je to pravda, tak sa na výstup pošle jedna a neurón sa pustí ďalej. V opačnom prípade sa na výstup pošle nula a neurón sa nespustí. [54]

Obmedzenie modelu

Model neurónu má kopec obmedzení. Najzásadnejším obmedzením je, že vstupné hodnoty sú len booleanovské hodnoty. Ďalším obmedzením je, že nezohľadňuje váhy v modelu. Týmto nezohľadnuje, ktorá zo vstupných funkcií je dôležitejšia a ktorá nie. Posledným dôležité obmedzenie definuje prah ako rozhodcu, či sa neurón spustí alebo nie. Tento prah je určený človekom. [55]

3.6.2 Perceptrón

V roku 1957 bol popísaný ďalší model neurónu a to perceptrón. Podobne ako McCulloch-Pitts model sa skladá zo štyroch častí. [55]

Vstupy perceptrónu môžu byť reálne čísla. Model perceptrónu k vstupu má aj pridruženú váhu. **Váhy** sú spočiatku neznáme. Perceptrón sa ich učí behom tréningovej fázy. [55]

Neurón je podobne definovaný ako u McCulloch-Pits modelu. Jediným rozdielom je, že sumačná a aktivačná funkcia je definovaná inak. Definícia týchto funkcií je nasledovná:

- *Sumačná funkcia* - výpočet súčtu prichádzajúcich vstupov vynásobenými príslušnými váhami.
- *Aktivačná funkcia* - využíva krokovú funkciu. Popis krokovej funkcie je v podkapitole 3.3 rovnica 3.4. Pri perceptróne sa do vstupu krokovej funkcie posiela výsledok sumačnej funkcie.

Výstup neurónu je rovnako definovaný ako u McCulloch-Pits modelu. Teda má iba binárny výstup.

Fungovanie modelu

Perceptrón funguje odlišne ako McCulloch-Pitss model. Prvým rozdielom je, že má od začiatku povolený vstup do neurónu. Následne sa spočíta sumačná funkcia. Ďalej sa vypočíta aktivačná funkcia. Ako bolo spomenuté vstupom do tejto funkcie je výstup zo sumačnej funkcie. Pokiaľ je výstup z aktivačnej funkcie jedna, tak aj na výstupe bude jedna a neurón sa pošle ďalej. V opačnom prípade sa na výstup pošle nula a neurón sa nepošle ďalej. [55]

Obmedzenie modelu

Obmedzením perceptrónu je, že má striknú aktivačnú funkciu. Zo vzťahu 3.4 pre krokovú funkciu je vidieť, že v okolí prahu príde náhle rozhodnutie z 0 na 1. [56] Príkladom môže byť vytvorený perceptrón na predpoveď, či si človek s nejakým ročným príjmom môže kúpiť dom. Povedzme, že prah je 2 milióny českých korún. Ak človek zarobí 1,99 milióna, tak perceptrón mu povie, že nemôže si kúpiť dom, ale ak má 2 milióny, tak môže.

3.6.3 Sigmoid

Sigmoid neurón, ktorý v jadre obsahuje aktivačnú funkciu sigmoid v podkapitole 3.3 rovnica 3.6. Je to podobný model ako perceptrón, ale namiesto krokovej funkcie využíva sigmoid funkciu. Vďaka tomuto rieši obmedzenia perceptrónu. Sigmoid sa skladá z 5 častí.

Vstupy sigmoid neurónu môžu predstavovať akékoľvek reálne číslo. [56]

Bias neurónu je ďalší parameter učenia, ktorý pomáha generovať posun aktivačnej funkcie. Pomáha presunúť graf aktivačnej funkcie, aby lepšie odpovedalo dátam. Bias nie je závislý na vstupe. [56]

Podobne ako u perceptrónu, tak aj sigmoid má váhu spätú so vstupom. Tieto **váhy** sú na začiatku neznáme, ale sú učené behom tréningovej fázy. [56]

Neurón má podobnú definíciu ako perceptrón model, až na to, že ako aktivačnú funkciu používa sigmoid.

Výstup neurónu, ktorý môže nadobúdať hodnoty od 0 do 1. [56]

Fungovanie modelu

Model funguje podobne ako perceptrón. Rozdielom vo fungovaní modelu je práve už niekoľkokrát spomínaná aktivačná funkcia. Výstupom z aktivačnej funkcie, ale aj z neurónu sú hodnoty od nula do jedna.

3.6.4 Moderný umelý neurón

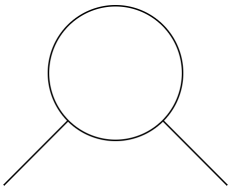
Ako bolo spomenuté v úvode tejto podkapitoly 3.6, tak tento model je len pomenovaním autora článku [53]. Tento typ neurónu sa používa v súčasnej dobe pri hlbokom učení. Tento model od ostatných troch predošlých neurónov sa líši v použitej aktivačnej funkcii. Aktivačná funkcia je základným kameňom tohto neurónu. Skladá sa z piatich častí, tak ako sigmoid neurón. Jediným rozdielom je práve iný typ aktivačnej funkice. Aktivačná funkcia môže byť napríklad tanh, relu alebo softmax. Tieto jednotlivé aktivačné funkcie sú popísané v podkapitole 3.3.

Fungovanie modelu

Model funguje podobne ako sigmoid. Rozdielom vo fungovaní modelu je práve už niekoľkokrát spomínaná aktivačná funkcia. Výstupom z tohto neurónu je rozsah daný aktivačnou funkciou.

3.7 Viacvrstvové neurónové siete

V predchádzajúcej kapitole 3.6 boli popísané modely ako perceptrón, sigmoid a moderný umelý neurón. Tieto modely môžu tvoriť jednoduchú jednovrstvovú neurónovú sieť. Taktiež boli popísané obmedzenia pri klasifikácii. Jednovrstvová neurónová sieť sa skladá iba z jedného modelu neurónu. Toto je možné vidieť na obrázku 3.3.



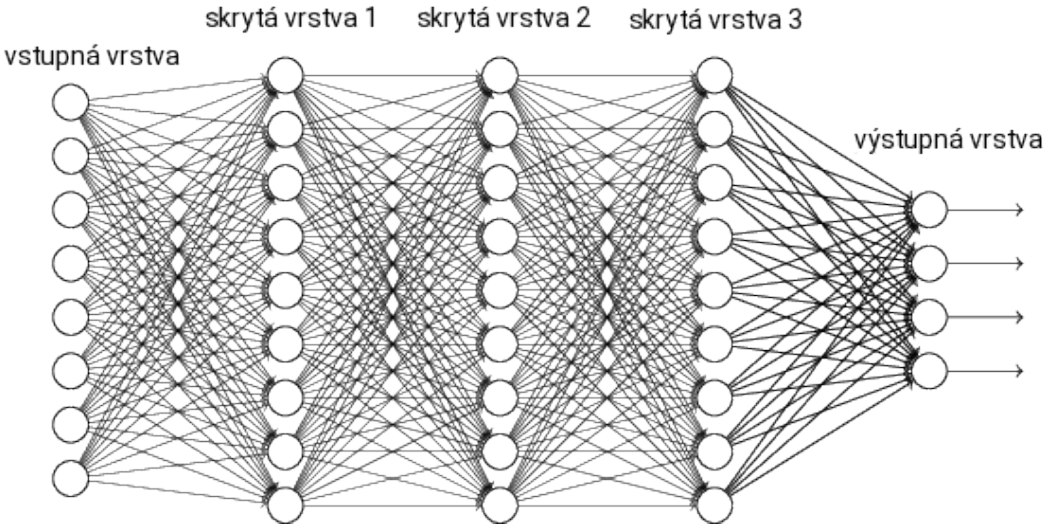
Obrázok 3.3: Jednovrstvová neurónová sieť.

Pre praktické použitie neurónových sietí je potrebné vytvoriť viacvrstvové neurónové siete. Príklad viacvrstvovej neurónovej siete je možné vidieť na obrázku 3.4. Ich základnou vlastnosťou je, že majú viacero vrstiev. Vrstvy neurónových sietí sa môžu rozdeliť na vstupno-výstupné a skryté vrstvy. Skryté vrstvy prepájajú vstupno-výstupné vrstvy s vstupno-výstupnými uzlami. Skryté vrstvy sú oddelené od vstupno-výstupných uzlov. [33]

Vstupné vrstvy sú dôležité pre neurónové siete. Pomocou týchto vrstviev vstupuje napríklad obrázok na trénovanie alebo detekciu. V týchto vrstvách je definovasný počet dávok (*batch*) obrázkov. To hovorí koľko obrázkov sa jeden okamžik vloží do siete. Ďalej v týchto vrstvách sa definuje šírka a výška jednotlivého obrázka plus počet kanálov. Ak sú tri kanály značí to farebný obrázok. Obrázok v tieňoch sivej má kanál iba jeden. Sú plne prepojené tieto vrstvy.

Skryté vrstvy sú medzi vstupnými a výstupnými vrstvami. Skryté vrstvy vo viacvrstvových sieťach sú plne prepojené. Tieto vrstvy pomáhajú počítať sumačnú funkciu a počítajú aktivačnú funkciu. Na základe výsledku aktivačnej funkcie preposielajú informácie do ďalších neurónov a vrstiev.

Výstupné vrstvy sú posledné v sieti. Tieto vrstvy definujú, koľko rôznych tried bude klasifikátor môcť určiť. Taktiež sú plne prepojené.



Obrázok 3.4: Viacvrstvová neurónová sieť. Upravené, zdroj: [60].

Dôležitou vlastnosťou aktivačnej funkcie každého neurónu je, že musí byť diferencovateľná. Táto vlastnosť sa využíva pri stratégii spätného šírenia (anglicky *backpropagation optimization strategy*). Viacvrstvové neurónové siete sa vyznačujú vysokou konektivitou vďaka synaptickým váham. [33] [66]

S vyššie uvedenými vlastnosťami sa spája problém, že vieme strašne málo o fungovaní takýchto sietí. Pri využívaní skrytých vrstiev, sa dá len ťažko vizualizovať učiaci proces. Učiaci proces rozhoduje o tom, ktoré vstupné znaky majú byť použité v skrytých vrstvách. Vysoká konektivita je tiež ďalšou vlastnosťou, ktorá nám neuľahčuje zistenie fungovania viacvrstvých neurónových sietí. [66]

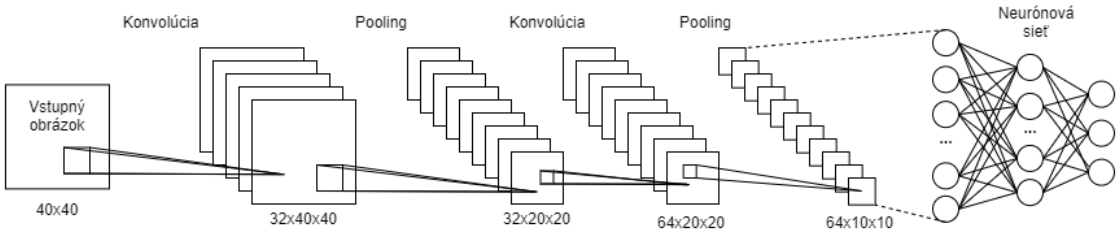
Problémom viacvrstvových neurónových sietí je plné prepojenie skrytých vrstviev. Tento problém je obmedzením v prípade, ak nie je veľká vstupná databáza, pretože sa nedokážu pokryť všetky kombinácie plne prepojených skrytých vrstviev. Ďalej sú závislé na veľkej databáze. Z tohto pohľadu nie sú odolné voči posunu a šumu. Príkladom môže byť rukou písané číslo, ktoré má rôzne sklony. Ďalším obmedzením je, že takáto sieť bude vo výsledku veľmi veľká a ťažko použiteľná. Našťastie tento problém riešia napríklad konvolučné neurónové siete, ktoré sú popísané na nasledujúcej kapitole.

3.8 Konvolučné neurónové siete

Konvolučné neurónové siete (v skratke *CNN* z anglického *Convolutional Neural Network*) sú špeciálne typy neurónových sietí pre spracovanie dát. Jej topológia je podobná mriežke. Príkladom konvolučnej neurónovej siete s 1D mriežkou môže byť meranie vlhkosti zeminy v pravidelných časových úsekoch. Druhým príkladom sú obrazové dáta, ktoré je možné považovať za 2D mriežku. Konvolučné neurónové siete dostali názov podľa používanej matematickej operácie konvolúcia. Tieto siete majú nižšie požiadavky ako bežné siete. Vďaka týmto nižším požiadavkám sa využívajú v praxi. [30] Ako bolo spomenuté v podkapitole 3.7, tak konvolučné neurónové siete riešia problém plne prepojených skrytých vrstiev.

V konvolučných neurónových sieťach je možné pozorovať tri návrhové myšlienky. Tými myšlienkami sú získavanie príznakov z recepčných polí, zdieľanie váh a podvzorkovanie obrazu, ktoré zabezpečujú nemennosť voči šumu, posunom a zmene veľkosti. Recepčné

polia nám umožňujú získavať základné časti ako je orientácia, konce, rohy, veľkosti uhlov a iné. Základné časti sú potom skladané do väčších znakov. Príklad takej konvolučnej siete je možné vidieť na obrázku 3.5. [32] [66]



Obrázok 3.5: Architektúra konvolučnej neurónovej siete. Zdroj: [66].

Konvolučná neurónová sieť používa usporiadanie neurónov do roviny a zdieľa rovnakú množinu váh. Neuróny vykonávajú tú istú operáciu nad rôznymi časťami obrázku. Výstupy z neurónov sú organizované do máp vlastností. Vrstva konvolučnej siete je tvorená z viacerých máp, ktoré vznikajú použitím rôznych váh. Z obrázku 3.5 je možné si všimnúť operáciu *pooling*. Pooling spája dve operácie. Prvou je zmenšenie rozlíšenia (*sub-sampling*) a druhá je lokálne spriemerovaná mapa. Pooling znižuje vplyv šumu a posunu výstupu. [32] [66]

Konvolučné neurónové siete môžu používať aj *max-pooling* vrstvy. Max-pooling vrstva slúži na podzvorkovanie vstupu. Funguje tak, že vyberá maximálnu hodnotu z matice pokrytej filtrom. Výstup z tejto vrstvy je matica, ktorá obsahuje najvyššiu hodnotu z predošlej matice. Max-pooling združuje dve operácie. Prvá operácia je filtrovanie a druhá operácia *stride* zmenší maticu len na najvyššie hodnoty. Podobnou vrstvou je *average pooling vrstva*. Táto vrstva funguje na rovnakom pricípe ako max-pooling vrstva, ale s rozdielom, že hľadá priemer hodnoty v matici.

3.9 Učenie konvolučnej siete

Pre učenie neurónových sietí sa používa najznámejšia metóda spätného šírenia (Backpropagation method). Táto metóda využíva gradient descent algoritmus 3.5.1. Metóda znižuje nepresnosti pri učiacom procese, ktorá upravuje jednotlivé prepojenia váh. Váhy sú závislé na vypočítanej stratovej funkcii. Výstup z funkcie je prenesený do vyšších vrstiev pomocou metódy spätného šírenia. [32]

Metóda spätného šírenia má veľmi veľké uplatnenie. Jedno z najvýznamnejších je rozpoznávanie vzorov. Jej myšlienkou je, že prechod môže byť efektívne vypočítaný z výstupu na vstup. Môže byť náročná na veľkých sieťach. [32]

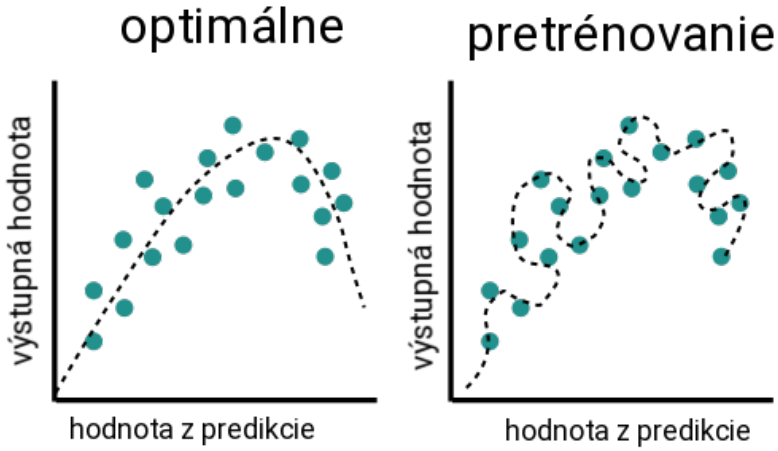
Pre učenie je dobré poznať tri pojmy, ktoré sú dôležité pri trénovaní siete. Prvým pojmom veľkosť dávky (batch size). Tento pojem hovorí, koľko dát alebo obrázkov sa zmestí do siete. Napríklad, ak dávka je šesťnásť, tak do modelu siete môže naraz šesťnásť obrázkov. Druhým pojmom je *iterácia*. Iterácia nám hovorí, koľko takých dávok bude v jednom učiacom cykle. Iterácia môže byť definovaná používateľom alebo definovaná podielom počtu obrázkov v databáze a veľkosťou dávky. Jeden celý učiaci cyklus sa nazýva *epocha*.

Pri učení sa stáva, že neurónové siete sa môžu buď pretrénovať (*overfitting*) alebo podtrénovať (*underfitting*). Nasledujúce podkapitoly vysvetlia rozdiel medzi nimi.

3.9.1 Pretrénovanie siete

Pretrénovanie siete je prípad, keď sa po dlhú dobu nechá pustený trénovací algoritmus. Natrénovaný model sa pri tomto stave naučí vzory a šumy, ktoré nie sú potrebné. To má za následok, že nebude môcť nájsť vzory v reálnych dátach. Síce bude mať vysokú úspešnosť trénovania, ale pre praktické použitie bude nepoužiteľný. [1]

Aby model nebol pretrénovaný, tak sa môže použiť predčasné ukončenie trénovania. Toto môže spôsobiť, že model bude podtrénovaný. V praxi používaným spôsobom pre zamedzenie pretrénovania siete sa používa validačná databáza. Jedná sa o databázu, ktorá sa vyčlení z trénovacej databázy. K validačnej databáze nemá model počas učenia prístup. Po učení sa na validačnej databáze overí natrénovaný model. Validačná databáza simuluje reálne dáta na detekciu. Na obrázku 3.6 je vidieť model pretrénovaný a model, ktorý sa natrénoval správne.



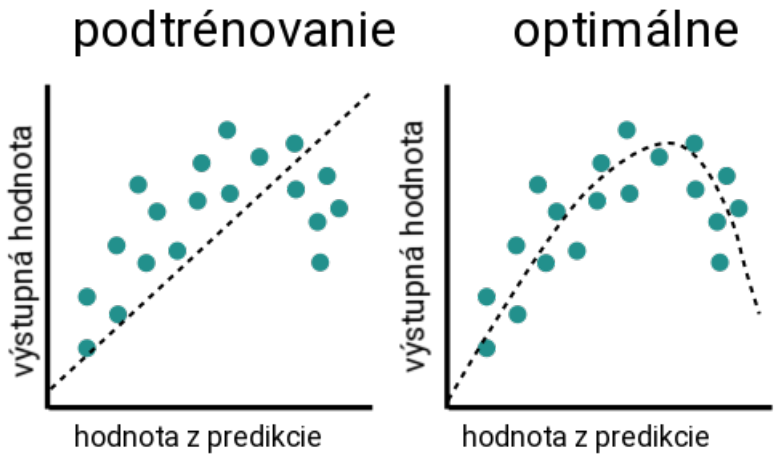
Obrázok 3.6: Pretrénovanie. Upravené, zdroj: [3].

3.9.2 Podtrénovanie siete

Podtrénovanie je prípad, kedy sa model dostatočne nenaučil na tréningových dátach. Následok je nízke zovšeobecnenie a nespoľahlivé predpovede. Podtrénovanie je tiež veľmi zlé ako pretrénovanie. Vo vysokom skreslení nemusí byť model dostatočne flexibilný. Na obrázku 3.7 je vidieť podtrénovaný a optimálne natrénovaný model. Podtrénovanie môže vzniknuť aj predčasným ukončením trénovania, ako to bolo spomenuté v podkapitole 3.9.1. [1]

3.10 Predtrénované modely pre spracovanie obrazu

Predtrénované modely sú modely, ktoré trénoval niekto iný. Tieto modely riešia rovnaké alebo podobné problémy, ktoré vývojár alebo výskumník rieši. Použitie takýchto modelov nám uľahčí prácu pri vylepšovaní klasifikátora pre špecifické potreby. Predtrénovaný model bol trénovaný na obrovských databázach a v dlhom časovom úseku. Zároveň je možnosť využiť iba kód modelu a trénovať ho od začiatku. Pri druhej možnosti strácame výhody predtrénovaného modelu. Na druhú stranu, nám poskytuje model pre trénovanie, ktorý už niekto pred nami vymyslel. Problémom môže byť, že všetky modely sú trénované na farebných obrázkoch, ale sieť bude potrebovať vstupné obrázky v odtieňoch sivej. Tento



Obrázok 3.7: Podtrénovanie. Upravené, zdroj: [3].

problém by sa mal dať zmeniť, že sa zmení prvá vstupná vrstva, ktorá definuje kanál s farbami.

Predtrénované modely v praxi sú vyvinuté skoro vždy technologickými gigantmi alebo skupinou úspešných vedcov. V nasledujúcej podkapitole budú popísaé modely VGG16, VGG19, Inceptionv3 (GoogleLeNet), ResNet50.

3.10.1 VGG16

Model VGG16 je vytvorený K. Simoyan a A. Zisserman z Oxforskej univerzity. Model vyhral súťaž ILSVR (ImageNet Large Scale Visual Recognition Challenge) v roku 2014. Jedná sa o súťaž v detekcii rôznych obrázkov. Dodnes je považovaný za jeden z najlepších modelov pre počítačové videnie. Architektúra modelu používa konvolučné vrstvy s malými recepčnými poliami namiesto používania veľkého množstva hyperparametrov. [59] [62]

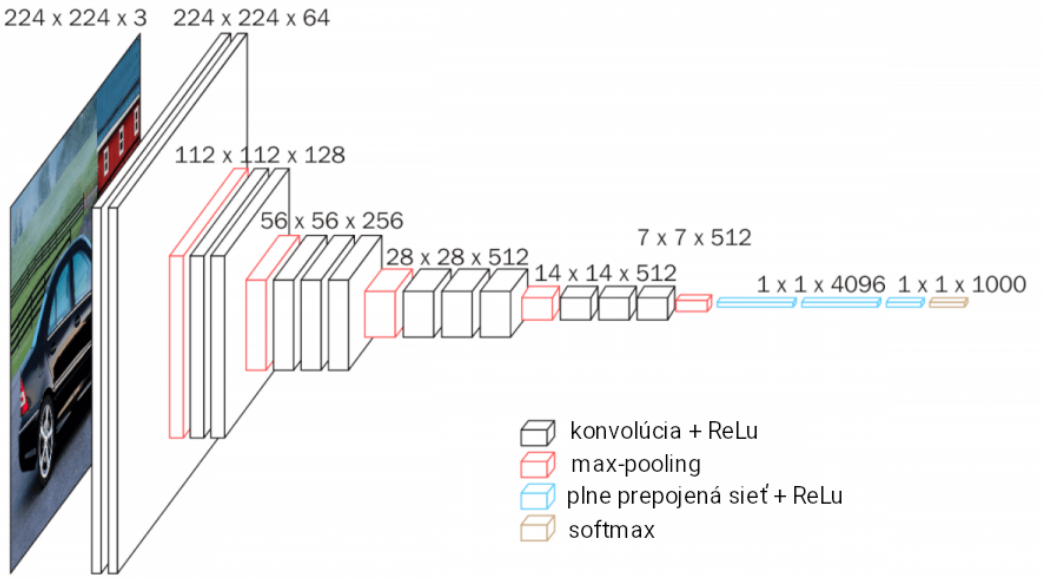
Vstupom do prvej vrstvy je pevne daný farebný obrázok 224×224. Model obsahuje niekoľko konvolučných vrstiev, ktoré používajú filtre s veľmi malým receptívnym poľom o veľkosti 3×3. Je to najmenšia veľkosť k zachyteniu postupu okolia pixelu. Konvolučný krok je zachovaný na jeden pixel. [59] [62]

Konvolučné vrstvy môžu byť zaradené viackrát za sebou. Všetky skryté vrstvy používajú ako aktivačnú funkciu ReLu, ktorá je popísaná v podkapitole 3.3 rovnica 3.9. Priestorové združovanie sa prevádza piatimi vrstvami max-pooling. Max-pooling sa vykonáva v okne o veľkosti 2×2. Výsledkom max-pooling je o polovicu menšia matica. Na obrázku 3.8 je vidieť zoradené tieto vrstvy. [59] [62]

Po konvolučných vrstvách nasledujú tri plne prepojené vrstvy, ktoré sledujú tieto konvolučné vrstvy. Prvé dve majú 4096 kanálov každá. Tretia obsahuje 1000 kanálov, pre každú triedu jednu. Posledná vrstva je soft-max, ktorá je zároveň výstupnou vrstvou. [59] [62]

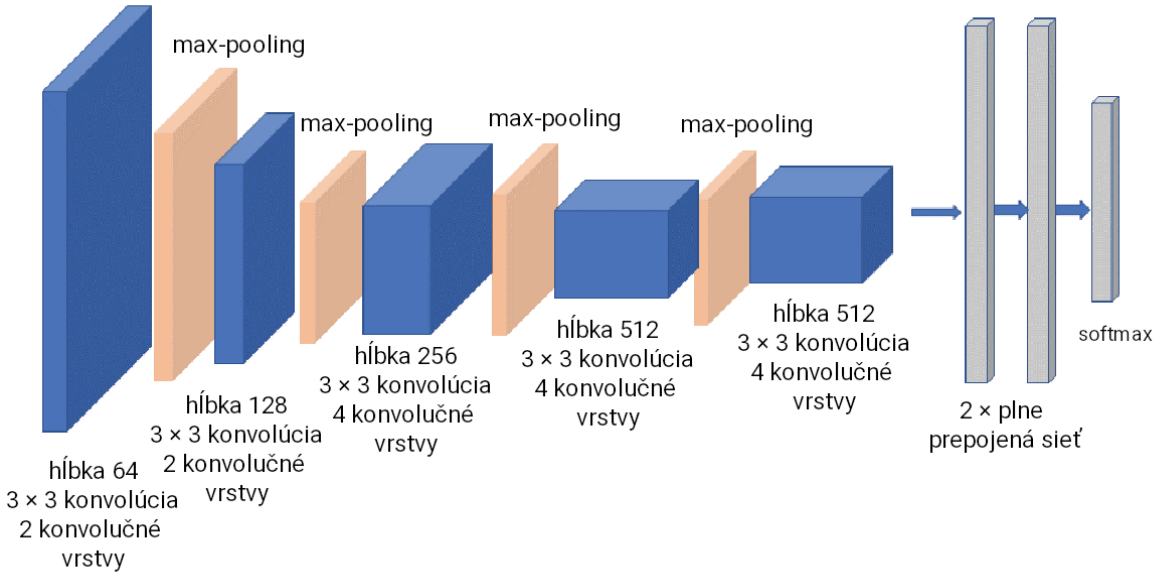
3.10.2 VGG19

Model VGG19 ako aj VGG16 je vytvorený K .Simoyan a A. Zisserman z Oxforskej univerzity. Má 19 vrstiev [70]. Vstup do siete je farebný obrázok o veľkosti 224×224. Jediným predspracovaním je odpočítanie priemernej hodnoty RGB od každého pixelu. Priemer je vypočítaný pre celý súbor dát. Podobne ako VGG16, popísaný v podkapitole 3.10.1 funguje



Obrázok 3.8: Architektúra VGG16 modelu. Upravené, zdroj: [58].

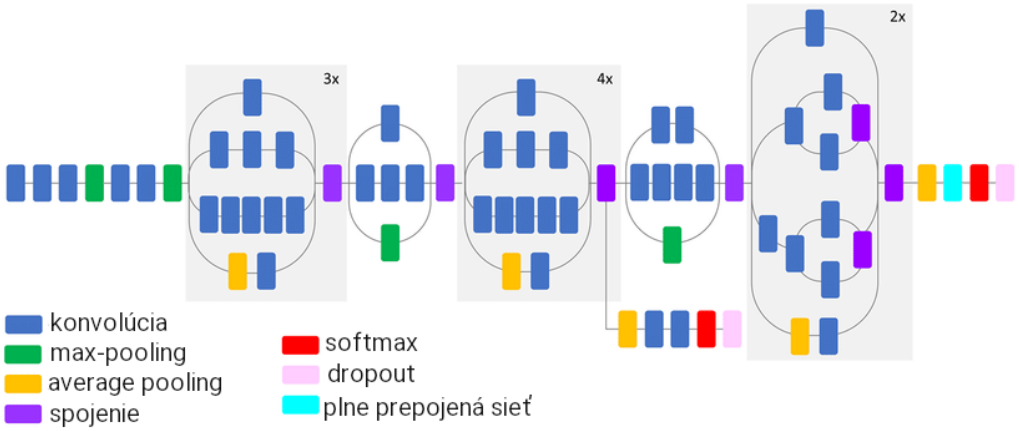
aj model VGG19. Jadro architektúry je rovnaké ako u VGG16, kde skryté vrstvy používajú ReLu aktivačnú funkciu. Rovnako aj po konvolučných vrstvách sú implementované tri plne prepojené vrstvy, ktoré na poslednej vrstve majú funkciu softmax, ktorá dokáže identifikovať do 1000 tried. Toto je možné vidieť na obrázku 3.9. [43] [62]



Obrázok 3.9: Architektúra VGG19 modelu. Upravené, zdroj: [11].

3.10.3 Inceptionv3 (GoogLeNet)

Inceptionv3 je konvolučná neurónová sieť, ktorá má hĺbku 50 vrstiev a je vytvorená spoločnosťou Google. [63] Jedná sa o tretiu verziu tejto siete. Architektúra Inceptionv3 sa skladá z piatich princípov, ktoré sú krok za krokom vytvárané. Prvým krokom je faktorizovaná konvolúcia. Tento krok pomáha znížiť vypočtové nároky a počet parametrov, ktoré vstupujú do siete. Zároveň kontroluje účinnosť samotnej siete. Druhým krokom je menšia konvolúcia. Pomocou tejto konvolúcie sa znižuje počet parametrov. Používajú sa dva filtre o veľkosti 3×3. Nasledujúcim krokom je asymetrická konvolúcia. Konvolúcia 3×3 môže byť nahradená konvolúciou 1×3 a nasledujúca konvolúcia by bola 3×1. Tento princíp znižuje počet parametrov. Predposledným krokom je pomocný klasifikátor. Jedná sa o malú konvolučnú neurónovu sieť, ktorá je vložená medzi vrstvy behom tréningu. Strata s tejto siete sa pripočíta ku celkovej strate celej siete. V tejto sieti funguje ako regulátor. Posledným krokom je efektívne zmenšenie matice. Architektúra nepoužíva tradičné zmenšenie matice pomocou operácie pooling. Snaží sa zabrániť vytvoreniu úzkeho bodu pred aplikovaním max-pooling alebo average pooling. Pre tieto problémy je v architektúre navrhnutý paralérny blok, ktorý expanduje filtre. Z pohľadu výkonu je to jenoduchá operácia a zabráni sa vytvoreniu úzkemu bodu. Model bol trénovaný na farebných obrázkoch s rozlíšením 299x299 pixelov. Architektúra je znázornená na obrázku 3.10. [63] [64]

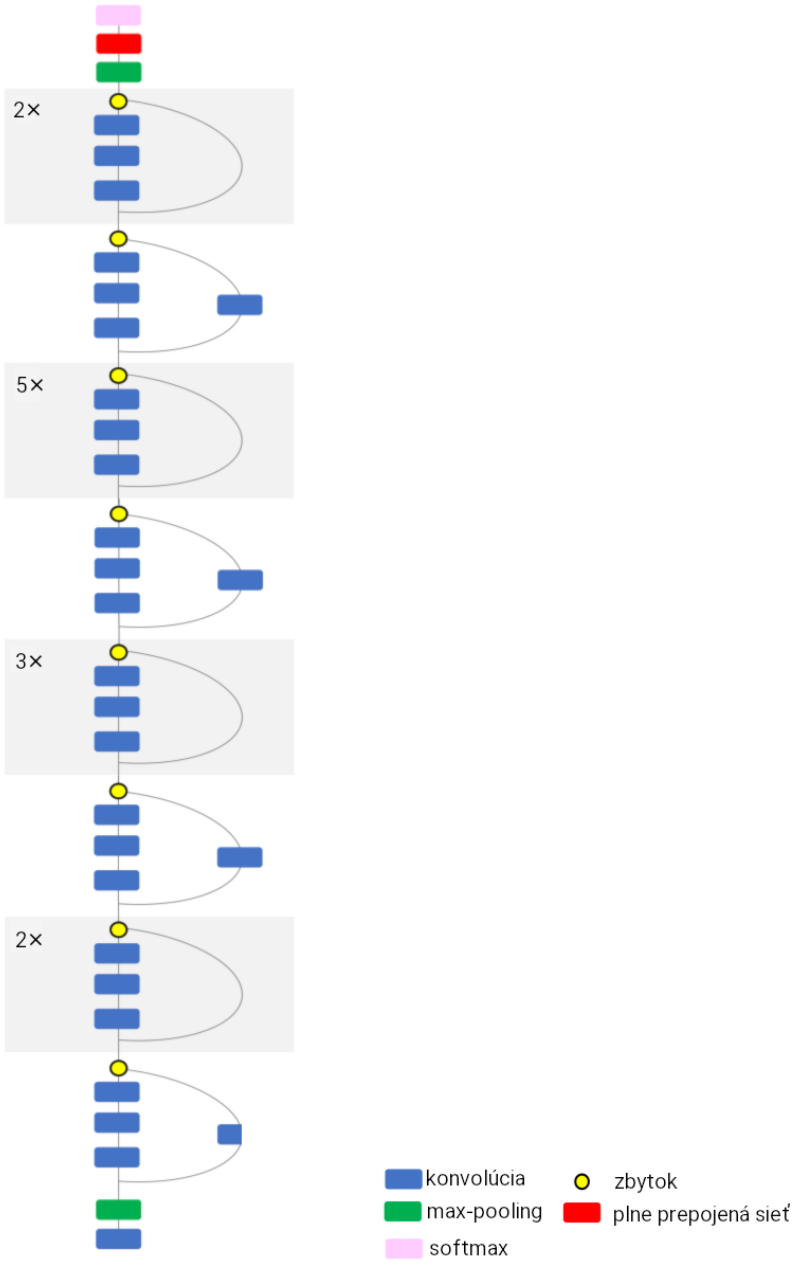


Obrázok 3.10: Architektúra Inceptionv3 modelu. Upravené, zdroj: [44].

3.10.4 ResNet50

ResNet50 je konvolučná neurónová sieť, ktorá má hĺbku 50 vrstiev a je vyvinutá spoločnosťou Microsoft [70]. Základom tejto konvolučnej siete sú zbytkové siete. Zbytkové siete sú bloky v konvolučnej sieti, ktoré majú medzi sebou skratky. Pomocou skratky môžu preskočiť časť blokov v konvolučnej sieti. Tento spôsob umožňuje predísť pretrénovaniu siete. ResNet50 má štyri fázy. Sieť môže brať na vstup obrázok, ktorý má výšku a šírku v násobkoch 32 a 3 ako šírku kanálu. Architektúra prevádza počiatočnú konvolúciu. Max-pooling používá 7×7 a 3×3 pixelov. Po tomto začne prvá etapa, ktorá má 3 zbytkové bloky. Každý blok obsahuje 3 vrstvy. [34] Pre konvolúciu vo všetkých 3 vrstvách bloku, sa v jadre používa 64, 64 a 128. Do druhej fázy sa veľkosť vstupu zníži o polovicu, ale šírka kanála sa

zdvojnásobí. Ako sa prechádza z fáze do fáze, tak vstup sa zníži o polovicu a šírka kanálu sa zdvojnásobí. ResNet50 má 3 vrstvy preskladané jedna na druhú pri zbytkovej funkcii. Výstupná vrstva je plne prepojená a dokáže klasifikovať až 1000 tried. Obrázok 3.11 zobrazuje architektúru siete, kde je agregovaný pohľad na ňu. [34]



Obrázok 3.11: Architektúra ResNet50 modelu. Upravené, zdroj: [44]

Kapitola 4

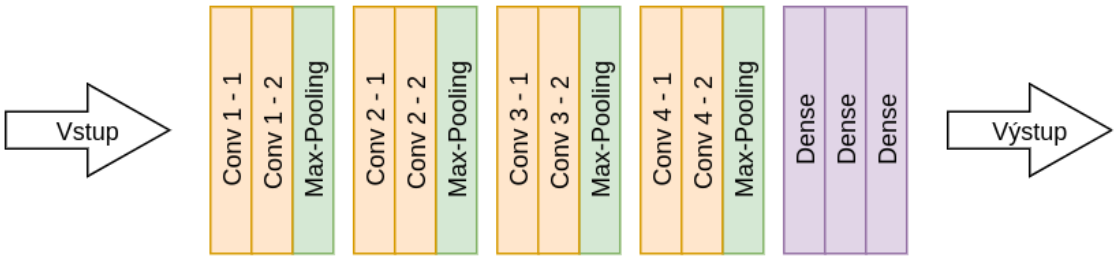
Návrh riešenia

Cieľom práce je experimentálne vylepšenie stávajúcej konvolučnej siete pre detekciu a klasifikáciu ochorení odtlačku prstu, ktorá bola vyvinutá v bakalárskej práci [66]. Ochoreniami sú bradavice a dyshidróza, ktoré sú predstavené v podkapitole 2.8. Po experimentálnom vylepšení je ďalším cieľom vytvorenie nového klasifikátora, ktorý bude detekovať a klasifikovať poškodenie odtlačku prstu pri snímaní. Cieľom je vytvorenie takých klasifikátorov, aby fungovali bez zásahu používateľa. Z týchto dôvodov je postup návrhu nasledovný:

- Popis stávajúceho riešenia.
- Návrh experimentálneho vylepšenia.
- Návrh datasetu pre ochorenia a poškodenie pri snímaní.
- Preskúmať a navrhnúť predspracovanie pre vstup do konvolučnej siete.
- Nový klasifikátor, pre poškodenie odtlačkov prstov pri snímaní.

4.1 Popis aktuálneho stavu

Aktuálne riešenie konvolučnej neurónovej siete je vyvinuté pomocou knižnice Keras [12]. Keras je knižnica pre vývoj neurónových sietí, ktorá beží nad platformou TensorFlow [31]. Knižnica je napísaná v jazyku Python. Toto riešenie používa internú databázu odtlačkov prstov skupiny STRaDe. Ďalšou databázou, ktorú riešenie používa je NIST SD4 DB [67]. Na nasledujúcom obrázku 4.1 je popísaná architektúra súčasného riešenia.



Obrázok 4.1: Model súčasného riešenia. Zdroj: [66]

4.1.1 Použitý model

Pre výslednú konvolučnú sieť je vytvorený sekvenčný model, ktorý berie za predtrénovaný vzor VGG16, ktorý je popísaný v podkapitole 3.10.1. Vzor bol vybraný v práci kvôli častému používaniu v praxi. [66] Nevýhodou modelu VGG16 je, že zaberá 528 MB [61]. Model obsahuje vstupnú vrstvu, cez ktorú bude vstupovať obrázok na vyhodnotenie. Veľkosť obrázka pre vstup do siete je 128×128 pixelov. Vstupná vrstva využíva 64 filtrov s veľkosťou 3×3. Skladá sa z niekoľkých konvolučných blokov a max-pooling vrstiev. [66] Toto zloženie je vidieť na obrázku 4.1. Posledná max-pooling vrstva je prepojená s vyhodnocovacím blokom, ktorý obsahuje dve plne prepojené vrstvy a jednu výstupnú vrstvu. Výstupná vrstva obsahuje pravdepodobnosti výskytu jednotlivých tried. [66]

4.1.2 Použitá aktivačná, stratová a optimalizačná funkcia

Riešenie používa ReLu funkciu v skrytých vrstvách, ktorá je popísaná v podkapitole 3.3. V podkapitole je popis funkcie ale aj nevýhody tejto funkcie. Pre rozhodovanie na výstupnej vrstve používa funkciu softmax, ak pre trénovanie sa určia tri výstupy. Pre klasifikáciu pomocou sigmoid funkcie sa použije, ak používateľ určí výstup modelu pre dve triedy. Táto funkcia je taktiež popísana v podkapitole 3.3.

Z návrhu riešenia v bakalárskej práci [66] vyplýva, že pre výstup dvoch tried sa použije ako stratová funkcia binárna krížová entropia. V prípade, detekcie viac ako dvoch tried sa použije kategorická krížová entropia.

Aktuálne riešenie využíva SGD optimalizáciu [66]. Popis tejto aktualizácií je v podkapitole 3.5.2. Tento typ optimalizácie je vybraný v bakalárskej práci [66] z dôvodu najčastejšieho použitia.

4.1.3 Rozbor klasifikácie a učenia

Klasifikácia priebeha tak, že na vstup konvolučnej siete sa dostane obrázok, ktorý bol upravený. Prvou úpravou obrázka je, že ak je farebný, tak sa prevedie do odtieňov šedej. Následne je upravovaný prahovaním, aby výstupili kontúry odtlačku prsta. [66] Po týchto úpravách sa ešte očistí okolie a rôzne šumy a odtlačok je pripravený na vstup. Poslednými úpravami je zmena rozmeru pomocou mierky *(scale)* na 128×128 pixelov a nastavenie jedného kanála. Samotná klasifikácia ochorenia funguje tak, že sa posúva okno konvolučnej siete. Pre rozhodovanie na poslednej vrstve sa používa funkcia softmax alebo sigmoid, podľa natrénovaného modelu. [66]

V riešení používateľ môže ovplyvniť učenie neurónovej siete. Ovplyvní to vstupnými prepínačmi skriptu, pomocou ktorých môže ovplyvniť počet epoch, počet krokov v jednotlivej epoche a počet vzoriek, ktoré vstupujú do epochy. Samotné učenie prebieha pomocou funkcií z knižnice Keras. [66]

4.2 Návrh experimentálneho vylepšenia

Návrh experimentálneho vylepšenia spočíva v troch základných oblastiach. Prvou oblasťou bude vymenený predtrénovaný model. Následne sa rozoberú pohľady na aktivačnú funkciu. Poslednou oblasťou bude rozobratá zmena optimalizátora.

4.2.1 Zmena predtrénovaného modelu

Ako bolo spomenuté v podkapitole 4.1.1, tak použitý predtrénovaný model je VGG16. Spomína sa tiež, že jeho veľkosť je 528 MB. Pre menšie neurónové siete, ktoré rozpoznávajú do pár desiatok tried je model veľký.

Prvým predtrénovaným modelom, ktorý by sa mohol použiť pre zlepšenie je model VGG19. Tento model je popísaný v podkapitole 3.10.2 a vychádza z rovnakého konceptu ako model VGG16. Jeho veľkosť je 549 MB [70]. Z tohto dôvodu sa iný model architektúrou VGG neoplatí testovať pre zmenu predtrénovaného modelu z dôvodu väčšej veľkosti ako je VGG16.

Nasledujúcim kandidátom na zmenu predtrénovaného modelu je Inceptionv3, popísaný v podkapitole 3.10.3. Tento predtrénovaný model je zaujímavý z pohľadu, že používa hľadanie optimálnych, lokálnych, riedkych štruktúr v konvolučnej sieti. Je to iný princíp ako používa VGG16. Tento model sa vyznačuje tým, že sa snaží nájsť optimálne miestne konštrukcie a opakovať ich do priestoru. Vďaka ďalšiemu konceptu prekladať vrstvy podľa korelácie, je možné zlepšiť existujúce riešenie pre detekciu a klasifikáciu odtlačkov prsta. Ďalšou pozitívnou vlastnosťou je, že bol trénovaný na obrázkoch 299×299, čím je obrázok na šírku a výšku o 171 pixelov väčší. Ďalším plusom tohto predtrénového modelu je, že lepšie odhaduje triedu ako VGG19 a veľkosť modelu Inceptionv3 je len 92 MB [70].

Oproti predtrénovanému modelu Inceptionv3 od spoločnosti Google stojí predtrénovaný model od spoločnosti Microsoft ResNet50, popísaný v podkapitole 3.10.4. Tento model vychádza z architektúry ResNet. Táto architektúra ako je spomínané v popise ResNet50, tak využíva zbytkové siete. Keďže sa konvolúcia používa na zbytkových blokoch, tak toto môže priniesť zlepšenie pri problémoch malých bradavíc ako aj lepšie rozpoznanie dyshidrózy. Ďalším zaujímavým faktom je, že pri každej ďalšej fáze sa vstup zmenší o polovicu, ale kanály sa zdvojnásobia. Pozitívnou vlastnosťou je jeho veľkosť, ktorá je len o 6 MB väčšia ako u predtrénovaného modelu Inceptionv3. Architektúra ResNet má viac modelov, podľa toho, koľko vrstiev obsahuje. ResNet50 obsahuje 50 vrstiev ako aj Inceptionv3.

Pre zmenu modelu budú využité obidva modely. Učenie jednotlivých neurónových sietí môže prebehnúť paralelne. Po tomto učení prebehne testovacia fáza, ktorá rozhodne, či zmena modelu mala vplyv na výsledky, ktoré sú popísané v bakalárskej práci [66].

4.2.2 Zmena aktivačnej funkcie

V podkapitole 4.1.2 je popísaný typ používanej funkcii v aktuálnom riešení [66]. Tou funkciou je ReLu. Táto funkcia trpí nedostatkom pri záporných hodnotách, kde vracia nulu. Opravu tohto nedostatku rieši leaky ReLu, ktorá je popísaná v podkapitole 3.3. Možným zlepšením môže byť aj tanh funkcia, ktorá je popísaná v podkapitole 3.3. Podobne ako zmena modelu, tak aj zmena aktivačnej funkcie bude podrobená testovaniu po učení a porovnávanie s výsledkami z bakalárskej práce [66].

4.2.3 Zmena optimalizátora

Zmena optimalizátora nie je triviálna záležitosť ako sa môže zdať. V podkapitole 3.5 je vysvetlené, že aj najlepší model môže byť zlý, ak sa použije nesprávny typ optimalizátora. Pre overenie fungovania aktuálneho optimalizátora sa uvidí až po zmene modelu a aktivačnej funkcii. Nádejným kandidátom na zmenu optimalizátoru je Mini-Batch Gradient Descent popísaný v podkapitole 3.5.3. Uvažovanie o tejto optimalizačnej metóde je z dovôdu, že je

vylepšením metódy SGD. Tou najzásadnejšou zmenou je, že aktualizuje parametre modelu po každej dávke oproti SGD, kde sa aktualizujú parametre častejšie pre celý súbor dát.

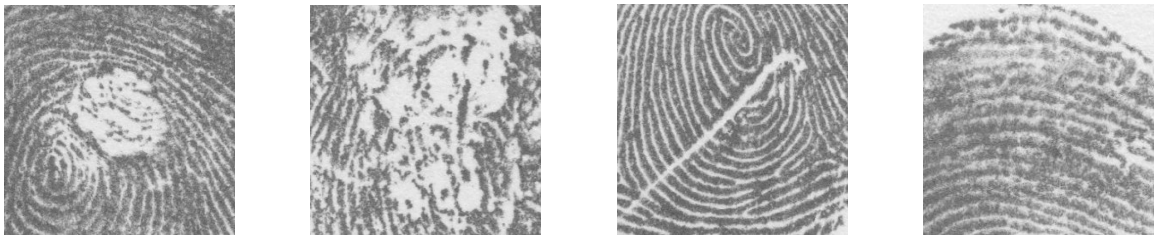
4.2.4 Zhrnutie

V následujúcom zhrnutí je popísané, aké zmeny budú vykonané.

- Zmena modelu na Inceptionv3 alebo ResNet50 podľa výstupných testov po učení aktuálnej konvolučnej siete.
- Zmena aktivačnej funkcie z ReLu na leaky ReLu alebo tanh.
- Zmena optimalizátoru až po testovaní prvých dvoch zmien.

4.3 Trénovacia množina pre ochorenia

Trénovacia množina pre ochorenia bude zostavená z viacerých zdrojov. Najdôležitejším zdrojom odtlačkov prstov budú syntetické odtlačky prstov. Do týchto syntetických odtlačkov budú vygerenované prejavy sledovaných ochorení. Ďalším zdrojom pre trénovaciu množinu budú reálne odtlačky prstov, ktorých je len malé množstvo. Táto databáza bude poskytnutá výskumnou skupinou STRaDe. Reálne odtlačky prstov s jednotlivými prejavmi môžu zlepšiť výsledky detekcie a klasifikácie. Poslednou časťou, ktoré budú tvoriť zdroj do databázy, je množina odtlačkov prstov bez ochorení alebo inými poškodeniami.



(a) Bradavica. (b) Dyshidróza. (c) Iné poškodenie. (d) Zdravá oblasť

Obrázok 4.2: Príklad vstupných výrezov. Zdroj: Interná databáza skupiny STRaDe.

Pri príprave trénovacej množiny je veľmi dôležité pokrytie čo najväčšieho množstva ochorení, zdravých oblastí alebo iných poškodení ako je napríklad bežné defekty pri snímaní. Na nasledujúcom obrázku 4.2 sú zobrazené jednotlivé druhy, ktoré neurónová sieť môže stretnúť. Neurónová sieť, ktorá bude trénovaná neuvidí komplexne celý odtlačok prsta ako je to zobrazené na obrázku 4.2. Uvidí len jeho časť a navyše obrázok bude normalizovaný pre vstupnú vrstvu.

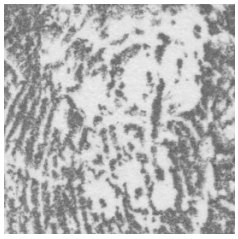
Ďalším z dôležitých krokov pri príprave trénovacej množiny je označenie výskytu ochorenia na odtlačku prsta. V prípade reálnych odtlačkov sa bude jednať o manuálne označovanie, v prípade syntetických je pri vygenerovaní chorôb aj označenie, kde sa dané poškodenie nachádza.

4.3.1 Trénovacia množina pre poškodenia pri snímaní

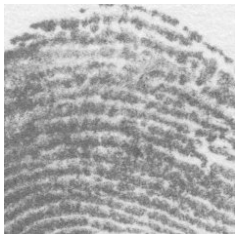
Podobne ako pri trénovacej množine pre ochorenia, je potrebné čerpať z viacerých zdrojov. Pri poškodeniach odtlačkov prstov pri snímaní budú najväčšiu časť tvoriť syntetické odtlačky prstov.

Doplnené budú reálnymi odtlačkami, ktoré majú nejaký defekt pri snímaní, no týchto odtlačkov bude obmedzené množstvo. Posledú časť bude tvoriť skupina zdravých oblastí odtlačkov prstov ako aj vyššie uvedené odtlačky prstov pre ochorenia bradavice a dyshidrózy. Na nasledujúcom obrázku 4.3 je vidieť príklad odtlačkov prstov.

(a) Rôzny tlak a vlhkosť.



(b) Dyshidróza.



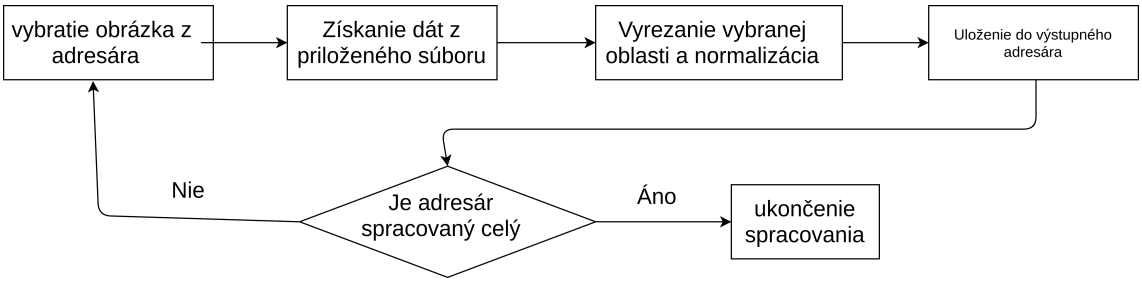
(c) Zdravá oblasť

Obrázok 4.3: Príklad vstupných výrezov. Zdroj: Interná databáza skupiny STRaDe a vygenerovaný odtlačok z SFinGe.

4.4 Návrh datasetu pre ochorenia a poškodenia pri snímaní

Návrh datasetu vychádza z bakalárskej práce [66], kde sa používa generátorová funkcia. Táto funkcia pristupuje do vytvorenej databázy a vyberá z nej položky podľa potreby. Ďalej je uvedené, že to šetrí operačnú pamäť za cenu zdržania samotného učenia, pretože úpravy sa robia za behu učenia. [66] Prístup je vhodný, pokiaľ sa nevykonávajú rôzne predspracovania alebo obrázkov na trénovanie nie je veľa. Nový návrh datasetu počíta so zmenou predspracovania, a tým pádom bude treba prehodiť jednotlivé fázy. Myšlienka generátorov, ktoré by sa starali o výber obrázkov a ich spracovanie, bude v novom návrhu zachovaná.

Obrázky s odtlačkami budú načítavané z adresára, ktorý používateľ nadefinuje. V danom adresári budú adresáre pre odtlačky s chorobou dyshidrózy a bradavice. Ďalšími adresármi búdu odtlačky s poškodeniami pri snímaní, ktoré sú tlak a vlhkosť. V poslednom adresári budú odtlačky s čistými oblasťami alebo inými ochoreniami a poškodeniami. Každý obrázok, ako sa spomína v bakalárskej práci [66], bude musieť byť sprevádzaný istými informáciami v metadatách. Obrázky, ktoré nebudú obsahovať sprievodné metadáta, budú vylúčené, aby to neovplyvnilo dataset.



Obrázok 4.4: Vývojový diagram prípravy dát.

Samotný princíp spracovania v generátore bude prebratý z bakalárskej práce [66]. Na obrázku 4.4 je vidieť tento princíp. Princípom je, že sa z metadát spracuje a určí oblasť, kde sa nachádza ochorenie. Pre zdravé oblasti bude súbor s metadátami prázdny. V tom prípade sa zoberie náhodná časť a označí sa ako nepoškodená. Oblasť sa označí, vyreže, odstráni sa okolie, zvýšia sa kontúry a použije sa adaptívne prahovanie ako to je v bakalárskej práci [66]. Uvažuje sa aj o inom predspracovaní ako je Gáborov filter. Po tejto fáze sa obrázky upravia na vstupnú vrstvu a uložia sa. Posledná fáza bude rozdielna od bakalárskej práci [66]. Tento rozdiel je o vytvorení dočasných súborov, ktoré budú môcť byť odstránené po trénovaní sieti. Tento spôsob nám vylepší rýchlosť učenia neurónovej siete a zároveň tento proces sa môže paralelizovať pre jednotlivé adresáre a znížiť tak fázu tvorby datasetu.

V podkapitole 2.5 bol popísaný proces rozpoznávania odtlačku prsta. Pri tomto procese je spomínané vylepšenie obrazu. Z bakalárskej práce [66] vyplýva, že je použité iba adaptívne prahovanie, ktoré sa používa pre binarizáciu. Návrh uvažuje o použití Gáborovho filtra pre vylepšenie obrazu. Použitie tohto filtru môže pomôcť pri lepšej detekcii bradavíc. Ďalším z mnohých vylepšení obrazu je použiť princípy orientačných polí, ktoré sú spomínané v ďalšej bakalárskej práci [2]. Pomohlo by to pri učení, aby sa lepšie označili oblasti, ktoré sú dôležité pri učení.

4.5 Detekcia poškodenia odtlačkov

V rámci práce bude vytvorená nová klasifikácia a detekcia poškodenia odtlačku prstu pri snímaní. Pozorovanými javmi pri poškodení odtlačku prstu pri snímaní bude tlak, skreslenie a vlhkosť, ktoré boli popísané v podkapitole 2.7. Návrh na vytvorenie nového klasifikátora sa môže pozerať z dvoch pohľadov. Každý z týchto pohľadov má kladné aj záporné stránky. Týmito pohľadmi sú:

- Vytvorenie nového klasifikátora.
- Detekciu a klasifikáciu pre poškodenie odtlačku pri snímaní zakomponovať do vylepšenej konvolučnej siete.

4.5.1 Nový klasifikátor

Jednou z prvých možností je vytvoriť nový klasifikátor. Nový klasifikátor bude s najväčšou pravdepodobnosťou vychádzať z návrhu architektúry pre zlepšenie aktuálneho riešenia. Výhodou bude, že sa nebude musieť vymýšlať nanovo architektúra pre klasifikátor, ale pužije sa základ architektúry už niečo, čo je vyvinuté a odladené. Nevýhodou tohto prístupu je, že bude tretia neurónová sieť. Táto sieť by bola jednovrstvová a skladala by sa zo sigmoid modelu, ktorý je popísaný v podkapitole 3.6.3. Ďalšou z mnohých nevýhod je, že siete sa budú učiť oddelene a potom tretia vrstva nad nimi rozhodne, ktorý výstup sa použije. Toto môže mať za následok zväčšeného falošne detekovného obrazu. Alternatívou k vytvoreniu jednoduchej sieti je vytvorenie obyčajného rozhodovacieho algoritmu, ktorý dostane infmácie zo sietí a následne rozhodne. Táto možnosť bude mať s najväčšou pravdepodobnosťou vyšší počet falošne detekovaného obrazu z obrázkov ako použitie jednoduchej neurónovej siete. Alternatívou celého nového klasifikátora je použitie vylepšenej neurónovej siete z návrhu.

4.5.2 Zakomponovanie do vylepšeného riešenia

Alternatíva k vytvoreniu nového klasifikátora a zloženie neurónovej siete z dvoch neurónových sietí, je zakomponovať detekciu a klasifikáciu do vylepšeného riešenia popísaného

v podkapitole 4.2. Výhodou tohto riešenia je, že odpadne vytvorenie nového klasifikátora a jednovrstovej neurónovej siete. Ďalšou výhodou je, že výstupná vrstva používa softmax funkciu. Ďalšou z mnohých výhod môže byť lepšia klasifikácia, keďže sa bude učiť ochorenia a poškodenia zároveň.

Pri tomto type riešenia sa práca viac nebude zaoberať binárnou klasifikáciou, ale už len viactriednou klasifikáciou. Pre viactriednu klasifikáciu sa používajú funkcie, ktoré sú popísané v podkapitole 3.4.3. Pri návrhu riešenia uvažovanou funkciou bude viac triedna krížová entropia alebo kategorická krížová entropia.

Nevýhodou tohto riešenia z pohľadu návrhu môže byť, že bude treba zmeniť počet konvolučných blokov. Ďalším aspektom môže byť ďalšia zmena optimalizátora po vylepšení riešení, ktoré je popísané v podkapitole 4.2.3. Všetky tieto aspekty sa uvidia pri implementácii alebo sa môže zistiť, že zakomponovanie nového klasifikátora do vylepšeného riešenia prinieslo väčší zisk ako bolo riziko pri návrhu.

Rozhodnúť, či vytvorenie nového klasifikátora alebo zakomponovanie do vylepšeného riešenia sa uvidí pri samotnej implementácií. Určite stojí za vyskúšanie implementovať obidve riešenia. Následne ich otestovať a porovnať na rovnakých vstupných databázach, kde sa určí, ktorý spôsob bude vo výsledku lepšie detekovať.

4.6 Výstup detekcie poškodenia odtlačkov prstov

Pre výstup z detekcie sa bude používať upravené implementované riešenie z bakalárskej práce [66]. Toto riešenie implementuje prechod celého obrázka, kde na konci vytvorí výstupný súbor vo formáte JSON (*JavaScript Object Notation*). V tomto súbore budú zaznačené oblasti jednotlivých poškodení, ktoré klasifikátor bude detekovať. Formát súboru bude upresnený pri dobe implementácií, ale budú tam informácie názov poškodenia pri snímaní, oblasť výskytu a pravdepodobnosť výskytu. Praktickejšie pre používateľa je vykreslenie týchto oblastí do obrázka. Pri tomto kroku v implementácii dôjde k zmene, ale podstata zostane zachovaná. Tou podstatou je vykreslenie poškodenia do obrázku a zobraziť ho používateľovi.

Kapitola 5

Implementácia

Nasledujúca kapitola popisuje implementáciu návrhu riešenia. Kapitola je rozdelená do viacerých podkapitol. Prvá podkapitola 5.1 popisuje framework PyTorch, platformu TensorFlow a API Keras. Podkapitola rozoberie samostatne tieto možnosti s rozdielmi. Druhá podkapitola 5.2 bude venovaná použitým prostrediam a komponentám. Nasledujúca podkapita 5.3 bude rozoberať implementáciu prepisu z knižnice Keras do knižnice PyTorch. Ďalšou podkapitolou 5.4 bude popísaná implementácia predspracovania pomocou Gáborovho filtra. Podkapitola 5.5 sa zaoberá prípravou anotácie čistých odtlačkov prstov. Následne v podkapitole 5.6 bude rozobratá príprava datasetu a v podkapitole 5.7 implementácia prípravy datasetu. Podkapitola 5.8 popisuje trénovanie konvolučnej neurónovej siete. Táto podkapitola rozoberie implementáciu trénovania, ktorá bude využívať návrh experimentálnych vylepšení, ktoré sú v podkapitole 4.2. Predposledná podkapitola 5.9 popíše implementáciu klasifikátora. V poslednej podkapitole 5.10 bude popísaná implementácia prípravy testovania.

5.1 Možnosti implementácie konvolučnej siete

Podkapitola sa bude zaoberať API Keras, ktorá bola použitá v bakalárskej práci [66]. Budú popísané základné rysy a štruktúra. Následne bude popísaný framework PyTorch, ktorý je využitý pri implementácii diplomovej práci. Tiež budú popísané základné rysy a štruktúra. Nakoniec v krátkosti budú zhrnuté hlavné rozdiely, a prečo sa v diplomovej práci využíva framework PyTorch.

5.1.1 TensorFlow a API Keras

Keras sa na mnohých rôznych serveroch označuje ako knižnica ale podľa oficiálnej stránky [12] je to API nad platformou TensorFlow. Keras je podporovaný firmou Google a jeho prvé vydanie bolo v marci 2015. Keras je vyvinutý v jazyku Python. API Keras umožňuje trénovanie ako na procesore, tak aj na grafických kartách. Keras je zameraný hlavne na tvorbu konvolučných a rekurzívnych sietí. Jedná sa o vysoko úrovňové API, ktoré sa hodí pre rýchle prototypovanie a experimentovanie. Jej základ tvorí model. Jedná sa o objekt, v ktorom sú združené inštancie tried pre všetky vrstvy. [12]

Keď sa akceptovateľný model skompiluje, tak sa môže prejsť na fázu trénovania. Trénovanie sa zavolá pomocou funkcie ***fit()*** . Pri trénovaní sa vytvoria dve množiny vstupná a výstupná. Po trénovaní je možné tento model uložiť. Uložený model alebo po trénovaní je

možné zavolať funkciu ***evaluate()***, pomocou ktorej je možné detekovať obrázky naučenou sieťou. [12]

5.1.2 PyTorch

Podobne ako Keras, tak aj PyTorch je nesprávne na mnohých serveroch označovaný ako knižnica. Podľa oficiálnych stránok [23] je to framework. PyTorch je podporovaný spoločnosťou Facebook. Jeho prvá verzia bola vydaná v septembri 2016. Jedná sa o framework, ktorý využíva nižšiu úroveň API. [23]

Jeho základným rysom je, že využíva triednu definíciu modelu. V praxi to znamená, že keď sa chce vytvoriť vlastný model, musí byť vytvorený ako trieda. Po vytvorení modelu je možné pomocou metódy ***cuda()*** použiť pre trénovanie alebo detekciu grafickú kartu. V opačnom prípade bude využitý procesor. [23]

Ak sa chce model použiť pre trénovanie, tak sa nad modelom zavolá metóda ***train()***. Ak sa chce model vyhodnocovať, tak sa nad modelom zavolá metóda ***eval()***. V PyTorch sa vstupná množina vytvára pomocou triedy ***DataLoader***. PyTorch má vstavaných niekoľko tried s datasetmi. Pre vytvorenie vlastného datasetu sa musí vytvoriť trieda, ktorá má predka, napríklad ***Dataset*** [23]

5.1.3 Rozdiely

V tejto podkapitole sa rozoberú rozdiely medzi API Keras a PyTorch. Prvým rozdielom je prístup k modelu. Ako bolo už vyššie spomenuté, tak Keras model obsahuje v sebe inštancie vrstiev. Naproti tomu v PyTorch sa vytvára model ako trieda. Ďalším rozdielom je, že Keras je vysoko úrovňové API, ale PyTorch je vyvinuté ako nižšie úrovňové API. Vďaka tomu je viac možností pri nastavovaní trénovania ako aj pri detekcii. To však prináša aj o niečo dlhšiu učiacu krivku.

Podľa článku [52] je používanejší framework PyTorch pre vedecké výskumy. Zároveň popisuje, že Keras má vyborný tutoriál a znovu použiteľný kód. Namiesto toho PyTorch má aktívny vývoj a výbornú podporu pomocou komunity. Ale najdôležitejšie z toho článku je, že PyTorch je rýchlejší pre trénovanie ako Keras.

Ďalším kladom pre framework PyTorch je stálosť volania funkcií. Namiesto toho pri API Keras sa často stáva, že názov funkcie je zmenený alebo má iné parametre. Framework PyTorch fungujestabilnešjie ako Keras.

Z posledného uvedeného rozdielu z článku [52] je pre diplomovú prácu vybraný framework PyTorch. Bol vybraný aj z dôvodu stáleho vývoja a možnosti pokročilejšieho nastavovania pri trénovaní.

5.2 Prostredie a použité komponenty

Implementácia aplikácie je konzolová. Aplikácia je napísaná v jazyku Python vo verzii 3.8, ktorá využíva objektovo orientované programovanie s využitím princípu *DRY (Don't repeat yourself)*. Aplikácia bola implementovaná vo vývojovom prostredí PyCharm. Použité závislosti na knižniciach sú napísané v *requirements.txt*, aby sa dali ľahko doinštalovať. Na nasledujúcom zozname sú popísané knižnice a k čomu sa využívajú:

- *torch (verzia 1.7.0)* knižnica PyTorch, ktorá sa používa na trénovanie a pre správu neurónovej siete.

- *torchvision (verzia 0.8.1)* podporná knižnica k *torch*, na pracovanie s obrázkami a transformácie pred vložením obrázka do siete.
- *opencv-python (verzia 3.4.8.29)* OpenCV knižnica, ktorá sa používa pri pracovaní s obrázkami.
- *numpy (verzia 1.20.2)* knižnica Numpy sa používa na prácu s maticami.
- *pandas (verzia 1.2.4)* Pandas je knižnica, ktorá sa v diplomovej práci využíva na vytvorenie anotácie a načítanie do datasetu pre knižnicu PyTorch.
- *pillow (verzia 8.2.0)* knižnica Pillow na prácu s obrázkami, ktorá sa využíva pri implementácii vlastného datasetu.
- *matplotlib (verzia 3.3.4)* knižnica Matplotlib sa používa na vykreslenie ochorenia alebo poškodenia po detekcii.

Pre testovanie trénovania bol použitý osobný počítač, ale konečné trénovanie a testovanie prebiehalo na cloudovom riešení od spoločnosti MetaCentrum. V nasledujúcej podkapitole bude stručne popisané cloudové riešenie MetaCentrum.

5.2.1 MetaCentrum VO

Jedná sa o virtuálnu organizáciu, ktorá združuje akademických pracovníkov, vedcov, študentov, akademických zamestancov alebo vedecké inštitúcie po celej Českej republike. [51]

Organzácia, ktorá umožnuje využiť heterogénny *GRID* pre rôzne procesorové alebo grafické náročné výpočty. Tento Grid sa skladá z rôznych uzlov ako napríklad Praha, Ostrava, Brno, Plzeň alebo Olomouc. Vďaka týmto uzlom, je možné využiť rôzne výkonné stroje s rôznymi počtami procesorov, ramiek, voľného priestoru a grafických kariet.

Pre používanie MetaCentra sa musí človek najskôr registrovať. Registrácia je možná pomocou *eduId*. Následne odsúhlasí licenčné podmienky a môže využívať tieto výpočtové prostriedky.

Pre využitie týchto prostriedkov je nutné požiadať pomocou plánovacieho systému úloh PBS (Portable Batch System). Tento systém funguje na interaktívnom alebo neinteraktívnom režime. Ak používateľ použije interaktívny režim, potom pri pridelení prostriedkov a času môže do konzoly zadať jednotlivé príkazy k výpočtu. Druhou možnosťou je využiť neinteraktívneho režimu, teda spustiť dávkový súbor, v ktorom sú napísané jednotlivé príkazy pre výpočet.

Nasledujúci algoritmus 5.1 zobrazuje PBS dávku pre detekciu neurónovou sieťou z bakalárskej práce.

```
#!/bin/bash
#PBS -N PyTorch_Job
#PBS -q gpu
#PBS -l select=1:ncpus=1:mem=32gb:scratch_local=10gb
#PBS -l ngpus=1:gpu_cap=cuda61:cuda_version=11.2:cpu_flag=avx512dq
#PBS -l walltime=0:15:00

DATADIR=/storage/brno3-cerit/home/xvican02/
cp -r $DATADIR/salko_bp/ $SCRATCHDIR
cp $DATADIR/salko_bp/weights-improvement-output3-epoch-07-val-0.93.hdf5 $SCRATCHDIR
cd $SCRATCHDIR

module add py-numpy/py-numpy-1.19.0-intel-19.0.4-m6yrhiu
module add py-keras-applications/py-keras-applications-1.0.8-intel-19.0.4-ej45cy3
```

```
module add py-keras-preprocessing/py-keras-preprocessing-1.1.0-intel-19.0.4-cm4kf4i
module add opencv-3.4.5-py36
module add tensorflow-1.13.1-gpu-python3
module add hdf5-1.8.14-gcc

python3.7 salko_bp/main.py --debug -f salko_bp/PREDICT/dysh/FP_00507.png >detect.out 2> detect.err
cp detection.out detection.err $DATADIR
cp -r salko_bp/output/ $DATADIR || { echo >&2 "Result file(s) copying failed (with a~code $?) !!"; exit 4; }
clean_scratch
```

Algoritmus 5.1: Dávkový súbor pre detekciu z BP.

Pre potreby učenia pomocou knižnice PyTorch, musí byť dávkový súbor upravený. Úprava je z dôvodu, že sa využívajú oficiálne kontajnery, kde je virtuálne rozbehnutý operačný systém Ubunta s danou verziu knižnice PyTorch od spoločnosti NVIDIA. Upravený dávkový súbor je zobrazený na nasledujúcom kóde 5.2:

```
#!/bin/bash
#PBS -N Train_PyTorch
#PBS -q gpu
#PBS -l select=1:ncpus=1:mem=32gb:scratch_local=15gb:ngpus=1:gpu_cap=cuda61:cuda_version=11.2
#PBS -l walltime=5:00:00
#PBS -m ae

singularity run --nv /cvmfs/singularity.metacentrum.cz/NGC/opencv-NGC-PyTorch21.02.SIF \
                    /storage/brno2/home/xvican02/first_train/train_model.sh
```

Algoritmus 5.2: Dávkový súbor pre učenie v knižnici PyTorch.

V kóde 5.2 je možné vidieť, že sa volá *train_model.sh*. Tento skript vo vnútri obsahuje, ako sa má volať neurónová sieť.

5.3 Prepis modelu z Keras do PyTorch

Podkapitola sa bude zaoberať prepisom modelu z bakalárskej práce [66]. Model aj celá konvolučná neurónová sieť je napísaná v knižnici Keras. Diplomová práca využíva knižnicu PyTorch, rozdiely medzi knižnicami sú popísané v podkapitole 5.1.3. Celý kód Keras modelu sa nachádza v prílohe B a kód Pytorch modelu v prílohe C. Na následujúcich ukážkach kódov budú ukázané dôležité zmeny, ktoré museli byť vykonané.

Prvou zásadnou zmenou je, že PyTorch využíva triedny prístup k vytvoreniu modelu. Toto je možné si všimnúť na algoritme 5.4 a definovanie funkcie na získanie modelu v knižnici Keras na algoritme 5.3.

```
def create_model(debug=True, number_of_outputs=2, shape=(128, 128, 1)):
  model = Sequential()
```

Algoritmus 5.3: Funkcia na vytvorenie modelu.

```
class FingerprintModelBP(nn.Module):
  def ___init___(self, number_of_outputs, shape):
    super(FingerprintModelBP, self).___init___()
```

Algoritmus 5.4: Trieda pre model.

2D konvolučná vrstva sa v knižnici Keras definuje tak, ako je to zobrazené na algoritme 5.5. Z tohto algoritmu je vidieť, že táto konvolučná vrstva definuje vstupný obrázok. V knižnici Pytorch sa vstupný obrázok dáva ako Tenzor, teda v konvolučnej vrstve nedefinuje vstupný rozmer obrázka. Rozdiel medzi implementáciou 2D konvolučnej vrstvy je

možné vidieť na algortime 5.5 a 5.6. V knižnici PyTorch sa konvolučná sieť skladá v metóde forward, ako je to zobrazené na algoritme 5.6.

```
def create_model(debug=True, number_of_outputs=2, shape=(128, 128, 1)):
  ...
  model.add(Conv2D(64, (3, 3), activation='relu', input_shape=shape, data_format="channels_last"))
```

Algoritmus 5.5: 2D konvolučná vrstva v Keras.

```
class FingerprintModelBP(nn.Module):
  def __init__(self, number_of_outputs, shape):
   self.conv1 = nn.Conv2d(1, 64, (3, 3))
   self.relu = nn.ReLU()

  def forward(self, model):
    model = self.relu(self.conv1(model))
```

Algoritmus 5.6: 2D konvolučná vrstva v PyTorch.

Ďalším rozdielom je prepis *max-pooling* vrstvy. Tento rozdiel je popísaný na algoritmoch 5.7 a 5.8.

```
def create_model(debug=True, number_of_outputs=2, shape=(128, 128, 1)):
  ...
  model.add(MaxPooling2D((2, 2), strides=(2, 2)))
```

Algoritmus 5.7: Max-pooling vrstva v Keras.

```
class FingerprintModelBP(nn.Module):
  def __init__(self, number_of_outputs, shape):
    self.pool = nn.MaxPool2d(2, 2)

  def forward(self, model):
    model = self.pool(model)
```

Algoritmus 5.8: Max-pooling vrstva v PyTorch.

V knižnici Keras sa na zjednodušenie siete používa *Flatten* vrstva. Toto je zobrazené na algoritme 5.9.

```
def create_model(debug=True, number_of_outputs=2, shape=(128, 128, 1)):
  ...
  model.add(Flatten())
```

Algoritmus 5.9: Flatten vrstva v Keras.

V PyTorch sa to definuje ako zmena pohľadu (view), ktorá má dva vstupné parametre. Jej vstupné parametre sú veľkosť modelu a -1. Parameter -1 hovorí, že sa to má roztiahnuť. Toto je vidieť na algortime 5.10.

```
class FingerprintModelBP(nn.Module):

  def forward(self, model):
    model = model.view(model.size(0), -1)
```

Algoritmus 5.10: Flatten vrstva v PyTorch.

Dropout vrstva sa prepíše veľmi ľahko. Sú veľmi podobné zápisy až na zloženie pomocou ReLu aktivačnej funkcie. Tento prepis je ukázaný na algoritme 5.11 a 5.12.

```
def create_model(debug=True, number_of_outputs=2, shape=(128, 128, 1)):
  ...
  model.add(Dropout(0.5))
```

Algoritmus 5.11: Dropout vrstva v Keras.

```
class FingerprintModelBP(nn.Module):
  def ___init___(self, number_of_outputs, shape):
    self.dropout = nn.Dropout(0.5)

  def forward(self, model):
    model = self.dropout(model)
```

Algoritmus 5.12: Dropout vrstva v PyTorch.

Najzložitejšie sa prepisuje *Dense* vrstva, ktorá celú sieť zjednodušuje do jedného čísla. Implementácia Dense vrstvy v knižnici Keras je zobrazená na algoritme 5.13. V knižnici PyTorch, je to zložitejšie, pretože sa používa ako lineárna funkcia. Lineárna funkcia má dva parametre. Prvý parameter je vstupná hodnota. V tejto hodnote treba zobrať do úvahy šírku a výšku obrázka. Tieto parametre sa vynásobia spolu s predchadzajúcou výstupnou hodnotou. Druhý parameter je výstupná hodnota. Prepis tejto vrstvy do knižnice PyTorch je zobrazený na algoritme 5.14.

```
def create_model(debug=True, number_of_outputs=2, shape=(128, 128, 1)):
  ...
  model.add(Dense(768, activation='relu'))
```

Algoritmus 5.13: Dense vrstva v Keras.

```
class FingerprintModelBP(nn.Module):
  def ___init___(self, number_of_outputs, shape):
    self.relu = nn.ReLU()
    self.linear1 = nn.Linear(256 * 4 * 4, 768)

  def forward(self, model):
    model = self.relu(self.linear1(model))
```

Algoritmus 5.14: Dense vrstva v PyTorch.

Vo funkcii pre vytvorenie modelu sa v knižnici Keras nakoniec kompiluje model so stratovou funkciou a optimalizátorom. Toto je zobrazené na algoritme 5.15. To sa v knižnici PyTorch robí až pri tvorení algoritmu učenia. Na algoritme 5.16 je vidieť ako sa zapisuje stratová funkcia a optimalizátor v PyTorch.

```
def create_model(debug=True, number_of_outputs=2, shape=(128, 128, 1)):
  ...
  opt = SGD(learning_rate=0.001, momentum=0.0, nesterov=False)
  model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=["accuracy"])
```

Algoritmus 5.15: Stratová funkcia a optimalizátor v Keras.

```
def train():
  cnn_net = FingerprintModelBP(3, (1, 128, 128))
  criterion = nn.CrossEntropyLoss()
  optimizer = optim.SGD(cnn_net.parameters(), lr=0.001, momentum=0.0)
```

Algoritmus 5.16: Stratová funkcia a optimalizátor v PyTorch.

5.4 Implementácia predspracovania

V podkapitole 4.4 sa rozoberalo, aké možnosti predspracovania je možné použiť. V implementácii je použitý Gáborov filter. Jeho implementácia vychádza z rovnice pre 2D Gáborovu funkciu [26]. V algoritme 5.17 je možné vidieť implementáciu reálnej časti pre Gáborov filter.

```
def ___apply__gabor__equation__to__each__value(self, half__kernel__size, angle):
  gabor__equation = np.zeros((self.___KERNEL__SIZE, self.___KERNEL__SIZE), dtype=np.float32)

  for y in range(self.___KERNEL__SIZE):
    for x in range(self.___KERNEL__SIZE):
      kernels = self.___get__width__height__kernels(angle, x − half__kernel__size, y − half__kernel__size)

      real__part__exp = −(kernels[0] ** 2 + self.___GAMMA ** 2 * kernels[1] ** 2) / (2 * self.___SIGMA ** 2)
      real__part__cos = 2 * np.pi * kernels[0] / self.___LAMBDA + self.___PSI
      gabor__equation[y, x] = np.exp(real__part__exp) * np.cos(real__part__cos)

  return gabor__equation / np.sum(np.abs(gabor__equation))

def ___get__width__height__kernels(self, angle, distance__x, distance__y):
  angle__in__radian = self.___convert__degree__to__radian(angle)

  x__kernel = np.cos(angle__in__radian) * distance__x + np.sin(angle__in__radian) * distance__y
  y__kernel = −np.sin(angle__in__radian) * distance__x + np.cos(angle__in__radian) * distance__y
  return x__kernel, y__kernel
```

Algoritmus 5.17: Metódy pre reálnu časť Gáborovho filtra.

Gáborov filter je implementovaný ako trieda. Z algoritmu 5.18 je možné vidieť dva povinné parametre v konštruktore. Prvým je *count*, ktorý značí koľkokrát sa má aplikovať Gáborov filter. Nasledujúci parameter *angle* hovorí, o koľko v každom kroku sa posunie uhol výpočtu Gáborovho filtra. V implementácii sú nastavené tieto parametre na hodnoty šesť a tridsať stupňov.

```
def ___init___(self, count, angle):
  self.___count = count
  self.___angle__rotation = angle
```

Algoritmus 5.18: Konštruktor triedy s Gáborovým filtrom.

Nad inštanciou triedy sa zavolá metóda *image__after__gabor*, kde je jeden parameter a to je obrázok do Gáborovho filtra.

5.5 Príprava anotácie čistých odtlačkov prsta

Príprava anotácie čistých odtlačkov prstov bolo nutné implementovať z dôvodu predprípravy komplet celého datasetu. Anotácia je tiež ako Gáborov filter implementovaná ako trieda. Z konštruktoru 5.19 je vidieť, že sa posiela cesta k databáze. V konštruktore sa k ceste pridá zložka *clear* v ktorej sú odtlačky prsta vybrané bez ochorení. V druhej privátnej triednej premennej sa nachádza cesta s anotovanými odtlačkami prsta.

```
def ___init___(self, path):
  self.___directory = path + "clear/"
  self.___marked__clear = path + "marked__clear"
```

Algoritmus 5.19: Konštruktor triedy pre anotovanie.

Anotovanie sa pustí pomocou zavolania metódy *mark__clear__fingerprint*. Samotné anotovanie funguje tak, že sa prechádza každým obrázkom v priečinku. Nad každým obrázkom sa vygeneruje päť pseudonáhodných výrezov z obrázka a ich súradnice s označením, že je to *clear* sa uloží do json súboru a obrázok sa prekopíruje. Toto anotovanie je zobrazené na algoritme 5.20

```
class ClearAnnotation:
  ...
  def ___generate_values(self, max_value):
    while True:
      axe_min = rand.randint(self.___MIN_VALUE, max_value)
      axe_max = rand.randint(self.___MIN_VALUE, max_value)

      if self.___MIN_DISTANCE < axe_max − axe_min < self.___MAX_DISTANCE:
        break
    return axe_min, axe_max

  def ___generate_pseudo_random_parts(self, width, height):
    dataset = []

    for i~in range(5):
      x_min, x_max = self.___generate_values(width)
      y_min, y_max = self.___generate_values(height)
          dataset.append({"Type": "clear", "p1": [x_min, y_min], "p2": [x_max, y_max]})
      return dataset

  def mark_clear_fingerprint(self):
      ...
      clear_images = os.listdir(self.___directory)
      count = 0

      for file in clear_images:
        print("Marking clear fingerprint... ", int((count / len(clear_images)) ∗ 100), "%")

        file_name, _ = file.split('.')
        image = cv2.imread(self.___directory + file)
        height, width, _ = image.shape
        target_file_name = self.___marked_clear + "/" + file_name + ".json"

        with open(target_file_name, 'w') as out_file:
          json.dump(self.___generate_pseudo_random_parts(width, height), out_file)

        cv2.imwrite(self.___marked_clear + "/" + file_name + ".png", image)
        count += 1
```

Algoritmus 5.20: Anotovanie čistých odtlačkov prstov

5.6 Príprava databázy

Databáza odtlačkov prstov je najdôležitejšia súčasť pri trénovaní konvolučnej neurónovej siete. V podkapitole 5.5 bolo spomenuté anotovanie odtlačkov prstov, ktoré nemajú žiadne ochorenie. Kvôli tomu, že databáza reálnych odtlačkov prstov s ochoreniami je veľmi malá, tak sa používajú aj syntetické odtlačky prstov, ktoré boli spomenuté v podkapitole 2.6.

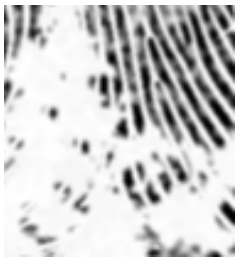
Prvou fázou bolo vygenerovanie odtlačkov prstov pomocou nástroju SFinGe. Tento nástroj vygeneroval, čo najviac reálne vypadajúce nasnímané odtlačky prstov. Do týchto odtlačkov prstov boli vygenerované pomocou generátora ochorenia dyshidróza a bradavice. Tento generátor bol napísaný v rámci bakalárskej práce [66]. Pri trénovaní sa použila aj databáza s reálnymi odtlačkami s danými ochoreniami. Tento typ databázy je využívaný pre experimentálne vylepšenie stávajúcej konvolučnej neurónovej siete.

Po experimentálnom vylepšení sa pridala databaza s reálnymi odtlačkami pre psoriázu. Pridanie databázy so psoriázou znamená rozšírenie klasifikátora o ďalšie rozpoznávanie. Databáza je takto zostavená, aby sa otestovalo ako ovplyvní takáto malá zmena celý klasifikátor. Ďalej do databázy boli pridané ďalšie odtlačky prsta spôsobené poškodením tla-

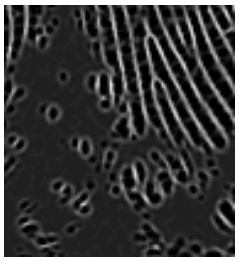
kom. Toto poškodenie sa generovalo generátorom, ktorý bol vývinutý vo výskumnej skupine STRaDe.

5.7 Implementácia vytvorenia datasetu

Implementácia vytvorenia datasetu vychádza z generátora množiny *TLMaker.py*, ktorý slúžil v bakalárskej práci [66] na vygnerovanie datasetu. Ale ako bolo zmienené v podkapitole 4.4, tak dataset sa vytvorí pred trénovaním siete. Táto zmena je vykonaná kvôli optimalizácii pri učení siete. Pre tieto účely slúži skript *set_up.py*, kde sa pomocou parametrov určí, aký dataset sa má vytvoriť.



(a) Orezaná oblasť.



(b) Aplikovaný Gáborov filter.

(c) Aplikácia rozmazania a adaptívneho prahovania.

Obrázok 5.1: Výrez vygenerovaného ochorenia dyshidrózy po úpravách do datasetu.

Tento skript zavolá inštanciu triedy *PrepareDataset*, ktorá sa stará o orezanie obrázku pre dataset *TRAIN* a *VALID*. Následne sa použije na takto orezaný obrázok Gáborov filter, ktorý bol popísaný v podkapitole 5.4. Následne sa na takýto obrázok aplikuje rozmazanie a nakoniec sa použije adaptívne prahovanie. Na obrázku 5.1 je možné vidieť vyššie zmienené úpravy.

Do priečinku *PREDICT* ukladá celé obrázky aj s json súbormi, kde sa nachádzajú ochorenia, aby sa dalo otestovať ako sa kvalitne naučila neurónová sieť. Testovanie a výsledky sú popísané v kapitole 6.

Po vytvorení všetkých potrebných datasetov, sa spustí skript *create_csv_database.py*, ktorý pre zložky TRAIN a VALID vygeneruje csv súbor. Tento súbor obsahuje relatívnu cestu k obrázku a jeho triedu, či sa jedná o čistú oblasť, alebo oblasť s niektorými druhmi ochorenia.

5.8 Implementácia trénovania

Nasledujúca podkapitola bude rozoberať implementáciu trénovania. V tejto podkapitole budú popísané ako sú v implementácii zakomponované jednotlivé experimentálne vylepšenia z podkapitoly 4.2.

Trénovanie je implementované v *pytorch_learn_cnn.py*. V tomto súbore je trieda, ktorá sa stará o celé trénovanie. Konštruktor má jeden povinný parameter a to *work_dir*, ktorým sa nastavuje pracovný adresár. V triede sú dve verejné metódy.

Prvou z nich je *update_parameters_from_arguments()*. Má jeden parameter a to sú argumenty z príkazovej riadky. Tieto argumenty sú získavané pomocou vstavanej knižnice *argparse* v súbore learn_cnn.py. V tomto súbore okrem parsovania je aj volanie vyššie

zmienenej triedy a funkcii. Pomocou argumentov sa dajú nastaviť rôzne nastavenia pre trénovanie. Pomocou argumentov sa dajú nastaviť experimentálne vylepšenia, a tak umožňujú zmeniť model pre trénovanie, aktivačnú funkciu pre model z bakalalárskej práce [66]. Pre použitie predtrénovaných modelov bolo nutné zmeniť prvú konvolučnú vrstvu, kvôli odtieňom sivej a výstup z modelu na požadovaný počet tried. Zároveň v argumentoch sa dá zmeniť počet výstupov z neurónovej siete, čím sa dá rozšíriť o učenie nových typov ochorení alebo poškodení pri snímaní.

Druhá verejná metóda sa volá *learn()*. Trénovací proces a jeho nastavenie je zložitejšie ako pri použití API Keras. V nasledujúcich podkapitolách bude rozobraté vytvorenie trénovacej databázy, validačnej databázy ich inicializácia nastavenie siete a nakoniec samotné trénovanie.

5.8.1 Vytvorenie načítania obrázkov z databázy a jej inicializácia

Ako bolo spomínané v podkapitole 5.1.2, tak PyTorch má vstavané databázy a ich načítanie, ale ak sa chce použiť vlastná databáza, tak je potrebné vytvoriť aj vlastné načítanie tejto databázy. Na nasledujúcom algoritme 5.21 je vidieť implementácia vlastného načítania databázy pre účely diplomovej práce. Z algoritmu 5.21 je vidieť, že konštruktor má dva povinné parametre a jeden voliteľny. Prvým parametrom je priečinok, kde sa nachádza csv súbor, ktorý je popísaný v podkapitole 5.7. Druhý parameter je názov csv súboru. Posledný parameter definuje transformácie nad obrázkom. Ďalej je možné si všimnúť, že trieda má ako predka Dataset, ktorý je tiež spomenutý v podkapitole 5.7.

```
class FingerPrintDataset(Dataset):
  def __init__(self, directory, annotation_file, transform=None):
    self.__directory = directory
    self.__annotations = pd.read_csv(directory + "/" + annotation_file)
    self.__transform = transform

  def __len__(self):
    return len(self.__annotations)

  def __getitem__(self, index):
    image_id = self.__annotations.iloc[index, 0]
    image = Image.open(os.path.join(self.__directory, image_id))
    class_type = self.__annotations.iloc[index, 1]
    if self.__transform is not None:
      image = self.__transform(image)
    return image, class_type
```

Algoritmus 5.21: Definícia vlastného načítania databázy.

Inicializácia trénovacej databázy je zobrazená na algoritme 5.22. Inicializácia validačnej databázy je zobrazená na algoritme 5.23. Z algoritmu 5.22 a 5.23 je zrejmé, že sa inicializujú rovnako, až na definovanie csv súboru, odkiaľ sa načítavajú trénovacie alebo validačné obrázky.

```
self.__valid_set = FingerPrintDataset(self.__path_valid, "train.csv", self.__transform(self.__BP_IMAGE_X))
self.__valid_loader = DataLoader(self.__valid_set, batch_size=self.__batch_size, shuffle=True,
num_workers=0)
```

Algoritmus 5.22: Inicializácia trénovacej databázy a jej načítania.

```
self.__valid_set = FingerPrintDataset(self.__path_valid, "valid.csv", self.__transform(self.__BP_IMAGE_X))
self.__valid_loader = DataLoader(self.__valid_set, batch_size=self.__batch_size, shuffle=True,
num_workers=0)
```

Algoritmus 5.23: Inicializácia validačnej databázy a jej načítania.

Transformácia je definovaná pomocou lambda funkcie, ktorá je zobrazená na algoritme 5.24. Lambda funkcia má jediný parameter a to veľkosť obrázka, ktorý je rozdielny pre model z bakalárskej práce [66], Inceptionv3 model z podkapitoly 3.10.3 a Resnet50 z podkapitoly 3.10.4

```
self.___transform = lambda x: Compose([RandomVerticalFlip(), RandomHorizontalFlip(),
                                        Resize((x, x)), ToTensor()])
```

Algoritmus 5.24: Transformácia pomocou lambda funkcie

5.8.2 Trénovanie

Pred samotným trénovaním sa musia nastaviť ešte aká stratová funkcia a optimalizátor bude použitý. Aby sa mohla sieť učiť na grafikcej karte, tak je zapotreby zistiť, či sa nachádza na stroji grafická karta. Ak áno tak sa sieť prepne na grafickú kartu ako to je znázornené na algoritme 5.25.

```
if torch.cuda.is_available():
  self.___cnn_model.cuda()
```

Algoritmus 5.25: Testovanie dostupnosti grafickej karty a zapnutie učenia na nej.

Samotné trénovanie je implementované tak, že prvý cyklus *for* hovorí koľko epôch sa vykoná. Sieť sa trénovala na sto epochách. Nasleduje trénovacia časť, kde sa načítajú dáta z načítania databázy. Tu sa musí tiež kontrolovať, či sa trénuje na grafickej karte, aby sa aj dáta a triedy prepli na grafickú kartu. Následne sa pošle dávka obrázkov do siete. Po dávke sa vypočíta strata trénovania. Implementovaná je kontrola učenia po definovanom počtu iterácii v epoche. V tejto validácii sa zistí, ako sa trénovaniu darí. Ak sa model lepšie naučil, tak sa zapíše slovník z definovanými váhami, epocha a úspešnosť. Algoritmus je implementovaný tak, že do súboru uloží len najlepší model. Tento učiaci proces je zobrazený na algoritme 5.26. Algoritmus je zjednodušený len na hlavné operácie. Inicializácia premenných, priebežné ukladanie modelu nebudú zobrazené.

```
for epoch in range(self.___epochs):
  ...
  self.___cnn_model.train()
  for batch_idx, (data, labels) in enumerate(self.___train_loader, 0):
    if self.___is_gpu_available:
      data, labels = data.cuda(), labels.cuda()

    optimizer.zero_grad()
    target = self.___cnn_model(data)
    loss = loss_function(target, labels)
    loss.backward()
    optimizer.step()
    train_loss += loss.item() * data.size(0)
    number_images += self.___batch_size

    if batch_idx % self.___valid_control == 0 and batch_idx != 0:
      ...
      self.___cnn_model.eval()
      for valid_batch_idx, (valid_data, valid_labels) in enumerate(self.___valid_loader, 0):
        if self.___is_gpu_available:
          valid_data, valid_labels = valid_data.cuda(), valid_labels.cuda()

        if self.___valid_steps == valid_batch_idx:
          break

          target = self.___cnn_model(valid_data)
```

```
        loss = loss_function(target, valid_labels)

        _, predicted = torch.max(target.data, 1)
        correct_valid += predicted.eq(valid_labels.data).sum().item()
        valid_loss = loss.item() * data.size(0)
        number_valid_images += self.___batch_size

    compute_validation_loss = valid_loss / number_valid_images
    compute_accuracy = correct_valid / number_valid_images

    if self.___saved_accuracy < compute_accuracy:
      ...

    self.___cnn_model.train()
    if self.___training_steps == batch_idx:
      break
```

Algoritmus 5.26: Trénovací proces.

Po trénovacom procese sa uloží model do súboru ako to bolo zmienené vyššie. Zároveň s modelom sa uložia informácie o modeli v json súbore. Tieto informácie sú použité pre načítanie modelu zo súboru. Na nasledujúcom algoritme 5.27 je zobrazený json súbor.

```
{
    "activation_function": 0,
    "bp_model": true,
    "number_of_outputs": 3,
    "pretrained_model": 0,
    "trained_gpu": true
}
```

Algoritmus 5.27: Informácie o uloženom modeli.

5.9 Implementácia klasifikátora

Implementácia klasifikátora je prebratá z bakalárskej práce [66]. Klasifikátor sa niektorými implementačnými detailmi líši ale podstata je rovnaká. V prvom rade klasifikátor je implementovaný ako samostatný skript. Skladá sa z dvoch súborov podobne ako implementácia trénovania. Prvý súbor je *predict_cnn.py*. V tomto súbore sa nachádza funkcia na parsovanie argumentov z príkazovej riadky. Následne sa tieto informácie prenesú do triedy *PredictCNN*, ktorá je v súbore *pytorch_predict_cnn.py*. Konštruktor triedy má 2 argumenty. Prvým argumentom je pracovný adresár a druhým je inštancia pre logovanie. Táto trieda obsahuje dve verejné metódy. Prvou je *update_arguments()*, ktorá ma jeden parameter a to argumenty z príkazovej riadky. Táto funkcia nastavuje adresár pre natrénovaný model, názov modelu a či sa bude predikovať súbor alebo celý priečinok. Druhou funkciou je *find_diseases()*. Ak sa nastavil súbor, tak vo funkcii sa zavolá iba *find_disease()* inak sa prechádza cyklom cez priečinok a volá sa postupne metóda *find_disease()*. Táto metóda obsahuje prebratú implementáciu z bakalárskej práce [66]. Implementácia musela prejsť drobnou úpravou, ktorá bude vysvetlená. V implementácii pribudlo po vykreslení kontúr implementované predspracovanie a to Gáborov filter. Popis implementácie Gáborovho filtra je popísané v podkapitole 5.4. Nastavenie Gáborovho filtra je rovnaké ako pri generovaní trénovacej a validačnej databázy. Algoritmus kĺzavého konvolučného okna je taktiež prevzatý z bakalárskej práce [66]. Ale aj tam sa museli spraviť určité zmeny, keďže klasifikátor je rozšírený o vyhľadávanie poškodenia pri snímaní. Ďalej bol klasifikátor rozšírený o vyhľadávanie psoriázy. Zároveň musel byť tento algoritmus adaptovaný na PyTorch framework.

Rovnako ako v bakalárskej práci [66], tak aj implementovaný klasifikátor ukladá obrázok, kde sa nachádza dané ochorenie plus json súbor s anotáciou ochorenia.

5.10 Implementácia testovania

Implementácia testovania je taktiež ako klasifikátor prebratá z bakalárskej práce [66]. Implementácia klasifikátora aj implementácie je prebratá z titulu, že tieto veci sú spracované dobre a nebolo ich treba extrémne meniť. Ale aj pri implementácii testovania sa museli prispôsobiť veci k implementácii diplomovej práci. Zmeny sa týkali čistejšej implementácie. Z tohto dóvodu budú v krátkosti predstavené jednotlivé testovania. Z bakalárskej práce [66] vyplývajú dva druhy testovania. Prvým je test plochy a druhý je test správneho určenia.

Test plochy je druh testu, ktorý skúma dve oblasti. Prvou oblasťou je koľko plochy bolo označené správne, koľko nesprávne a koľko z celkovej plochy sa označilo správne. Údaje sú počítané v percentách, kvôli rôznej veľkosti obrázka. V princípe ide o dvojprechodný cyklus. V prvom cykle sa nájdu všetky označené plochy z klasifikátora. V druhom cykle sa prejde s reálnymi anotáciami a porovnáva sa, ako sa klasifikátor trafil alebo netrafil. [66]

Test správneho určenia je druh testu, ktorý skúma tiež ako test plochy dve oblasti. Prvá oblasť je aká je ušpená sieť. Druhá oblasť testuje koĺko falošne pozitívnych ochorení sieť označila. Tento test prechádza jedným cyklom. Načíta sa anotácia z klasifikátora a anotácia s poškodeným odtlačkom prsta. Pri prechádzaní týmito anotáciami môžu nastať tri situácie. Prvou situáciou je, že klasifikátor určil správne dané ochorenie. Druhá situácia je, že nájde ochorenie, ale jej priradí zlý typ ochorenia. Posledná situácia je, že označí neexistujúce ochorenie. Na konci testu sa používateľovi vypíšu percentuálne ako sa darilo. [66]

Kapitola 6

Testovanie a výsledky

Kapitola sa zaoberá testovaním a zhodnotením výsledkov. Testovanie prebieha s viacerými natrénovanými konvolučnými sieťami, ktoré boli vytvorené ako experimentálne riešenia. Ďalej budú testované siete trénované na rôznych trénovacích databázach. Pokiaľ to nebude určené inak tak, v nasledujúcich podkapitolách sa budú označovať trénovacie databázy následovne:

- *databáza A* obsahuje syntetické odtlačky z SFinGe, čiastočne NIST_DB4 [67] a reálne odtlačky prstov. Nad touto Databázou bol použitý Gáborov filter s počtom krokov dva a s posunom v každom kroku o 30 stupňov. Zloženie databázy je 37000 čistých vzoriek, 25000 s dyshidrózou a 20000 s bradavicami.
- *databáza B* obsahuje syntetické odtlačky z SFinGe, čiastočne NIST_DB4 [67] a reálne odtlačky prstov. Nad touto Databázou bol použitý Gáborov filter s počtom krokov dva a s posunom v každom kroku o 30 stupňov. Zloženie databázy je 7600 tísic čistých vzoriek, 7600 s dyshidrózou a 7600 s bradavicami.

Ďalej pokiaľ nebude uvedené inak, tak testovacie databázy sú definované následovne:

- *databáza AB* testovacia databáza s 37 čistými odtlačkami, 36 odtlačkov prstov s bradavicou a 37 odtlačkov s dyshidrózou.
- *databáza C1* testovacia databáza s 37 čistými odtlačkami, 40 odtlačkov prstov s bradavicou a 41 odtlačkov s dyshidrózou. V tejto databáze sú syntetické odtlačky, odtlačky z NIST DB4 datbázi a reálne odtlačky.
- *databáza D1* testovacia databáza s 37 čistými odtlačkami.

Testovanie bolo vykonané s dvoma rôznymi testami, ktoré sú popísané v podkapitole 5.10. Na nasledujúcich podkapitolách sú zhrnuté výsledky z testovania. Klasifikátor pri všetkých testoch mal Gáborov filter nastavený na počet krokov dva s posunom o 30 stupňov v každom kroku.

6.1 Testovanie ReLu aktivačnej funkcie

Podkapitola sa bude zaoberať testovaním experimentálneho vylepšenia modelu z bakalárskej práce [66]. Experimentálne vylepšenie je pridané predspracovanie pomocou Gáborovho filtra. Konvolučná sieť sa trénovala nad dvoma trénovacími databázami a testovaná bola

na troch testovacích databázach. Natrénovaná konvolučná sieť s databázou A má úspešnosť trénovania 96 %. U sieti s databázou B je úspešnosť trénovania 44 %. Z dôvodu nízkej úspešnosti sa testuje len sieť s trénovanou databázou A. Na nasledujúcih dvoch testoch sú popísané výsledky.

Test plochy

Prvým testom plochy sa testuje ako sa sieti darilo alebo nedarilo rozpoznať dané ochorenie. Testovanie prebiehalo nad testovacími databázami AB, C1 a D1. Na Prvej tabuľke 6.1 sú zobrazené výsledky z testovania nad databázou AB. Ako je možné vidieť, tak správne označená oblasť u dyshidrózy je 86,31 %. Naproti tomu u bradavíc je celkovo správne označených len 31,35 %. Bez označenia bradavíc je 44,72 %, ale u dyshidrózy len 0,03 %. Dyshidróza sa lepšie naučila pri tomto teste.

	Dyshidróza	Bradavice
Správne označená oblasť	86,31 %	31,35 %
Zle označená oblasť	13,60 %	23,93 %
Bez označenia oblasti	0,03 %	44,72 %

Tabuľka 6.1: Test plochy, sieť trénovaná nad databázou A a testovaná nad AB.

Druhá tabuľka 6.2 zobrazuje výsledky testovania, keď sieť bola testovaná nad databázou C1. Z tabuliek 6.1 a 6.2 vyplýva, že podiel správne označenej oblasti u bradavíc vzrástol o 2,6 %. Naopak u dyshidrózy podiel správne označenej plochy klesol o 2,09 %.

	Dyshidróza	Bradavice
Správne označená oblasť	84,22 %	33,95 %
Zle označená oblasť	14,69 %	47,89 %
Bez označenia oblasti	1,09 %	18,16 %

Tabuľka 6.2: Test plochy, sieť trénovaná nad databázou A a testovaná nad C1.

Tretia tabuľka 6.3 popisuje testovanie a výsledky konvolučnej siete testovanej nad databázou D1. Na tejto tabuľke by percentá mali byť nulové. Ale z tabuľky 6.3 to nevyplýva. Potešujúce je, že nad čistými odtlačkami označilo chorobu dyshidrózy len 0,97 % a 0,47 % z celkovej plochy odtlačkov prstov.

	Dyshidróza	Bradavice
Správne označená oblasť	0 %	0 %
Zle označená oblasť	0,97 %	0,47 %
Bez označenia oblasti	0 %	0 %

Tabuľka 6.3: Test plochy, sieť trénovaná nad databázou A a testovaná nad D1.

Nasledujúci test plochy určuje, koľko percent z celkovej označenej plochy bolo mimo anotácii. Z tabuliek 6.4 a 6.5 vyplýva, že podiel mimo označenej plochy u dyshidrózy je cez 54 % a u bradavíc cez 43 %. Toto sú v celku vysoké čísla pre použitie v praxi. Znamená to, že polovicu celkovej označenej plochy u dyshidrózy je mimo ochorenie. Tretia tabuľka 6.6 má nula percent, lebo odtlačky boli celé označené ako čisté.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	54,45 %	46,88 %

Tabuľka 6.4: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou A a testovaná nad AB.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	56,40 %	43,45 %

Tabuľka 6.5: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou A a testovaná nad C1.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	0 %	0 %

Tabuľka 6.6: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou A a testovaná nad D1.

Test správneho určenia

Test správneho určenia testuje ako sa sieti darilo správne označiť ochorenie, alebo si pomýlilo triedu, alebo označenie bolo falošne pozitívne. Rovnako ako test plochy, tak aj test správneho určenia prebiehal nad celou testovacou databázou. Prvá tabuľka 6.7 popisuje ako sa darilo sieti, ktorá bola testovaná nad databázou AB. U dyshidrózy je určenie správnej triedy 99,5 %, čo je slušný výsledok. Naproti tomu u bradavíc je to len 4,13 %. Ďalej vyplýva, že pri testovacej databáza AB je miera falošnej pozitivy pre bradavice na 52,61 %.

	Dyshidróza	Bradavice
Správna trieda	99,50 %	4,13 %
Zlá trieda	0,40 %	43,26 %
Falošná pozitivita	0,10 %	52,61 %

Tabuľka 6.7: Test správneho určenia, sieť trénovaná nad databázou A a testovaná nad AB.

Z druhej tabuľky 6.8 je vidno, že miera falošnej pozitivity u bradavice sa zvýšil až na 91,14 %. Toto je veľmi zlý výsledok, ak by sa chcela takáto sieť používať v praxi.

	Dyshidróza	Bradavice
Správna trieda	98,32 %	1,01 %
Zlá trieda	1,61 %	7,86 %
Falošná pozitivita	0,07 %	91,14 %

Tabuľka 6.8: Test správneho určenia, sieť trénovaná nad databázou A a testovaná nad C1.

Tretia tabuľka 6.9 zobrazuje ako sa darilo konvolučnej sieti nad testovacou databázou D1. Z tabuľky vyplýva, že našlo ochorenie aj na čistých odtlačkoch prstoch. . Z tabuľky 6.3 sa dá vyčítať, že celkovej označenej plochy na odtlačkoch prsta chorobou dyshidrózy je len 0,97 % plochy bolo označené dyshidrózou a 0,47 % bolo označené bradavicou. Z tohto pohľadu si konvolučná sieť s aktivačnou funkciou relu a predspracovaním Gáborovým filtrom dobre poradí z rôznymi čistými odtlačkami.

	Dyshidróza	Bradavice
Správna trieda	0 %	0 %
Zlá trieda	0 %	0 %
Falošná pozitivita	100 %	100 %

Tabuľka 6.9: Test správneho určenia, sieť trénovaná nad databázou A a testovaná nad D1.

6.2 Testovanie zmeny leaky ReLu aktivačnej funkcie

Ako z názvu vyplýva, tak podkapitola sa bude zaoberať testovaním experimentálneho vylepšenia zmeny aktivačnej funkcie z ReLu na leaky ReLu. Podobne ako v podkapitole 6.1 sa bude test zaoberať dvoma natrénovanými sieťami a troma testovacími databázami. Úspešnosť trénovania nad databázou A je 96 %. Úspešnosť trénovania tejto sieti nad databázou B je iba 34 %. Z tohto dôvodu bude test vykonaný len nad sieťou s trénovacou databázou A. Na nasledujúcich testoch sú zobrazené výsledky z testovania.

Test plochy

Rovnako ako v predchadzajúcej podkapitole 6.1 sa bude testovať plocha. Tabuľka 6.10 popisuje ako sa darilo trénovanej sieti nad testovanou databázou AB. Nad testovacou databázou AB dyshidróza u leaky relu má až 90,21 % správneho určenia. No bradavice majú len 32,10 %. Podiel oblasti u bradavíc je 49,99 % ale u dyshidrózy je to len 0,08 % .

	Dyshidróza	Bradavice
Správne označená oblasť	90,21 %	32,10 %
Zle označená oblasť	9,71 %	17,91 %
Bez označenia oblasti	0,08 %	49,99 %

Tabuľka 6.10: Test plochy, sieť trénovaná nad databázou A a testovaná nad AB.

Druhá tabuľka 6.2 zobrazuje výsledky, keď sieť bola testovaná nad databázou C1. S pridaním reálnych a NIST DB4 odtlačkov vzrástla správnosť označenia plochy u bradavice na 40,39 %. Znížila sa aj oblasť bez označenia na 23,28 %.

	Dyshidróza	Bradavice
Správne označená oblasť	86,30 %	40,39 %
Zle označená oblasť	12,35 %	36,33 %
Bez označenia oblasti	1,35 %	23,28 %

Tabuľka 6.11: Test plochy, sieť trénovaná nad databázou A a testovaná nad C1.

Nasledujúca tabuľka 6.12 zobrazuje ako sa darilo sieti s trénovacou databázou A a nad testovacou databázou D1. Čísla z tabuľky 6.12 sú pozitívne, keď len 1,57 % čistých odtlačkov sú označené ako dyshidróza. Podobne optimisticky je to aj u bradavíc, kde to číslo je ešte menšie. Je to len 1,25 %.

Druhý test na test plochy je celkový podiel označenej plochy mimo anotáciu. Ako to vypadá z aktivačnou funkciou leaky Relu je zobrazené na nasledujúcich tabuľkách. Prvá tabuľka 6.13 popisuje testovanie nad konvolučnou sieťou, ktorá bola testovaná na testovacej databáze AB.

	Dyshidróza	Bradavice
Správne označená oblasť	0 %	0 %
Zle označená oblasť	1,57 %	1,25 %
Bez označenia oblasti	0 %	0 %

Tabuľka 6.12: Test plochy, sieť trénovaná nad databázou A a testovaná nad D1.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	54,45 %	45,53 %

Tabuľka 6.13: Podiel celkovej označenej plochy mimo anotáciu,sieť trénovaná nad databázou A a testovaná nad AB.

Na Druhej tabuľke 6.14 bola sieť testovaná nad C1. Z tabuliek 6.13 a 6.14 je vidieť, že podiel označených oblastí mimo anotácie klesol o 3,07 % v prípade bradavíc. V prípade dyshidrózy len o 1,24 %.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	53,21 %	42,46 %

Tabuľka 6.14: Podiel celkovej označenej plochy mimo anotáciu,sieť trénovaná nad databázou A a testovaná nad C1.

Posledná tabuľka zobrazuje 6.15 ako sa darilo sieti, keď bola testovaná nad D1. Podobne ako aj v minulom teste s databázou D1 je podiel nulový, lebo celá oblasť je označená ako čistá.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	0 %	0 %

Tabuľka 6.15: Podiel celkovej označenej plochy mimo anotáciu,sieť trénovaná nad databázou A a testovaná nad D1.

Test správneho určenia

Test správneho určenia podobne ako test plochy prebiehal nad troma testovacími databázami a zároveň je vyhodnocovaný nad celou databázou. Test má určiť koľko percent s klasifikovaného ochorenia je správna trieda, koľko percent je zlá trieda, a koľko percent je falošná pozitivita. Prvá tabuľka 6.16 zobrazuje ako sa darilo konvolučnej siete pri testovaní nad databázou AB.

	Dyshidróza	Bradavice
Správna trieda	99,59 %	5,64 %
Zlá trieda	0,37 %	35,59 %
Falošná pozitivita	0,04 %	58,78 %

Tabuľka 6.16: Test správneho určenia, sieť trénovaná nad databázou A a testovaná nad AB.

Druhá tabuľka 6.17 popisuje ako sa darilo v teste konvolučnej sieti, ktorá bola testovaná nad databázou C1. Z tabuliek 6.16 a 6.17 vyplýva, že pri testovaní s databázou C1 sa zvýšila falošná pozitiva u bradavíc na 91,64 %. Taktiež sa znížila správna trieda z 5,64 % na 1,37 % u bradavíc. Dyshidróza sa chová aj pri reálnych odtlačkoch výborne.

	Dyshidróza	Bradavice
Správna trieda	98,41 %	1,37 %
Zlá trieda	1,55 %	6,99 %
Falošná pozitivita	0,05 %	91,64 %

Tabuľka 6.17: Test správneho určenia, sieť trénovaná nad databázou A a testovaná nad C1.

Posledná tabuľka 6.18 zobrazuje ako sa darilo sieti pri testovaní nad databázou D1. Falošná pozitivita je síce 100 %, ale z tabuľky 6.12 vyplýva, že z celkovej plochy čistých odtlačkov bolo len 1,57 % označených dyshidrózou a 1,25 % označených ako bradavica. Toto je veľmi dobrý výsledok.

	Dyshidróza	Bradavice
Správna trieda	0 %	0 %
Zlá trieda	0 %	0 %
Falošná pozitivita	100 %	100 %

Tabuľka 6.18: Test správneho určenia, sieť trénovaná nad databázou B a testovaná nad AB.

6.3 Testovanie zmeny tanh aktivačnej funkcie

Podkapitola bude rozoberať testovanie experimentálneho vylepšenia pri zmene aktivačnej funkcie tanh. Úspešnosť trénovania nad databázou A je 98 %. Úspešnosť trénovania sieti nad databázou B je 96 %. Úspešnosť tejto sieti je nad 90 %. Z tohto dôvodu budú testy prevedené aj na tejto sieti. Na nasledujúcich dvoch testoch sa ukáže ako sa sieťam darilo.

Test plochy

Test plochy je zobrazený na nasledujúcich tabuľkách. Tabuľky 6.19 a 6.20 zobrazujú výsledky testovania, ktoré boli prevedené nad testovacou databázou AB. Z tých tabuliek vyplýva, že označeniu bradavice sa lepšie darilo správne určiť pri trénovaní s databázou A. Ale len o 2,24 %. Dyshidróze sa zas lepšie darilo, keď bola trénovaná nad databázou B. Lepšie sa jej darilo o 6,49 %.

	Dyshidróza	Bradavice
Správne označená oblasť	82,80 %	45,89 %
Zle označená oblasť	17,15 %	25,30 %
Bez označenia oblasti	0,05 %	28,81 %

Tabuľka 6.19: Test plochy, sieť trénovaná nad databázou A a testovaná nad AB.

Nasledujúce dve tabuľky 6.21 a 6.22 popisujú ako sa darilo sieťam nad testovacou databázou C1. Nad testom s reálnymi odtlačkami si lepšie viedla sieť trénovaná nad databázou A

	Dyshidróza	Bradavice
Správne označená oblasť	89,29 %	43,65 %
Zle označená oblasť	10,50 %	17,67 %
Bez označenia oblasti	0,20 %	38,68 %

Tabuľka 6.20: Test plochy, sieť trénovaná nad databázou B testovaná nad AB.

pri bradaviciach. Tam bolo správne určených 24,52 % plochy, zatiaľ čo pri sieti trénovanou nad databázou B len 17,09 %

	Dyshidróza	Bradavice
Správne označená oblasť	84,29 %	24,52 %
Zle označená oblasť	14,73 %	53,57 %
Bez označenia oblasti	0,97 %	21,92 %

Tabuľka 6.21: Test plochy, sieť trénovaná nad databázou A a testovaná nad C1.

	Dyshidróza	Bradavice
Správne označená oblasť	85,89 %	17,09 %
Zle označená oblasť	12,59 %	51,89 %
Bez označenia oblasti	1,52 %	31,02 %

Tabuľka 6.22: Test plochy, sieť trénovaná nad databázou B a testovaná nad C1.

Posledné dve tabuľky 6.23 a 6.24 popisujú ako sa darilo sieťam nad testovacou databázou D1. Pri tomto teste si lepšie viedla sieť trénovaná nad databázou B. Určenie choroby na čistých odtlačkoch pri dyshidróze a bradaviciach je pod 1,3 %, čo je veľmi slušný výsledok.

	Dyshidróza	Bradavice
Správne označená oblasť	0 %	0 %
Zle označená oblasť	1,45 %	1,26 %
Bez označenia oblasti	0 %	0 %

Tabuľka 6.23: Test plochy, sieť trénovaná nad databázou A a testovaná nad D1.

	Dyshidróza	Bradavice
Správne označená oblasť	0 %	0 %
Zle označená oblasť	1,24 %	1,19 %
Bez označenia oblasti	0 %	0 %

Tabuľka 6.24: Test plochy, sieť trénovaná nad databázou B a testovaná nad D1.

Nasledujúci test plochy určuje koľko percent z celkovej označenej plochy bolo mimo anotácii. Rovnako ako test plochy, tak aj tento test bude vyhodnotený v páre nad testovacou databázou Prvé dve tabuľky 6.25 a 6.26 popisujú koľko celkovej plochy bolo označené mimo anotácie. Lepšie výsledky pre bradavicu mimo označenia anotácii vychádza pri sieti trénovanou nad databázou B. Aj podiel mimo anotácii pri dyshidróze klesol s trénovacou databázou B.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	38,31 %	56,61 %

Tabuľka 6.25: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou A a testovaná nad AB.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	36,83 %	50,45 %

Tabuľka 6.26: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou B a testovaná nad AB.

Nasledujúce dve tabuľky 6.27 a 6.28 popisujú ako sa darilo sieťam, keď boli pridané do testovanej databázy reálne odtlačky. Taktiež ako v predchadzajúcom teste sa lepšie darilo nad dyshidrózou a bradavicami sieti s trénovacou databázou B.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	45,49 %	58,31 %

Tabuľka 6.27: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou A a testovaná nad C1.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	42,32 %	52,05 %

Tabuľka 6.28: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou B a testovaná nad C1.

Rovnako ako v ostatných testoch, tak aj teraz majú mimo anotáciu nula percent, kvôli tomu, že celý odtlačok je anotovaný ako čistý. Toto je vidieť na tabuľkách 6.29 a 6.30

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	0 %	0 %

Tabuľka 6.29: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou A a testovaná nad D1.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	0 %	0 %

Tabuľka 6.30: Podiel celkovej označenej plochy mimo anotáciu, sieť trénovaná nad databázou B a testovaná nad D1.

Test správneho určenia

Test správneho určenia sa rovnako ako test plochy testuje v páre nad testovacími databázami. Prvé dve tabuľky sú testované nad databázou AB. Z týchto tabuliek 6.31 a 6.32 je

možné si všimnúť, že úspešnosť správnej triedy pri bradavici sa znížil o 7,62 % pri sieti trénovanej nad databázou B. U dyshidrózy je to len 0,55 % menej. Sieť trénovaná nad databázou B sa zvýšila falošná pozitivita na 79,30 %.

	Dyshidróza	Bradavice
Správna trieda	99,54 %	9,82 %
Zlá trieda	0,10 %	74,60 %
Falošná pozitivita	0,35 %	15,58 %

Tabuľka 6.31: Test správneho určenia, sieť trénovaná nad databázou A a testovaná nad AB.

	Dyshidróza	Bradavice
Správna trieda	98,99 %	2,20 %
Zlá trieda	0,97 %	18,50 %
Falošná pozitivita	0,04 %	79,30 %

Tabuľka 6.32: Test správneho určenia, sieť trénovaná nad databázou B a testovaná nad AB.

Nasledujúce dve tabuľky 6.33 a 6.34 zobrazujú, ako sa darilo sieťam nad testovacou databázou C1. Z čísiel je zrejmé, že sa lepšie darilo sieti trénovanej nad databázou B. Úspešnosť určenie správnej triedy dyshidrózy je 99,55 %. U bradavíc je to len 1,85 %. Miera falošnej pozitivity u bradavíc je 86,64 %.

	Dyshidróza	Bradavice
Správna trieda	99,04 %	1,63 %
Zlá trieda	0,73 %	11,73 %
Falošná pozitivita	0,24 %	86,64 %

Tabuľka 6.33: Test správneho určenia, sieť trénovaná nad databázou A a testovaná nad C1.

	Dyshidróza	Bradavice
Správna trieda	99,55 %	1,85 %
Zlá trieda	0,26 %	17,45 %
Falošná pozitivita	0,19 %	80,70 %

Tabuľka 6.34: Test správneho určenia, sieť trénovaná nad databázou B a testovaná nad C1.

Posledné dve tabuľky 6.35 a 6.36 zobrazujú výsledky nad testovacou databázou D1. Síce je falošná pozitivita 100 %, ale celkové označené plochy sú veľmi nízke. Pre ochorenie dyshidrózu aj bradavicu sa lepšie darilo, keď sa sieť trénovala nad databázou B.

	Dyshidróza	Bradavice
Správna trieda	0 %	0 %
Zlá trieda	0 %	0 %
Falošná pozitivita	100 %	100 %

Tabuľka 6.35: Test správneho určenia, sieť trénovaná nad databázou A a testovaná nad D1.

	Dyshidróza	Bradavice
Správna trieda	0 %	0 %
Zlá trieda	0 %	0 %
Falošná pozitivita	100 %	100 %

Tabuľka 6.36: Test správneho určenia, sieť trénovaná nad databázou B a testovaná nad D1.

6.4 Testovanie zmeny modelu

Pre účely experimentálneho vylepšenia sa testovalo aj zmena modelu a to na tri možnosti. Prvým skúšaným modelom bol Resnet50. Druhým bol testovaný predtrénovaný model Inceptionv3 a z triedy zbytkových sietí bol ešte navyše zobratý Resnet152. Tento test nebol ani prevedený. Pri trénovaní došlo k zlyhaniu učiaceho procesu. všetky tri modely sú trénované na farebných obrázkoch. Problém nastal, keď sa zmenila prvá konvolučná vrstva. Všetky predtrénované známe modely sú trénované na farebných obrázkoch. Pre špecifické potreby ako sú odtiene sivej je zapotreby upraviť celý kód k predtrénovaným modelom. Týmto sa stráca výhodnosť použitia predtrénovaného modelu, ako bolo spomínané v podkapitole 3.10. Využiť predtrénovaný model, kde by všetky kanály boli v odtieni sivej. To by znamenalo, že obrázok by bol do siete poslaný trikrát. Tento postup bol zamietnutý z logického hľadiska. Kopírovať do troch kanálov rovnaký obsah nie je dobré a nie je to prístup, ktorý je používaný. Z tohto dôvodu, zostal model z bakalárskej práce pre trénovanie nových poškodení a rozšírený klasifikátor.

6.5 Testovanie rozšireného klasifikátora

Bohužiaľ sa nestíhlo natrénovať a otestovať rozšírenie klasifikátora. Tento problém vznikol, pretože diplomová práca veľkú časť času strávila na experimentálnom vylepšení. Problém experimentálneho vylepšenia vznikal pri tvorbe trénovacej databázi a určením správneho kroku Gáborovho filtra. Trénovanie je plne rozšírené na rôzny počet tried a klasifikátor je pripravený pre klasifikáciu rôzneho tlaku a psoriázy.

6.6 Testovanie siete z bakalárskej práce

Podkapitola sa bude zaoberať testovaním siete z bakalárskej práce [66]. Tak ako aj podkapitola vyššie, táto sieť bola testovaná nad testovacou databázou AB a databázou C. Ďalej bola sieť testovaná iba na čistých odtlačkoch. Nasledujúce podkapitoly budú popísané z pohľadu testovacej databázy.

Test plochy

Prvým testom je test plochy. Na nasledujúcich tabuľkách je popísané ako sa sieti darilo. Tabuľka 6.37 zobrazuje test plochy nad syntetickými odtlačkami prstov. Pri syntetických odtlačkoch konvolučná sieť moc nefunguje. Ako je vidieť z tabuľky 6.37, tak konvolučná sieť je výborne naučená na dyshidrózu ale veľmi zle na bradavicu. Ďalším test plochy bol vykonaný nad databázou C1. Výsledky nad touto databázou sú zobrazené v tabuľke 6.38. A posledný test plochy bol spravený nad 40 odtlačkami prstov, ktoré nemali žiadne poškodenie a boli to odtlačky vygenerované pomocou SFinGe. Výsledky sú zobrazené na nasledujúcej

tabuľke 6.39. Z prvých dvoch tabuliek je možné si všimnúť, že úspešnosť siete rozpoznať správne ochorenie klesá s reálnymi odtlačkami. Toto môže byť ovplyvnené viacerými faktormi. Prvý faktor je, že je málo reálnych odtlačkov prstov v databáze. Druhý faktor môže byť, že pri označovaní reálnych odtlačkov bolo neoznačené ochorenie alebo bolo označené zle. Z poslednej tabuľky vyplýva, že sieť si nevie poradiť s čistými odtlačkami prstov.

	Dyshidróza	Bradavice
Správne označená oblasť	95,95 %	76,67 %
Zle označená oblasť	1,13 %	5,67 %
Bez označenia oblasti	2,92 %	17,66 %

Tabuľka 6.37: Test plochy, testovacia databáza AB, sieť z bakalárskej práce.

	Dyshidróza	Bradavice
Správne označená oblasť	89,72 %	68,81 %
Zle označená oblasť	4,17 %	16,91 %
Bez označenia oblasti	6,12 %	14,28 %

Tabuľka 6.38: Test plochy, testovacia databáza C1, sieť z bakalárskej práce.

	Dyshidróza	Bradavice
Správne označená oblasť	0 %	0 %
Zle označená oblasť	85,45 %	16,5 %
Bez označenia oblasti	0 %	0 %

Tabuľka 6.39: Test plochy, testovacia databáza D1, sieť z bakalárskej práce.

S testom plochy sa robil ešte jeden test. Test označuje koľko celkovej označenej plochy je označené mimo anotáciu. Na prvej tabuľke 6.40 je označenie mimo anotácie nad testovacou databázou AB. Ako je vidieť z tejto tabuľky, tak až 76,17 % sa označuje dyshidróza. Oproti tomu bradavica má označenie mimo neoznačenej anotácie len 27,17 %. Z týchto údajov vyplýva, že dyshidróza je označovaná na rôznych odtlačkoch ako rôzne čiary, pri odtlačkoch, ktoré nemajú jasné kontúry. Druhá tabuľka 6.41 zobrazuje výsledky testy nad testovacou databázou C1. Ako sa pridali reálne odtlačky do testovacej databázy, tak podiel mimo označenej plochy pri dyshidróze klesol o 9,24 % a pri bradavici klesol o 0,66 %. Posledná tabuľka 6.42 zobrazuje výsledky nad trénovacou databázou D1. Výsledky sú nulové z dôvodu, že odtlačok prsta je celý bez poškodenia.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	76,17 %	27,17 %

Tabuľka 6.40: Podiel celkovej označenej plochy mimo anotáciu, testovacia databáza AB, sieť z bakalárskej práce.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	66,93 %	26,55 %

Tabuľka 6.41: Podiel celkovej označenej plochy mimo anotáciu, testovacia databáza C1, sieť z bakalárskej práce.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácií	0 %	0 %

Tabuľka 6.42: Podiel celkovej označenej plochy mimo anotáciu, testovacia databáza D1, sieť z bakalárskej práce.

Test správneho určenia

Test podobne ako test plochy prebiehal nad troma testovacími databázami. Podobne ako test plochy, tak test správneho určenia je vyhodnocovaný nad celou databázou. Testovanie sa bude zaoberať testovaním a výsledkami správnym určením, definovanie zlej triedy a falošne pozitívnym označením choroby. Prvá tabuľka 6.43 zobrazuje ako sa darilo nad testovacou databázou AB. Pri pohľade na čísla si sieť veľmi dobre poradila so správnym určením dyshidrózy s 98,86 %, čo je výborný výsledok. Naproti tomu pri pohľade na bradavicu je situácia odlišná. Zo všetkých obrázkov sieť určila správne triedu len 0,58 %, čo je veľmi zlé. Ešte horšie je na tom bradavica pri falošne pozitívnych označených. Tam je to 99,33 % nad celou databázou. Toto je v praxi nepoužiteľné, pretože označuje bradavicu, kde nie je alebo je to iné ochorenie.

	Dyshidróza	Bradavice
Správna trieda	98,86 %	0,58 %
Zlá trieda	1,13 %	0,09 %
Falošná pozitivita	0,01 %	99,33 %

Tabuľka 6.43: Test správneho určenia, testovacia databáza AB, sieť z bakalárskej práce.

Druhá tabuľka 6.44 zobrazuje ako sa darilo nad testovacou datab8zou C1. Ako je vidieť z tabuľky, tak oproti predchadzajúcemu testu si sieť pohoršila pri správnom určení dyshidrózy o 7, 86 %. U bradavice sa to znížilo o 0,29 %. Pri pridaní reálnych odtlačkov nad celou datbázou vzrástla miera falošne pozitívnych o 0,25 %. Z čísiel vychádza, že bradavice v rôznych odtlačkoch prstoch nie je možné nájsť. Nedá spoliehať, že sieť určí správne bradavicu. Na druhej strane dyshidróza je aj sreálnymi odtlačkami nad 90 % úspešnosti, čo radí sieť ako vhodnú na rozlišovanie dyshidrózy na praktické účely.

	Dyshidróza	Bradavice
Správna trieda	91 %	0,29 %
Zlá trieda	8,93 %	0,13 %
Falošná pozitivita	0,06 %	99,58 %

Tabuľka 6.44: Test správneho určenia, testovacia databáza C1, sieť z bakalárskej práce.

Posledná tabuľka 6.45 zobrazuje testovanie nad datbázou D1. Pri tejto tabuľke by mali byť percentá nulové, keďže sa jedná o čisté odtlačky prstov, bez žiadnej choroby. Z ta-

buľky 6.45 vyplýva, že falošne pozitvnych je 100 %. Z tohto vyplýva, že nad čistými odtlačkami, pokiaľ nie sú dokonalé nevie označiť celý odtlačok za čistý. Na čistom ochorení nájde vždy nejaké ochorenie buď dyshidrózu alebo bradavicu.

	Dyshidróza	Bradavice
Správna trieda	0 %	0 %
Zlá trieda	0 %	0 %
Falošná pozitivita	100 %	100 %

Tabuľka 6.45: Test správneho určenia, testovacia databáza D1, sieť z bakalárskej práce.

6.7 Zhodnotenie testovania

Podkapitola sa bude zaoberať vyhodnotením experimentálnych vylepšení, ktoré boli prevedené. Prvým vyhodnotením bude zmena modelu na predtrénovaný model. Ako sa spomína v podkapitole 6.4, tak táto zmena dopadla neúspechom. Problém vznikol pri zmene prvej konvolučnej vrstvy, ktorou sa chcelo docieliť aby obrázok mohol byť poslaný na vstup v odtieňoch sivej. Ďalšou úvahou bolo posielať do všetkých kanálov odtiene sivej, čo bolo zamietnuté z pohľadu posielania rovnakého kanála do troch rôznych kanálov. Využitie predtrénovaného modelu pre obrázky v odtieňoch sivej je nepoužiteľný. Alternatívou ako využiť tieto modely je, že sa nepoužije predtrénovaný model, ale sa upraví kód každého modelu. Tento spôsob by mohol priniesť nejaký úžitok, ale sa stráca výhoda použitia predtrénovaného modelu, ktorý je trénovaný nad veľkým počtom trénovacích dát.

V podkapitolách 6.1, 6.2, 6.3 a 6.4 sú popísané výsledky z testovania nad rôznymi experimentálnymi vylepšeniami. Z týchto podkapitol vyplýva, že najlepšia sieť na porovnanie so sieťou z bakalárskej práce [66] je sieť, ktorá bola trénovaná nad databázou B a ako aktivačnú funkciu má tanh. Predspracovanie trénovacej databázy bol Gáborov filer s počtom krokov dva. Pri každom kroku bolo posunutie o 30 stupňov. Klasifikátor mal rovnako nastavený Gáborov filter ako trénovacia databáza. Za povšimnutie stojí tabuľka 6.11, kde sa podarilo dostať správne označenie bradavice na 40,39 %. Z tabuľky 6.17 vyplýva, že až 91,64 % je falošne pozitívnych označení. Táto sieť sa učila nad trénovacou databázou A a test bol prevedený nad testovacou databázou C1. Na nasledujúcich podkapitolách bude zhodnotenie ako si sieť z bakalárskej práce viedla z novou sieťou.

6.7.1 Test plochy

Prvým testom plochy bolo určenie správnej plochy ochorenia, koľko bolo zle označenej, a koľko bolo označené mimo anotácie. Porovnávať sa bude nad testovacou databázou C1 a D1. Tabuľky 6.22 a 6.38 zobrazujú test nad testovacou databázou C1. Z týchto čísiel je zrejmé, že sieť z bakalárskej práce lepšie označuje správne oblasť pre dyshidrózu o 3,83 %. Pri zlej označenej oblasti je to o niečo horšie. Sieť z experimentálneho vylešenia označuje o 8,42 % horšie ako sieť z bakalárskej práce. Experimentálne vylepšenie označí bez označenia o 4,6 % menej ako sieť z bakalárskej práce. Pri bradavici je to diametrálne odlišné. U experimentálneho vylepšenia sa podiel označenej oblasti prepadol o 51,72 %, čo je veľká nevýhoda pre experimentálne vylepšenie. Experimentálne vylepšenie označí zlú oblasť pre bradavicu o 34,98 % viac ako sieť z bakalárskej práce. Bez označenia oblasti označí experimentálne vylepšenie o 7,64 % viac ako sieť z bakalárskej práce.

Testovanie nad testovacou databázou D1 je zobrazené na tabuľkách 6.24 a 6.39. Z týchto tabuliek vyplýva, že experimentálne vylepšenie označilo o 84,21 % menej dyshidrózy na čistých odtlačkoch prstoch. Na bradaviciach je to o 15,31 %.

Druhý test na plochu bolo označenie mimo anotácií nad testovacou databázou C1. Výsledky sú na tabuľkách 6.28 a 6.41. Z týchto tabuliek je možné vyčítať, že u sieti z bakalárskej práce sa označilo mimo anotáciu 66,93 % dyshidrózy a 26,55 % u bradavíc. Experimentálne vylepšenie má piel označenej plochy mimo anotáciu pre dyshidrózu 42,32 % a pre bradavicu má podiel označenej plochy mimo anotácie 52,05 %.

6.7.2 Test správneho určenia

Tento test testoval koľko percent sa trafilo do správnej triedy, koľko odtlačkov bolo určených zle a koľko bolo falošnej pozitivity. Tieto výsledky sú nad trénovacou databázou C1. Výsledky môžeme vidieť na tabuľkách 6.34 a 6.44. Z tohto vyplýva, že experimentálne vylepšenie zlepšilo určenie správnej triedy pre dyshidrózu na 99,55 %, určenie zlej triedy len na 0,26 %. Falošnú pozitivitu zvýšila na 0,19 %. U bradavíc je situácia iná. Tam sa podarilo určiť správnu triedu na 1,85 %, čo je zlepšenie od siete z bakalárskej práce. Ale zvýšil sa podiel zlej triedy na 17,45 %, ale znížila sa falošná pozitivita na 80,70 %. Podarilo sa vylepšiť rozpoznávanie dyshidrózy skoro k 100 %. Na druhú stranu sa nepodarilo zlepšiť určenie bradavice.

Problém mohol vzniknúť pri výbere a tvorbe databázy odtlačkov prstov. Databáza je dôležitou časťou. Z výsledkov vypadá, že trénovacia databáza bola zvolená zle. Dôležité na tomto vylepšení je, že sa podarilo znížiť označovanie chorôb na rôznych čistých odtlačkoch na minimum. Ďalší fakt, ktorý to môže ovplyvňovať je, že sa zvolil zlý uhol pre Gáborov filter. Ďalším problémom môže byť samotný model, ktorý sa nepodarilo vymeniť za predtrénovaný model.

6.7.3 Možnosti rozšírenia alebo vylepšenia do budúcna

Implementácia je ľahko rozšíriteľná o ďalšie typy ochorení alebo príznakov pri snímaní. Implementácia sa snažila byť implementovaná podľa objektového orientovaného programovania. V maximálnej miere je použitý princíp DRY *(Don't repeat yourself)*. Ďalším rozšírením je zmeniť model na nejaký z predtrénovaných ale zmeniť jeho konvolučné vrstvy tak, aby mohol byť vstup v odtieňoch sivej. Ďalšia zmena, ktorá by mohla byť použitá, tak je využitie konvolučnej sieti s regiónmi.

Kapitola 7

Záver

Diplomová práca sa zameriava na detekciu a klasifikáciu poškodenia odtlačku prsta pomocou neurónových sietí. Práca nadväzuje na bakalársku prácu [66]. Diplomová práca sa zameriava hlavne na experimentálne vylepšenie siete z bakalárskej práce. Najskôr ako sa mohlo začať nadväzovať na túto prácu, bolo potrebné naštudovať danú problematiku. Prvou problematikou je naštudovanie samotnej biometrii a pojmu odtlačku prstu. Pri odtlačkoch prstoch bolo dôležité prebrať oblasti, ako napríklad snímanie odtlačkov prstov, spracovanie odtlačkov prstov, ale aj poškodenie odtlačkov prstov pri snímaní a rôzne choroby. Ďalšou problematikou, ktorú bolo potrebné naštudovať, sú neurónové siete. Základnou časťou je porozumenie významu medzi biologickým a umelým neurónom. Následne bolo nutné pochopiť pojmy aktivačná funkcia, stratová funkcia, optimalizátory, modely neurónových sietí. Ďalej bolo potrebné naštudovať viacvrstvové siete, konvolučné siete a nakoniec proces trénovania. Posledným krokom pred experimentálnym vylepšením siete bolo si navrhnúť, aké zmeny sa budú vykonávať. Plánované zmeny sú výmena modelu za predtrénovaný model, zmena aktivačnej funkcie u modelu z bakalárskej práce [66], pridanie predspracovanie obrázku, zmena tvorby databázy. Použité odtlačky prstov boli NIST DB4, syntetické odtlačky vygenerované pomocou SFinGe a reálne odtlačky prstov. Nakoniec po experimentálnom vylepšení rozšíriť klasifikátor o nové ochorenie alebo poškodenie pri snímaní napríklad tlakom.

Prvým krokom pri implementácii bolo potrebné vymeniť API Keras za framework PyTorch. Táto zmena bola vykonaná kvôli rýchlosti učenia. Ďalším dôvodom prechodu je lepšia podpora v komunite, ako aj objektový prístup k modelu. Zároveň má objektový prístup k načítaniu databázy, ktorý pomáha udržať čistotu kódu. Ďalším krokom pri experimentálnom vylepšovaní bolo zmeniť predspracovanie databázy. Táto zmena mala urýchliť celý učiaci proces. Pri tejto zmene sa musel zmeniť aj spôsob načítavania databázy. Zároveň pri tejto úprave sa pridalo predpsracovanie obrázka pomocou Gáborovho filtra. Ďalší krok bol prepísať model z API Keras do modelu PyTorch. Táto úprava vyžadovala pochopenie ako funguje API Keras a PyTorch. Po tomto kroku sa implementovalo prvé experimentálne vylepšenie a to zmena aktivačnej funkcie. Zmena aktivačnej funkcie je možné cez argumenty príkazového riadku. Pri tejto zmene došlo aj k zmene počtu výstupov, tak aby konvolučná sieť mohla byť učená na rôzne ochorenia a poškodenia pri snímaní. Ďalším experimentálnym vylepšením bola zmena modelu na niektorý z predtrénovaných modelov. Pri tomto sa upravovali prvé konvolučné vrstvy, aby obrázok mohol byť poslaný na vstup v odtieňoch sivej. Nakoniec sa implementoval samotný učiaci proces. Tento proces je o niečo zložitejší na implementáciu ako v API Keras. Následovala implementácia klasifikátora, ktorá sa preberala z bakalárskej práce. Tento klasifikátor bol upravený pre potreby diplomovej práce. Prvá zmena bola prečistenie kódu a vytvorenie triedy pre klasifikátor. Ďalej muselo byť upravené

načítavanie modelu, keďže sa jednalo o experimentálne vylepšenie aj zmena frameworku PyTorch a rozšírenie klasifikátora. Implementácia testovania bola prebratá z bakalárskej práce ale musela byť opravená, keďže nebola ošetrená na delenie nulou. Taktiež prebehlo prečistenie kódu. Po tomto všetkom sa mohlo prejsť k trénovaniu a vyhodnoteniu.

Prvé vyhodnotenie je zmena modelu na niektorý z predtrénovaných modelov. Táto zmena dopadla neúspechom. Zmena dopadla neúspechom, pretože všetky predtrénované modely sú trénované nad farebnými obrázkami. Experimentálne sa vyskúšala zmeniť prvá konvolučná vrstva, aby na vstup mohol byť privedený obrázok v odtieňoch sivej. Po tomto neúspechu sa experimentálne vylepšenie zameralo na testovanie zmeny aktivačnej funkcie nad rôznymi trénovacími databázami. Výsledok tohto je porovnanie zmenená aktivačná funkcia z ReLu na tanh. Pri trénovaní bol upravený aj optimalizátor. V PyTorch je potrebné mať v SGD optimalizátore nastavený momentum na 0,9, aby sa sieť učila a zvyšovala presnosť. Záver z tohto je, že experimentálne vylepšená sieť znížila na celkovej ploche čistých odtlačkov prstov označenie dyshidrózy o 84,21 %. Ďalej sa zvýšila presnosť určenia dyshidrózy na 99,55 %. Pri bradavici došlo k poklesu správne určenej plochy o 51,72 %. Vplyv na zhoršenie môže mať inak definovaná trénovacia databáza alebo pridanie Gáborovho filtra. Gáborov filter mohol spôsobiť odstránenie dôležitých kontúr pre detekovanie bradavice.

Literatúra

[1] Al Masri, A. *What Are Overfitting and Underfitting in Machine Learning?* [online], 22. júna 2019 [cit. 2021-01-08]. Dostupné z: https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690.

[2] Barotová Štěpánka. *Detektor kožních onemocnění u technologie otisků prstů.* Brno, CZ, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/20057/.

[3] Bharat, N. *Overfitting and undefitting image* [online]. [cit. 2021-01-08]. Dostupné z: https://qph.fs.quoracdn.net/main-qimg-0d12f79a596c9b2ee1e23476fa3d44aa.

[4] Brannon, H. L. *Skin Anatomy: The Layers of Skin and Their Functions* [online]. About, Inc. (Dotdash), august 2020 [cit. 2020-11-27]. Dostupné z: https://www.verywellhealth.com/skin-anatomy-1068880.

[5] Brownlee, J. *Loss and Loss Functions for Training Deep Learning Neural Networks* [online]. Január 2019 [cit. 2021-01-06]. Dostupné z: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.

[6] Brownlee, J. *How to Choose Loss Functions When Training Deep Learning Neural Networks* [online], 30. januára 2020 [cit. 2021-01-06]. Dostupné z: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/.

[7] Bruballa, R. G. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names* [online]. Máj 2018 [cit. 2021-01-06]. Dostupné z: https://gombru.github.io/2018/05/23/cross_entropy_loss.

[8] Cappelli, R., Maio, D. a Maltoni, D. Synthetic fingerprint-database generation. In: *Object recognition supported by user interaction for service robots.* 2002, sv. 3, s. 744–747 vol.3. DOI: 10.1109/ICPR.2002.1048096.

[9] Cappelli, R. *SFinGe: an Approach to Synthetic Fingerprint Generation.* International Workshop on Biometric Technologies, 2004. 147-154 s.

[10] Chaloupka, R. *Generátor otisků prstů.* Brno, CZ, 2007. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/2480/.

[11] Charan, S. *What is the difference between VGG16 and VGG19 neural network?* [cit. 2021-01-08]. Dostupné z: https://qph.fs.quoracdn.net/main-qimg-958a1182c926442eae2fc228920f38f4.

[12] Chollet, F. a kol.. *Keras: the Python deep learning API* [online]. [cit. 2021-01-10]. Dostupné z: https://keras.io/.

[13] Cs231n. *CS231n Convolutional Neural Networks for Visual Recognition* [online]. Dostupné z: https://cs231n.github.io/assets/nn1/neuron.png.

[14] Cs231n. *CS231n Convolutional Neural Networks for Visual Recognition* [online]. Dostupné z: https://cs231n.github.io/assets/nn1/neuron_model.jpeg.

[15] Cs231n. *Neural networks. CS231n Convolutional Neural Networks for Visual Recognition* [online]. Stanford University [cit. 2021-01-05]. Dostupné z: https://cs231n.github.io/neural-networks-1/.

[16] Database Systems Lab Indian Institute of Science. *Anguli: Synthetic Fingerprint Generator* [online]. [cit. 2021-04-11]. Dostupné z: https://dsl.cds.iisc.ac.in/projects/Anguli/.

[17] Deep Learning Demystified. *Understanding Optimizers* [online]. Deep Learning Demystified [cit. 2021-01-08]. Dostupné z: https://deeplearningdemystified.com/article/fdl-4.

[18] Doshi, S. *Various Optimization Algorithms For Training Neural Network* [online], 13. januára 2019 [cit. 2021-01-08]. Dostupné z: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6.

[19] Drahanský, M., Orság, F. a kolektív. *Biometrie*. 1. vyd. [Brno: M. Drahanský], 2011. ISBN 978-80-254-8979-6.

[20] Du Vivier, A. *Atlas of clinical dermatology*. 4th ed. Elsevier Saunders, 2013. ISBN 978-0-7020-3421-3.

[21] Dvořák, J. *Synthetic Fingerprint Generation Using GAN*. Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22574/.

[22] Dyre, S. a Sumathi, C. P. *Fingerprint classification based on novel directional features using neural network* [online]. [cit. 2020-12-30]. Dostupné z: https://d3i71xaburhd42.cloudfront.net/d663bf88737b70b2b9f02943d71c599e8f009643/2-Figure1-1.png.

[23] Facebook's AI Researcher. *PyTorch* [online]. Facebook [cit. 2021-05-04]. Dostupné z: https://pytorch.org.

[24] Fbi. *IAFIS | Federal Bureau of Investigation* [online]. Federal Bureau of Investigation [cit. 2020-12-30]. Dostupné z: https://www.fbi.gov/services/information-management/foipa/privacy-impact-assessments/iafis.

[25] Fbi. *Next Generation Identification (NGI) | Federal Bureau of Investigation* [online]. Federal Bureau of Investigation [cit. 2020-12-30]. Dostupné z: https://www.fbi.gov/services/cjis/fingerprints-and-other-biometrics/ngi.

[26] Fogel, I. a Sagi, D. Gabor filters as texture discriminator. *Biological Cybernetics*. 1989, zv. 61, č. 2. DOI: 10.1007/BF00204594. ISSN 0340-1200. Dostupné z: http://link.springer.com/10.1007/BF00204594.

[27] Frankenfield, J. *Artificial Neural Network (ANN)* [online]. [cit. 2021-01-05]. Dostupné z: https://www.investopedia.com/terms/a/artificial-neural-networks-ann.asp.

[28] FSS:ENS118 Uvedení do biologie II. *Mechorosty (Bryophyta)* [online]. Masarykova Univerzita Brno, Fakulta sociálních studií [cit. 2021-05-11]. Dostupné z: https://is.muni.cz/el/fss/jaro2011/ENS118/BryophytaHEN.pdf?lang=en.

[29] Galton, F. *Finger Prints*. Macmillian and CO. and New York, 1892. Dostupné z: http://www.biometricbits.com/Galton-Fingerprints-1892.pdf.

[30] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. Dostupné z: http://www.deeplearningbook.org.

[31] Google. *TensorFlow* [online]. Google Brain Team [cit. 2021-01-10]. Dostupné z: https://www.tensorflow.org/.

[32] Haykin, S. a Kosko, B. GradientBased Learning Applied to Document Recognition. In: *Intelligent Signal Processing*. 2001, s. 306–351. DOI: 10.1109/9780470544976.ch9.

[33] Haykin, S. S. *Neural networks and learning machines*. 3rd ed. Pearson Education, c2009. ISBN 978-0-13-147139-9.

[34] He, K., Zhang, X., Ren, S. a Sun, J. *Deep Residual Learning for Image Recognition* [online]. arXiv:1512.03385v1, 10. decembra 2015 [cit. 2021-01-08]. Dostupné z: https://arxiv.org/pdf/1512.03385.pdf.

[35] Heidari, M., Kanich, O. a Drahanský, M. Processing of fingerprints influenced by skin diseases. In: *Hand-Based Biometrics: Methods and Technology*. The Institution of Engineering and Technology, 2018, s. 135–168. IET Book Series on Advances in Biometrics. ISBN 978-1-78561-224-4. Dostupné z: https://www.fit.vut.cz/research/publication/11693.

[36] Häggström, M. *Layers of the epidermis. 2010* [online]. [cit. 2020-12-19]. Dostupné z: https://commons.wikimedia.org/wiki/File:Epidermal_layers.svg.

[37] Jackson, A. R. a Jackson, J. M. *Forensic science*. 3rd edition. Prentice Hall, 2011. ISBN 0273738402 (pbk.).

[38] Jain, A. K., Flynn, P. a Ross, A. A. *Handbook of Biometrics*. Springer US, 2008. ISBN 978-0-387-71040-2.

[39] Jirásková, M. *Bradavice – věčný problém a co s nimi. Interní medicína pro praxi* [online]. [cit. 2021-01-12]. Dostupné z: https://www.solen.cz/pdfs/int/2009/12/11.pdf.

[40] Kanich, O. *Fingerprint damage simulation: a simulation of fingerprint distortion, damaged sensor, pressure and moisture*. Saarbrücken: LAP Lambert Academic Publishing, 2014. ISBN 978-3-659-63942-5.

[41] Kanich, O. *Research in Fingerprint Damage Simulations*. Brno, CZ, 2019. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/phd-thesis/832/.

[42] Kanich, O. a Drahanský, M. State of the art in fingerprint recognition. In: *Hand-Based Biometrics: Methods and Technology*. The Institution of Engineering and Technology, 2018, s. 83–110. IET Book Series on Advances in Biometrics. ISBN 978-1-78561-224-4. Dostupné z: https://www.fit.vut.cz/research/publication/11692.

[43] Kaushik, A. *Understanding the VGG19 Architecture* [online]. [cit. 2021-01-08]. Dostupné z: https://iq.opengenus.org/vgg19-architecture/.

[44] Mahdianpari, M., Salehi, B., Rezaee, M., Mohammadimanesh, F. a Zhang, Y. Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery. *Remote Sensing*. Júl 2018, zv. 10, s. 1119. DOI: 10.3390/rs10071119.

[45] Maltoni, D. *Handbook of Fingerprint Recognition*. 2. vyd. London: Springer London, 2009. ISBN 978-1-84882-253-5.

[46] Matzler, K., Bailom, F., Eichen, S. F. von den a Anschober, M. *Digitálna Disrupcia*. Bratislava: Slovenská inovačná a energetická agentúra, 2018. ISBN 978-80-88823-67-4.

[47] McCulloch, W. S. a Pitts, W. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5. zv. 1943, č. 5, s. 115–133. DOI: https://doi.org/10.1007/BF02478259.

[48] MeDitorial +. *Psoriáza* [online]. Meditorial+ [cit. 2021-01-12]. Dostupné z: https://www.revmaticke-nemoci.cz/dokumenty/psoriaza.pdf.

[49] Mehlig, B. *Artificial Neural Networks* [online]. version 2. ArXiv.org, február 2019. Dostupné z: https://arxiv.org/abs/1901.05639.

[50] Mehrotra, K., Mohan, C. a Ranka, S. *Elements of Artificial Neural Networks*. USA: MIT Press, 1997. ISBN 0-262-13328-8.

[51] MetaCentrum. *O MetaCentru VO* [online]. GeeksforGeeks [cit. 2021-04-23]. Dostupné z: https://metavo.metacentrum.cz/cs/about/index.html.

[52] Migdal, P. a Jakubanis, R. *Keras or PyTorch as your first deep learning framework* [online]. [cit. 2021-05-04]. Dostupné z: https://deepsense.ai/keras-or-pytorch/.

[53] Mlk. *Neural Network Primitives Final Part 4 – Modern Artificial Neuron* [online], 22. júna 2019 [cit. 2021-01-07]. Dostupné z: https://machinelearningknowledge.ai/artificial-neuron/.

[54] Mlk. *Neural Network Primitives Part 1 – McCulloch Pitts Neuron Model (1943)* [online], 7. januára 2019 [cit. 2021-01-07]. Dostupné z: https://machinelearningknowledge.ai/mcculloch-pitts-neuron-model/.

[55] Mlk. *Neural Network Primitives Part 2 – Perceptron Model (1957)* [online], 23. mája 2019 [cit. 2021-01-07]. Dostupné z: https://machinelearningknowledge.ai/neural-network-perceptron/.

[56] Mlk. *Neural Network Primitives Part 3 – Sigmoid Neuron* [online], 12. júna 2019 [cit. 2021-01-07]. Dostupné z: https://machinelearningknowledge.ai/sigmoid-neuron/.

[57] Munakata, T. *Fundamentals of the New Artificial Intelligence: Neural, Evolutionary, Fuzzy and More (Texts in Computer Science)*. 2nd ed. Springer, 2009. ISBN 978-1-84628-838-8.

[58] Neurohive. *VGG16 – Convolutional Network for Classification and Detection* [online]. Neurohive [cit. 2021-01-08]. Dostupné z: https://neurohive.io/wp-content/uploads/2018/11/vgg16-1-e1542731207177.png.

[59] Neurohive. *VGG16 – Convolutional Network for Classification and Detection* [online]. Neurohive, 20. novembra 2018 [cit. 2021-01-08]. Dostupné z: https://neurohive.io/en/popular-networks/vgg16/.

[60] Nielsen, M. *CHAPTER 5: Why are deep neural networks hard to train?* [online]. [cit. 2021-05-12]. Dostupné z: http://neuralnetworksanddeeplearning.com/images/tikz36.png.

[61] Pawangfg. *Keras: the Python deep learning API* [online]. GeeksforGeeks [cit. 2021-01-10]. Dostupné z: https://www.geeksforgeeks.org/vgg-16-cnn-model/.

[62] Simonyan, K. a Zisserman, A. *Very deep convolutional networks for large-scale image recognition* [online]. ArXiv preprint arXiv:1409.1556, 10. apríla 2015 [cit. 2021-01-08]. Dostupné z: https://arxiv.org/pdf/1409.1556.pdf.

[63] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. et al. *Going deeper with convolutions* [online]. arXiv:1409.4842v1, 17. septembra 2014 [cit. 2021-01-08]. Dostupné z: https://arxiv.org/pdf/1409.4842.pdf.

[64] Szegedy, C., Vanhoucke, V., Ioffe, S. a Shlens, J. *Rethinking the Inception Architecture for Computer Vision* [online]. arXiv:1512.00567v3, 11. decembra 2015 [cit. 2021-01-08]. Dostupné z: https://arxiv.org/pdf/1512.00567.pdf.

[65] University of Bologna. *Biometric System Laboratory* [online]. [cit. 2021-04-11]. Dostupné z: http://biolab.csr.unibo.it/research.asp?organize=Activities&select=&selObj=12&pathSubj=111%7C%7C12&.

[66] Šalko, M. *Detekce a klasifikace poškození otisku prstu s využitím neuronových sítí*. Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22567/.

[67] Watson, C. I. a Wilson, C. L. NIST special database 4. Fingerprint Database. National Institute of Standards and Technology. Marec 1992. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.2000&rep=rep1&type=pdf.

[68] Wellller, R. P. J. B. *Clinical dermatology*. 4th ed. Blackwell, 2008. ISBN 978-1-4051-4663-0.

[69] Win, K. N., Li, K., Chen, J., Viger, P. F. a Li, K. Fingerprint classification and identification algorithms for criminal investigation: A survey. *Future Generation Computer Systems*. 2020, zv. 110, s. 758 – 771. DOI: https://doi.org/10.1016/j.future.2019.10.019. ISSN 0167-739X. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0167739X19315109.

[70] Yalçın, O. G. *4 Pre-Trained CNN Models to Use for Computer Vision with Transfer Learning* [online], 23. septembra 2020 [cit. 2021-01-08]. Dostupné z: https://towardsdatascience.com/4-pre-trained-cnn-models-to-use-for-computer-vision-with-transfer-learning-885cb1b2dfc.

[71] ČVUT. *Math Tutor - Functions - Theory - Elementary Functions* [online]. Katedra Matematiky, ČVUT [cit. 2021-01-06]. Dostupné z: https://math.fel.cvut.cz/mt/txtb/4/txc3ba4l.htm.

[72] Šetelíková, A. *Pojem a podstata Daktyloskopie*. Praha, CZ, 2012. Diplomová práce. Univerzita Karlova v Praze, Právnická fakulta. Dostupné z: https://is.cuni.cz/webapps/zzp/download/120108677/?lang=cs/.

[73] Štork, J. *Dermatovenerologie*. 2. vyd. Praha: Galén, c2013. ISBN 978-80-7262-898-8.

Príloha A

Obsah priloženého DVD

Priložené DVD obsahuje:

- **documentation** - adresár s textom diplomovej práce písanej v LaTeX šablóne vrátane obrázkov
- **fingerprint_detection** - adresár so zdrojovým kódom
- **diploma_thesis.pdf** - textová časť v PDF
- **virtual_env.zip** - komprimovaný súbor, ktorý obsahuje virtuálne prostredie pre python.
- **ReadMe.md** - súbor s informáciami

Príloha B

Kód modelu v knižnici Keras

```
def create_model(debug=True, number_of_outputs=2, shape=(128, 128, 1)):
  ...
  model = Sequential()

  # 112
  model.add(Conv2D(64, (3, 3), activation='relu', input_shape=shape, data_format="channels_last"))
  model.add(Conv2D(64, (3, 3), activation='relu'))
  model.add(MaxPooling2D((2, 2), strides=(2, 2)))

  # 56
  model.add(Conv2D(128, (3, 3), activation='relu'))
  model.add(Conv2D(128, (3, 3), activation='relu'))
  model.add(MaxPooling2D((2, 2), strides=(2, 2)))

  # 28
  model.add(Conv2D(256, (3, 3), activation='relu'))
  model.add(Conv2D(256, (3, 3), activation='relu'))
  model.add(MaxPooling2D((2, 2), strides=(2, 2)))

  # 14
  model.add(Conv2D(256, (3, 3), activation='relu'))
  model.add(Conv2D(256, (3, 3), activation='relu'))
  model.add(MaxPooling2D((2, 2), strides=(2, 2)))

  # 7
  model.add(Flatten())
  model.add(Dense(768, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(768, activation='relu'))
  model.add(Dropout(0.5))

  if (number_of_outputs == 3):
    # Klasicky model rozlisujuci 2 ochorenia
    model.add(Dense(3, activation='softmax'))
    opt = SGD(learning_rate=0.001, momentum=0.0, nesterov=False)
    model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=["accuracy"])
  else:
    # Model detekujuci len jedno ochorenie
    model.add(Dense(2, activation='sigmoid'))
    opt = SGD(learning_rate=0.001, momentum=0.0, nesterov=False)
    model.compile(loss='binary_crossentropy', optimizer=opt, metrics=["accuracy"])

  ...

  return model
```

Algoritmus B.1: model v Keras.

Príloha C

Kód modelu v knižnici PyTorch

```
class FingerprintModelBP(nn.Module):
  def ___init___(self, number_of_outputs, shape):
    super(FingerprintModelBP, self).___init___()

    self.number_of_outputs = number_of_outputs
    self.shape = shape
    self.conv1 = nn.Conv2d(1, 64, (3, 3))
    self.conv2 = nn.Conv2d(64, 64, (3, 3))
    self.relu = nn.ReLU()
    self.pool = nn.MaxPool2d(2, 2)
    self.conv3 = nn.Conv2d(64, 128, (3, 3))
    self.conv4 = nn.Conv2d(128, 128, (3, 3))
    self.conv5 = nn.Conv2d(128, 256, (3, 3))
    self.conv6 = nn.Conv2d(256, 256, (3, 3))
    self.linear1 = nn.Linear(256 * 4 * 4, 768)
    self.dropout = nn.Dropout(0.5)
    self.linear2 = nn.Linear(768, 768)
    self.softmax = nn.Softmax(dim=1)
    self.linear_two_diseases = nn.Linear(768, 3)
    self.linear_one_disease = nn.Linear(768, 2)
    self.sigmoid = nn.Sigmoid()

  def forward(self, model):
    model = self.relu(self.conv1(model))
    model = self.relu(self.conv2(model))
    model = self.pool(model)
    model = self.relu(self.conv3(model))
    model = self.relu(self.conv4(model))
    model = self.pool(model)
    model = self.relu(self.conv5(model))
    model = self.relu(self.conv6(model))
    model = self.pool(model)
    model = self.relu(self.conv6(model))
    model = self.relu(self.conv6(model))
    model = self.pool(model)
    model = model.view(model.size(0), -1)
    model = self.relu(self.linear1(model))
    model = self.dropout(model)
    model = self.relu(self.linear2(model))
    model = self.dropout(model)

    if self.number_of_outputs == 2:
      model = self.sigmoid(self.linear_one_disease(model))
    else:
      model = self.softmax(self.linear_two_diseases(model))

    return model
```

Algoritmus C.1: model v PyTorch.