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 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

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 s 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 nechť 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.

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 2021
Datum odevzdání: 18. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Cieľom diplomovej práce je naštudovať a navrhnúť vylepšenie súčasnej konvolučnej neurónovej siete pre klasifikáciu a detekciu ochorenia odtlačkov prstov. Vylepšením súčasnej konvolučnej neurónovej siete je zmena knižnice pre algoritmus učenia, detekcie a klasifikácie poškodenia odtlačkov prsta. Ďalšími vylepšeniami sú zmena modelu konvolučnej neurónovej siete a zmena aktivačnej funkcie. Zároveň bude pridané predspracovanie pomocou Gáborovho filtra. Ďalšia zmena je v oblasti prahovania. Ďalej bude zmena algoritmov pre všeobecné použitie, ktoré zjednoduší prácu pre rožšírenie tvorby databázy, samotný učiaci proces, proces pre klasifikáciu a detekciu a proces pre testovanie siete. Táto sieť bude zároveň rozšírená o novú predikciu a klasifikáciu. Konkrétne o poškodenia ekzémom, psoriázou, tlakom a vlhkosťou. Vylepšená konvolučná neurónová sieť je implementovaná pomocou PyTorch. Sieť detekuje aká časť odtlačku prsta je poškodená a túto časť vykreslí do odtlačku prsta. Zároveň pri detekcii prebieha aj klasifikácia o aký typ ochorenia alebo poškodenia odtlačku ide. Pri trénovaní siete sú použité syntetické odtlačky prstov a sú doplnené o reálne odtlačky prstov.

Abstract

The aim of the diploma thesis is to study and propose improvement of the current convolutional neural network for the classification and detection of fingerprint disease. An improvement of the current convolutional neural network is the change of library for the algorithm of learning, detecting and classifying fingerprint damage. Other improvements are to change the convolutional neural network model and a change in the activation function. At the same time, preprocessing using the Gabor filter will be added. Another change is in the area of thresholding. Next, there will be a change in general-purpose algorithms that will simplify the work for expanding database creation, the learning process itself, the classification and detection process, and the network testing process. At the same time, this network will be expanded with a new prediction and classification. Specifically the damage caused by eczema, psoriasis, pressure and moisture. The improved convolutional neural network is implemented by PyTorch. The network detects which part of the fingerprint is damaged and draws this part into the fingerprint. At the same time, the type of disease or imprint damage is classified during detection. Synthetic fingerprints are used in network training and are supplemented by real 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, 2022. 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 Ph.D. Ďalšie informácie mi poskytol Bc. Milan Šalko Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Peter Vican
18. mája 2022

Poďakovanie

Týmto by som sa chcel poďakovať pánovi Ing. Ondřejovi Kanichovi Ph.D., 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 alebo 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.

Táto práca je zameraná na trénovanie konvolučnej neurónovej siete pre detekciu a klasifikáciu poškodenia odtlačkov prstov, ktorá by mohla 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 [70], 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. Hlavným cieľom diplomovej práce je vylepšenie stávajúceho riešenia a rozšírenia o nové choroby a poškodenia odtlačkov prstov spôsobené pri snímaní.

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 vylepšenia a návrh nového klasifikátora. Kapitola 5 obsahuje implementáciu riešenia. Posledná kapitola 6 sa zameriava na dosiahnuté výsledky a testovanie.

Kapitola 2

Spracovanie odtlačkov prstov

V priebehu dňa prebehne niekoľko miliónov spracovaní a rozpoznaní 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 alebo rozpoznanie 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 2.8, poškodenie odtlačku prsta pri snímaní v kapitole 2.6 a nakoniec choroby v kapitole 2.7.

2.1 Biometria

Digitalizácia je v súčasnej dobe na veľkom vzostupe a v najbližších rokoch bude exponenciálne rásť [48]. Z tohto dôvodu bude potreba lepšieho zabezpečenia dát, prístupu do chytrého bankovníctva, prístupy do budov, prístup do áut 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] [47]

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 biometrické charakteristiky sa používajú na automatické rozpoznávanie jednotlivcov. Hlavnou výhodou týchto charakteristík je, že človek ich nemôže zabudnúť, stratiť alebo jednoducho zneužiť inou osobou. Mnoho týchto charakteristík môže veľa napovedať o zdravotnom stave jedinca a tým pádom, treba k tomu pristupovať zodpovedne a bezpečne. Pred tým, než hociktorú charakteristiku zoberieme do úvahy je nutné overiť, nakoľko daná vlastnosť spĺňa nasledujúcich deväť vlastností: [19] [43] [70]

- *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 charakteristík medzi rozdielnymi triedami (osobami). Intratriedna premenlivosť udáva veľkosť rozdielu medzi charakteristikami v jednej triede (jedinec). [42] [70]

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 [72]. 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] [72]

- *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.

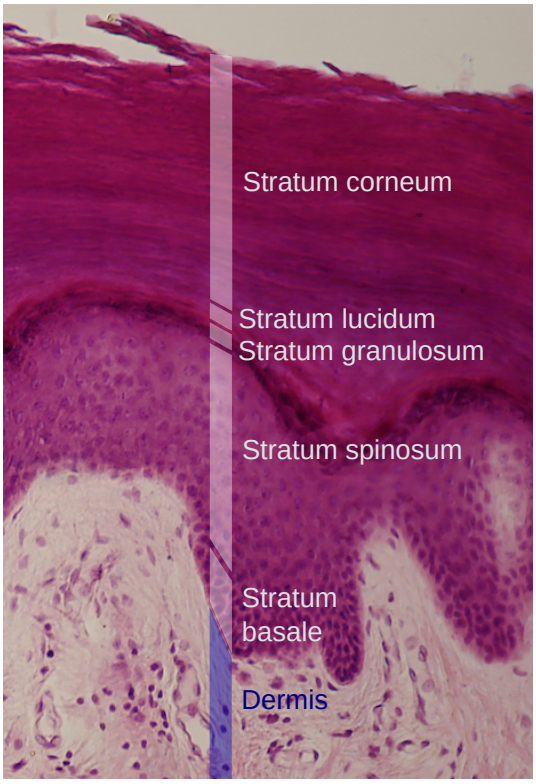
Vedný obor, ktorý sa zaoberá a popisuje štruktúru kože, choroby a jej príčiny, sa volá *dermatológia*.

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] [72]

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. [4] [20] [72]

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. [4] [20] [72]



Obrázok 2.1: Štruktúra epidermy. Zdroj: [38].

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. [4] [20] [72]

Stratum lucidum je tenká vrstva epidermis obsahujúca dve až tri vrstvy buniek. Tvorí dôležitú bariéru, ale existuje len na dlaniach a chodidlách. [4]

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: [20] [72]

- *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, označované ako papilárne línie. Dermu rozdeľujeme na: [76]

- *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. [72]

2.3 Odtlačky prstov

Odtlačky prstov sú podľa dochovaných čínskych textov používané viac než 3000 rokov [73]. 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. [39] [44]

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

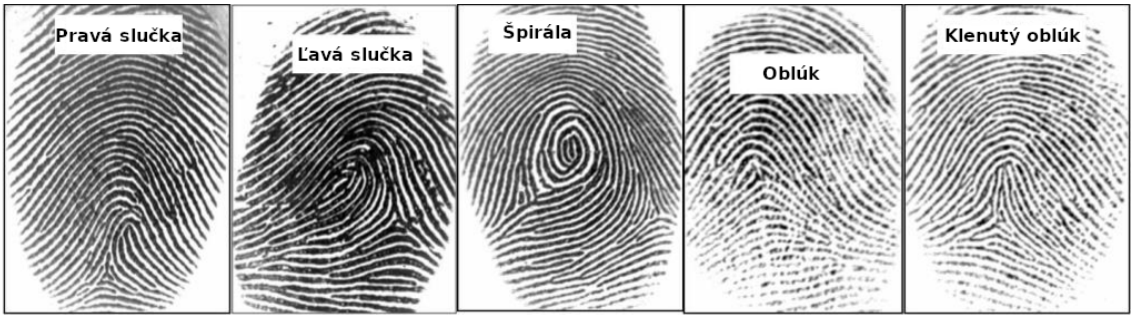
Papilárne línie, ako bolo spomenuté v podkapitole 2.2.2, sú výstupky z epidermy, ktoré sú tvorené z papíl a vrásnia kožu. Ich výskyt je na rukách a dlaniach. Ich výška môže byť 0,1 až 0,4 mm a dosahujú hrúbky 0,2 mm až 0,5 mm. Vyvinuli sa pre lepšie uchopenie predmetov, aby sa nekĺzali v rukách. Na ich vyústení sú potné žľazy, ktoré je vidieť pri kvalitnejšom obrázku ako malé bodky. Papilárne línie na konci brušiek prstov tvoria odtlačky prstov. [42] [43] [70]

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 [47], 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. [42] [43] [47] [70]

2.3.2 Klasifikácia odtlačkov

Porovnať dva odtlačky nie je jednoduchá úloha, ako to môže vyzerať na prvý pohľad. Najväčším priekopníkom v klasifikácii bol sir Francis Galton, ktorý v diele [30], 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 [25]. V roku 2017 FBI vydal ďalší systém NGI *(Next Generation Identification)*, ktorý sa nelimituje iba na odtlačky prstov [26]. 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.



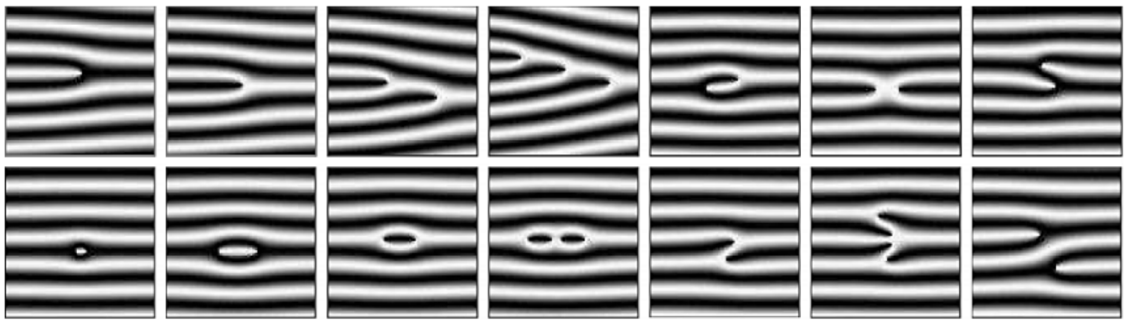
Obrázok 2.2: Rozšírené triedy klasifikácie. Upravené zo zdroja: [22].

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] [42] [44]

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 [47] pomenované na počesť sira Francisa Galtona. Galton popísal viac ako 100 druhov markantov [47].

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: [44].

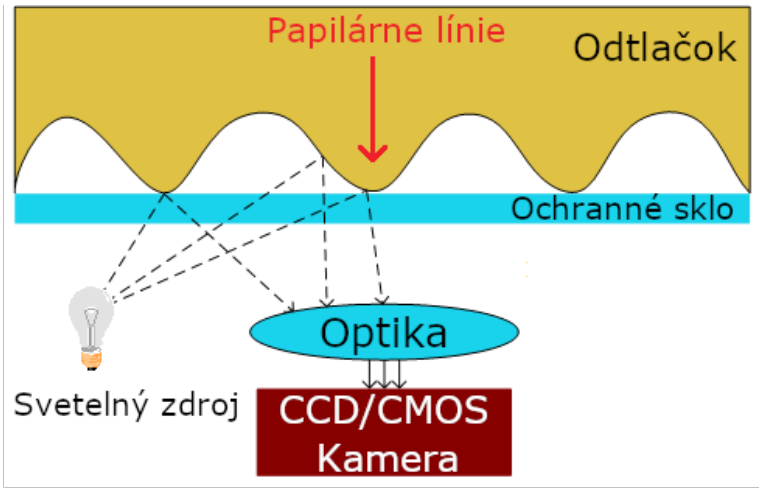
bočný kontakt, bod, interval, jednoduchá slučka, dvojitá slučka, jednoduchý most, dvojitý most a priesečná línia.

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] [40]

2.4 Získavanie odtlačkov prstov

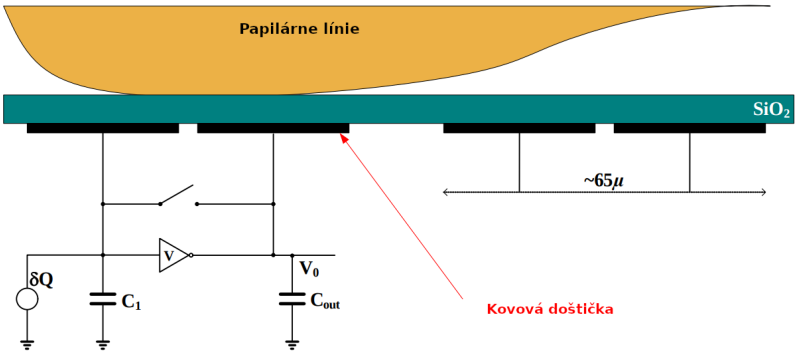
Pre potreby rozpoznania odtlačkov prstov je potrebné nejakým spôsobom získať odtlačky prstov do digitálnej podoby [42]. Existujú rôzne spôsoby pri získavaní, napríklad oskenovaný, latentný odtlačok prsta. Avšak existuje pohodlnejší spôsob, ako získať odtlačok prsta do počítača, ktorým je priame nasnímanie odtlačku prsta pomocou senzora pripojeného USB (*universal serial bus*) portom [42]. V podkapitole budú popísané dve najpoužívanejšie technológie. Konkrétne sa jedná o optickú a kapacitnú technológiu.

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 papilárných línií dotýkajú skla a údolia sú v diaľke. Zo svetelného zdroja, najčastejšie LED (*Light-Emitting Diode*), dopadá paprsok na povrch prstu a je odrazený od výstupkov a absorbovaný údoliami. Odrazené paprsky potom sníma LED CCD (*Charged Coupled Device*) / CMOS (*Complementary Metal Oxide Semiconductor)* 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] [42]



Obrázok 2.4: Optická technológia, upravené [42].

Pri **kapacitnej technológii** sa používa senzor, ktorý je vytvorený z matice mikrokapacitých plôch. Tieto plochy sú uložené na čipe. Senzor funguje tak, že pri položení prsta na senzor vzniká elektrický náboj medzi prstom a jednou z plôch. Sila náboja je väčšia, ak plocha sa dotýka papilárnych línií. Sila náboja slabne, ak sa na plochu prst oprie tak, že je tam údolie. Vďaka rozdielnym nábojom je možné nasnímať odtlačok prsta týmto senzorom. Na nasledujúcom obrázku 2.5 sa nachádza ukážka takého senzora. [42]



Obrázok 2.5: Kapacitná technológia, upravené [43].

2.5 Rozpoznávanie a spracovanie

Po získaní odtlačku prstu metódami, ktoré sme si popísali vyššie, je potrebné vysvetliť proces spracovania odtlačku prstu. Proces spracovania 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. Novodobé senzory sa snažia znížiť podvrhnutie senzora využitím detekcie 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. Tieto úpravy slúžia k vylepšeniu obrazu odtlačku prstu. 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 línie čierne a údolia biele. Následne prebieha proces ztenšenia markantov, kde sa pre tento účel používajú línie. V tomto kroku sú línie 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. Určenie polohy markantov je jedna z viacerých metód na rozpoznanie markantov nasnímaných senzorom a uložených v podobe šablóny v zariadení. Proces rozpoznania markantov nie je jednoduchý, pretože odtlačok prsta nie je zakaždým nasnímaný rovnako. Preto je potrebné brať do úvahy faktory, ktoré môžu byť napríklad posunutie, rotácia, tlak kože a šum. Posledná fáza je detekcia a extrakcia markantov. [19] [43] [47]

Obrázok 2.6: Proces spracovania odtlačkov prstov. Upravené, zdroj: [43].

2.6 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.

2.6.1 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.

2.6.2 Tlak a vlhkosť

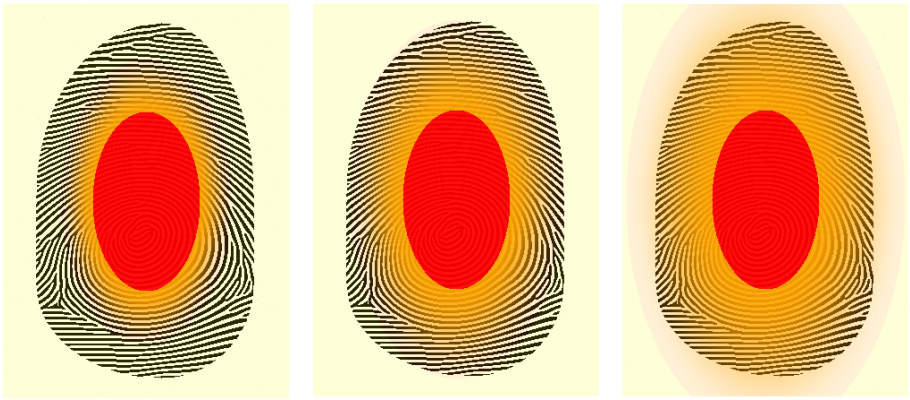
Možným poškodením pri snímaní je **tlak**. Tlak sa môže považovať za úmyselné poškodenie odtlačku prsta pri snímaní. Pokiaľ sa na senzor pritlačí, tak to spôsobuje zosilenie papilárnych línií. Papilárne línie pri veľmi veľkom tlaku zosilia natoľko, že ich nebude možné rozpoznať. Pri nízkom tlaku hrozí, že niektoré papilárne línie sa nenasnímajú vôbec alebo sa nepodarí odtlačok prsta vôbec nasnímať [43] **Vlhkosť** podobne ako tlak upravuje rôznu hrúbku a kontrast papilárnych línií [43]. Na obrázku 2.7 je vidieť syntetický odtlačok prsta, na ktorom boli prevedené tieto poškodenia.

Obrázok 2.7: Tlak a vlhkosť. Zdroj: Vygenerovaný pomocou SFinGe.

2.6.3 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 neortogonálny tlak prsta na snímač. Je to zapríčinené veľkou elasticitou kože. Samotné poškodenie zvyčajne nenarobí veľké škody pri zmene polohy markantov alebo pri zmene vzdialenosti markantov. Pri tomto poškodení majú hlavne problém algoritmy na rozponávanie odtlačkov prstov. Príklad takéhoto algoritmu môže byť algoritmus, ktorý rozpoznáva na základe polohy markantov. Pri tomto algoritme sa stávajú

problémy, že je problém s identifikáciou osôb. Prípady takýchto skreslení popisuje obrázok 2.8, na ktorom sú vyznačené rôzne oblasti skreslenia. Z obrázka 2.8 sú vidieť tri oblasti. Červená oblasť znázorňuje silné zatlačenie prsta a v tejto oblasti nemôže prísť k deformácii kože. Žltá oblasť znázorňuje oblasť, kde je tlak nízky natoľko, že koža je maximálne deformovaná. Oranžová oblasť kombinuje predchadzajúce dve oblasti. Čím je intenzita oranžovej oblasti väčšia, tým je skreslenie nižšie. [43]



Obrázok 2.8: Rôzne oblasti skreslenia. Zdroj: [43].

2.7 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 znižujú použitie odtlačku prsta ako biometrickú charakteristiku.

2.7.1 Bradavice

Táto podkapitola hodne vecí preberá z bakalárskej práce [70]. 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) [41] [77].

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. [41]

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. [41]

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

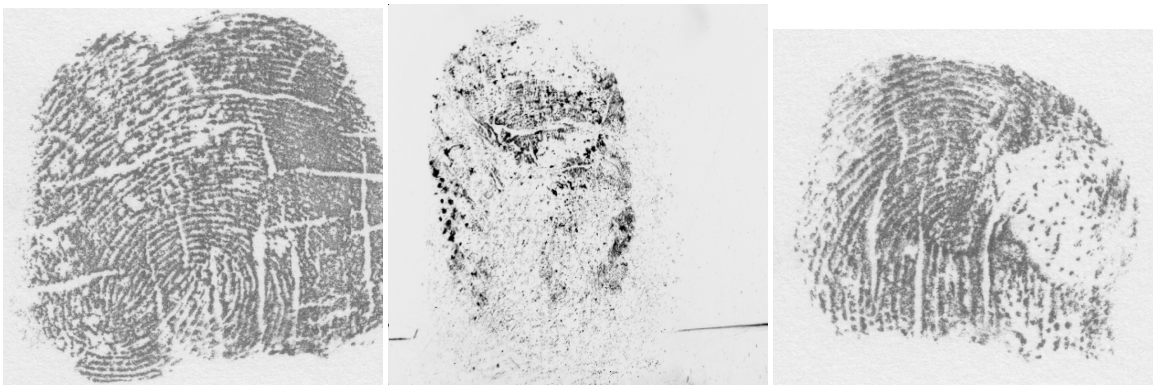
Obrázok 2.9: Bradavice na odtlačkoch prstov. Zdroj: Interná databáza skupiny STRaDe.

vôbec prejaviť pri menších bradaviciach. Poškodenie touto chorobou nie je až tak závažné. Na zvyšku odtlačku prsta sú zreteľne vidieť papilárne línie. [37]

2.7.2 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]



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

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

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. [37]

2.7.3 Atopický ekzém

Atopický ekzém (atopická dermatitída) sa označuje za celoživotné ochorenie. Má dve fázy. Prvou fázou je obdobie kľudu, kde dochádza k recesii. Druhá fáza je exacerbácia, kde sa naopak znova prejavujú príznaky tohto ochorenia sprevadzajúce silným výskytom ekzémov. Toto ochorenie je kožný zápal, ktorý má alergický pôvod a v niektorých prípadoch je kontinuálne. Ochorenie vzniká pri zníženej hydratácie pokožky. Koža pri zvýšenej suchosti pokožky je menej odolná proti posôbeniu spúšťajúcich faktorov. [23]

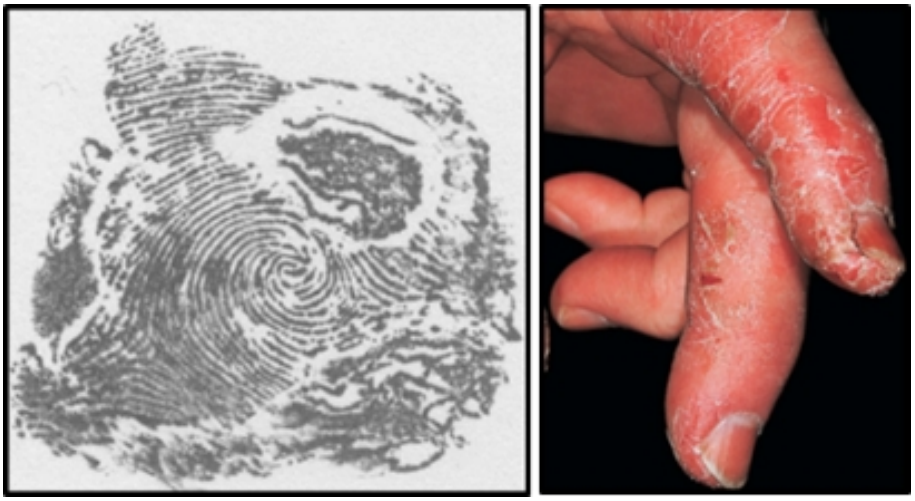
Pri rozvoji atopického ekzému sa podieľajú faktory ako je napríklad genetika, defekt kožnej bariéry, imunologická odpoveď a iné. Toto ochorenie zhoršujú alergény, mikróby, dráždivé látky a mnoho ďalšieho. Toto ochorenie postihuje rôzne vekové kategórie. Príznakmi atopického ekzému je začervenanie a svrbacie šupinaté ložiská. U deti sa vyrážka vytvorí najviac v ohybových miestach končatín, konkrétne zápästie, lakťová a zákolenná jamka. Taktiež sa môže objaviť na tvári a krku. Medzi najtypickejšie príznaky tohto ochorenia sú napríklad zvýšená suchosť a celkové podráždenie pokožky, svrbenie a opuch, odlupujúce sa plochy a následné chrasty. Ochorenie je znázornené na obrázku 2.11. Prvý zľava popisuje ranné štádium, v strede pokročilé štádium a úplne vpravo popisuje chronické štádium atopického ekzému. [23]

(a) Ranné štádium. (b) Pokročilé štádium. (c) Chronické štádium.

Obrázok 2.11: Atopický ekzém na odtlačkoch prstov. Zdroj: [33]

2.7.4 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 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 2.12 ilustruje psoriázu na končekoch prstoch a jej prejavy na odtlačkoch prstov. [50]



Obrázok 2.12: Ochorenie psoriáza. Zdroj: [2].

2.8 Syntetické odtlačky

Pri vývoji a testovaní algoritmov pre detekciu a klasifikáciu poškodenia odtlačku prsta sú potrebné trénovacie data. Pri získavaní dát sa naráža na problém, že databázy s poškodenými odtlačkami sú 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 potrebné veľké množstvo odtlačku prsta s danou chorobou. Získavanie takýchto veľkých databáz je náročné po časovej, ale aj finačnej stránke.

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, kvôli problému zneužitia tejto biometrickej charakteristiky. 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. [47]

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.

2.8.1 Generátor SFinGe

Generátor SFinGe *(Synthetic Fingerprint Generator)* je nástroj pre vytvorenie väčšej databázy odtlačkov prstov. Nástroj bol prvýkrát použitý v roku 2004 a na vývoji sa podieľala Boloňská univerzita. [9] Jedná sa o najznámejší a jeden z najstarších nástrojov pre generovanie syntetického odtlačku [42].

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. [69]

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]

2.8.2 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]

2.8.3 Generátor SyFDaS

Nástroj SyFDaS bol vyvinutý výskumnou skupinou STRaDe na FIT VUT. 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 [43]

2.8.4 Poškodzovače odtlačkov prstov

Diplomová práca využíva syntetické odtlačky na doplnenie alebo využitie celej databázy namiesto reálnych odtlačkov prstov pre choroby alebo poškodenia pri snímaní. Pre tieto učely slúžia poškodzovače odtlačkov prstov. Jedná sa o algoritmy, ktoré do čistého syntetického odtlačku prstov vygenerujú ochorenie alebo poškodenie pri snímaní.

Poškodzovače, ktoré sa používajú pre účely diplomovej práce sú nástroje vyvíjané výskumnou skupinou STRaDe. Poškodzovače pre ochorenia dyshidrózu, bradavicu a atopický ekzém sú konzolové aplikácie napísané v jazyku *Python*. Tieto poškodzovače majú výhodu, že sa dajú preniesť na rôzny operačný systém, ale chýba im grafické rozhranie. Ďalší poškodzovač, ktorý diplomová práca používa je grafická aplikácia napísaná pre operačný systém *Windows* a je napísaná v jazyku *C#*. Grafické rozhranie umožňuje nastaviť typ poškodenia, konkrétne tlak, vlhkosť a dyshidrózu. Umožňuje nastaviť intenzitu poškodenia. Poškodzovač dokáže vytvoriť do jedného odtlačku prstu všetky tri typy poškodenia naraz. Tento generátor bol napísaný v rámci diplomovej práci [66].

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. [59] Prepojenie medzi neurónmi je pomocou synapsií. Synapsie tvoria zložitejšie štruktúry a tie výtvarajú sieť pre spracovanie dát. [15]

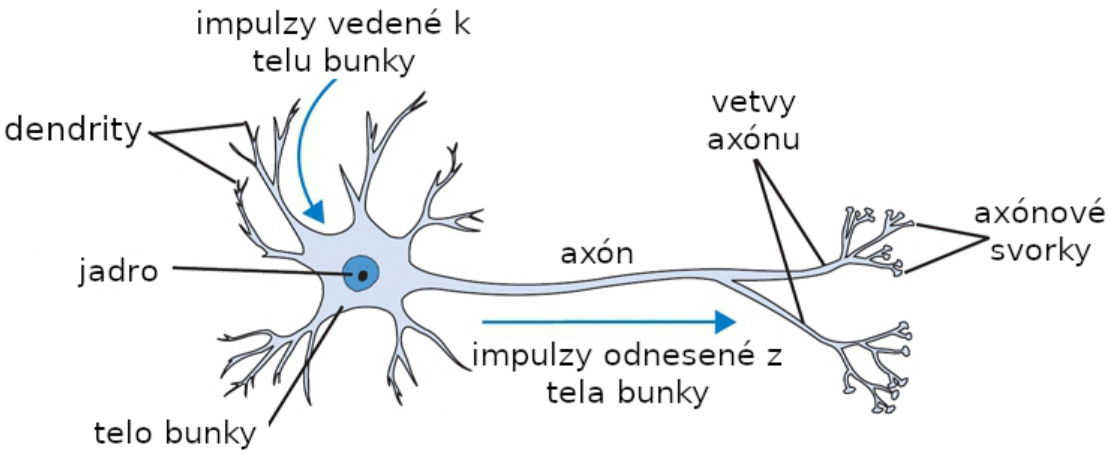
Na podobnom princípe ako bolo zmienené vyššie, fungujú umelé neurónové siete. Umelé neurónové siete je možné využiť pre rôzne úlohy spracovania dát. Môžeme využiť buď učenie s učiteľom, kde sa môže učiť rozpoznávať štruktúry v súboroch tréningových dát a zovšeobecniť, čo sa naučil. Druhá možnosť je využiť učenie bez učiteľa, ktoré môžeme použiť na analýzu veľkých súborov. 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. [51]

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 [28].

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

3.1 Biologický neurón

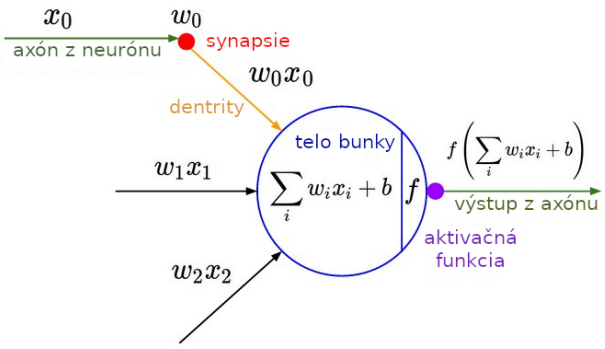
Biologický neurón sa skladá z bunkového tela, dendritov a axónov, ktoré sú vidieť na obrázku 3.1. Telo bunky sa stará o látkové premeny. Pomocou dendrít prijímajú vstupné vzruchy a pomocou axónu prenášajú vzruchy neurónov do ďalších. Axón neurónu je dlhý a tenký, vyznačujúci sa vysokým odporom a kapacitou. Preto axón môže byť modelovaný ako odporovo kapacitné prenosové vedenie. [35] 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. [52]



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 aktivovaný 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] [35] [70].

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. [75]

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

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.

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

Problémom krokovej funkcie je, že vygeneruje málo hodnôt. Tento problém 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

Po zoznámení sa s aktivačnou funkciou je potrebné pre učenie neurónových sietí vedieť, čo je to stratová funkcia. Stratová funkcia (anglicky *loss function*) je objektívna funkcia, pomocou ktorej 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. [70] 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] [70]

- **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] [70]
- **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] [70]

- **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 kvadratickej chyby. [5]

3.4.2 Funkcie straty binárnej klasifikácie

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

- **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]

3.4.3 Funkcie straty pri klasifikácii viacero tried

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

- **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 [29] 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] [70]

3.4.4 Funkcie straty regresie

Posledná podkapitola sa bude zaoberať rôznymi možnosťami pre riešenie problému 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 snížiť 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: [70]

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

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. [70]

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. [70]

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 [55] 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 [49]. V tomto diele prvýkrát popísali matematický model biologického neurónu. Model dostal podľa nich pomenovanie McCullich-Pitts. Skladá sa zo štyroch častí. [56]

Neurón je výpočtová jednotka, do ktorej vstupujú vstupné signály. Vypočítajú sa vstupné signály a výstup sa aktivuje. Neurón sa skladá z ďalších dvoch častí: [56]

- *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. [56]

Inhibičný vstup je ďalší typ vstupného signálu do neurónu. Ak je vstup zapnutý, tak neurón sa nepustí ďalej. [56]

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. [56]

Fungovanie modelu je následovné. Najprv sa 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. [56]

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. [57]

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. [57]

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. [57]

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 je odlišné 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. [57]

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. [58] 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. [58]

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. [58]

Podobne ako u perceptrónu, tak aj sigmoid má váhu spätú so vstupom. Tieto **váhy** sú na začiatku neznáme, ale sú učené behom tréningovej fázy. [58]

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. [58]

Fungovanie modelu je podobné 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.

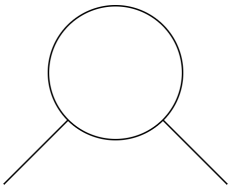
3.6.4 Moderný umelý neurón

Ako bolo spomenuté v úvode tejto podkapitoly 3.6, tak tento model je len pomenovaním autora článku [55]. 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 je podobné ako sigmoid. Rozdielom vo fungovaní modelu je práve už niekoľkokrát spomínaná aktivačná funkcia. Výstupom z tohto neurónu je rozsah daný aktivačnou funkciou.

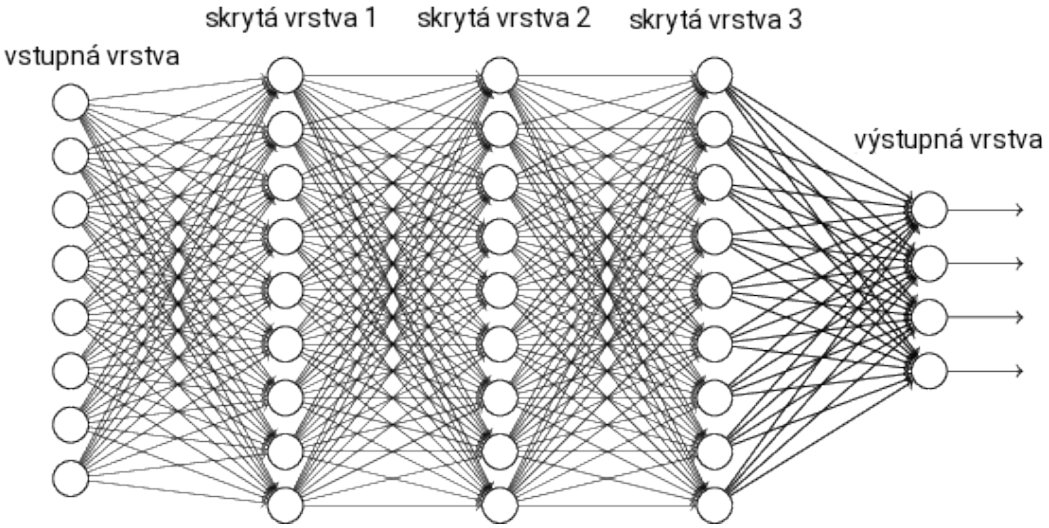
3.7 Viacvrstvové neurónové siete

V predchádzajúcej kapitole 3.6 boli popísané modely ako perceptrón, sigmoid a moderný umelý neurón. Tieto modely môžu tvoriť jednoduchú jednovrstvovú neurónovú sieť. Jednovrstvová neurónová sieť môže byť zložená dvoma spôsobmi. Prvý spôsob je, že v tejto sieti je iba jeden neurón. Toto je možné vidieť na obrázku 3.3. Druhým spôsobom zloženia jednovrstvovej siete je spojenie viacerých neurónov v jednej vrstve.



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. Vrstvy neurónových sietí sa môžu rozdeliť na vstupno-výstupné a skryté vrstvy.



Obrázok 3.4: Viacvrstvová neurónová sieť. Upravené, zdroj: [62].

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. Tieto vrstvy sú plne prepojené.

Skryté vrstvy sú medzi vstupnými a výstupnými vrstvami. Skryté vrstvy vo viacvrstvových sieťach sú plne prepojené. Plne prepojené vrstvy sa niekedy nazývaju aj ako *dense* vrstvy. Skryté vrstvy pomáhajú počítať sumačnú funkciu a počítajú aktivačnú funkciu. Na základe výsledku aktivačnej funkcie preposielajú informácie do ďalších neurónov a vrstiev. [35]

Výstupné vrstvy sú posledné v sieti. Tieto vrstvy definujú, koľko rôznych tried bude klasifikátor môcť určiť. Taktiež sú plne prepojené. [35]

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. [35] [70]

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. [70]

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 rieši napríklad konvolučné neurónové siete, ktoré sú popísané na nasledujúcej kapitole. [35] [70]

3.8 Konvolučné neurónové siete

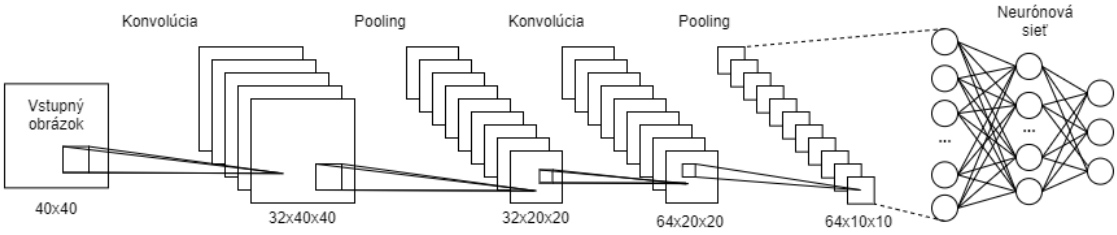
Konvolučné neurónové siete (v skratke *CNN* z anglického *Convolutional Neural Network*) sú špeciálne typy neurónových sietí pre spracovanie dát. Jej topológia je podobná mriežke. Príkladom konvolučnej neurónovej siete s 1D mriežkou môže byť meranie vlhkosti zeminy v pravidelných časových úsekoch. Druhým 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. [31] 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. [34] [70]

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. *Konvolučná vrstva* v konvolučnej sieti 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 (*subsampling*) a druhá je lokálne spriemerovaná mapa. Pooling znižuje vplyv šumu a posunu výstupu. [34] [70]

Konvolučné neurónové siete môžu používať aj *max-pooling* vrstvy. Max-pooling vrstva slúži na podzvorkovanie vstupu. Funguje tak, že vyberá maximálnu hodnotu z matice po-

krytej 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. [34] [70].



Obrázok 3.5: Architektúra konvolučnej neurónovej siete. Zdroj: [70].

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 zavislé na vypočítanej stratovej funkcii. Výstup z funkcie je prenesený do vyšších vrstiev pomocou metódy spätného šírenia. [34]

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. [34]

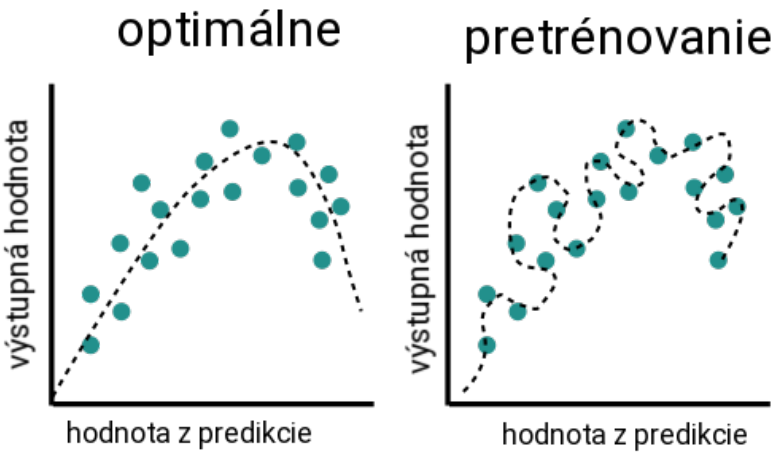
Pre učenie je dobré poznať tri pojmy, ktoré sú dôležité pri trénovaní siete. Prvým pojmom veľkosť dávky (*batch size*). Tento pojem hovorí, koľko dát alebo obrázkov sa zmestí do siete. Napríklad, ak dávka je šesťnásť, tak do modelu siete môže naraz šesťnásť obrázkov. Druhým pojmom je *iterácia*. Iterácia 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*. [64]

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]

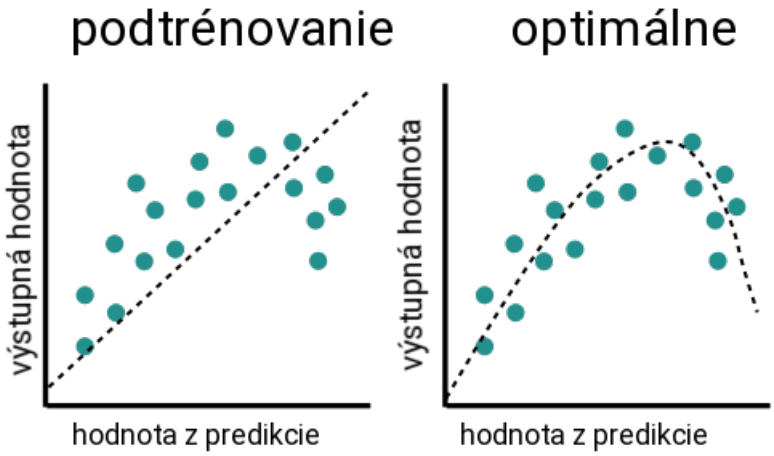
Aby model nebol pretrénovaný, tak sa môže použiť predčasné ukončenie trénovania. Toto môže spôsobiť, že model bude podtrénovaný. Pre zamedzenie pretrénovania siete sa v praxi používa validačná databáza. Jedná sa o databázu, ktorá sa vyčlení z trénovacej databázy. K validačnej databáze nemá model počas 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]



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

3.10 Predtrénované modely pre spracovanie obrazu

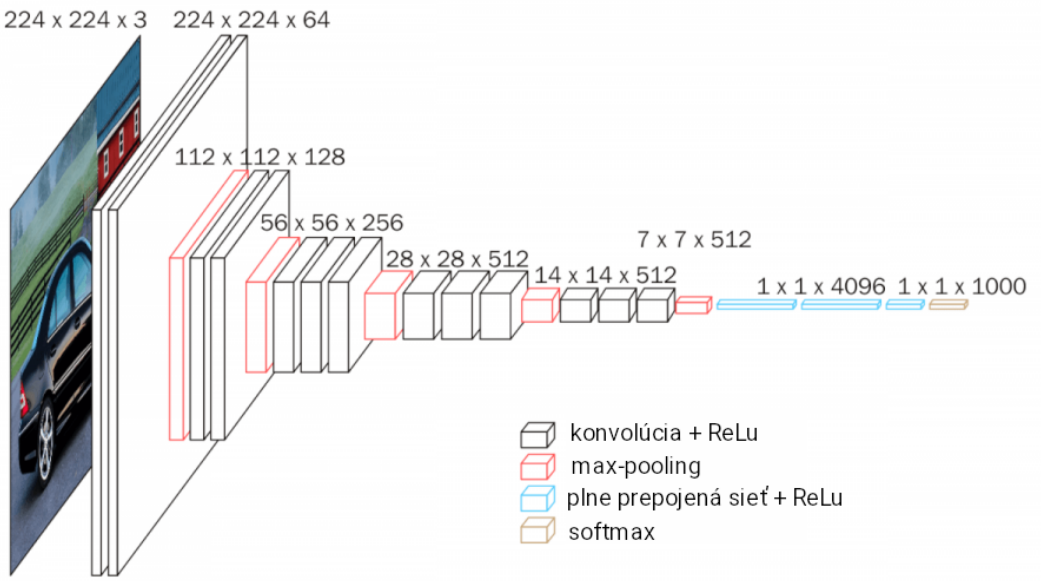
Predtrénované modely sú modely, ktoré trénoval niekto iný. Tieto modely riešia rovnaké alebo podobné problémy, ktoré vývojár alebo výskumník riešia. Použitie takýchto modelov nám uľahčí prácu pri vylepšovaní klasifikátora pre špecifické potreby. Predtrénovaný model bol trénovaný na obrovských databázach 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 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 popísané 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. [61] [65]



Obrázok 3.8: Architektúra VGG16 modelu. Upravené, zdroj: [60].

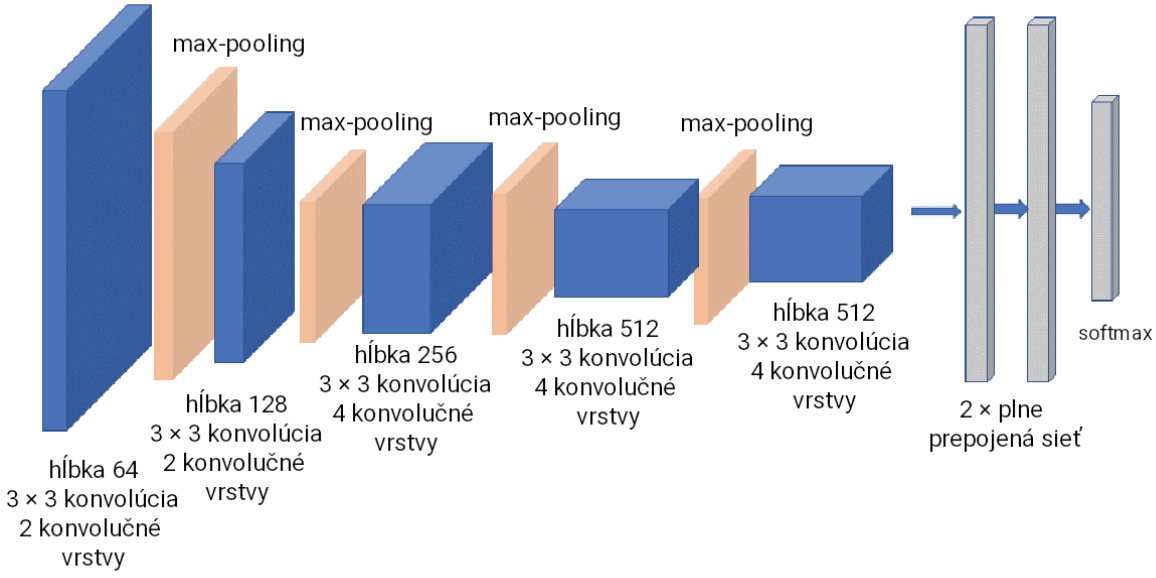
Vstupom do prvej vrstvy je pevne daný farebný obrázok 224×224 pixelov. Model obsahuje niekoľko konvolučných vrstiev, ktoré používajú filtre s veľmi malým receptívnym poľom o veľkosti 3×3. Je to najmenšia veľkosť k zachyteniu postupu okolia pixelu. Konvolučný krok je zachovaný na jeden pixel. [61] [65]

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. [61] [65]

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 softmax, ktorá je zároveň výstupnou vrstvou. [61] [65]

3.10.2 VGG19

Model VGG19 ako aj VGG16 je vytvorený K. Simoyan a A. Zisserman z Oxforskej univerzity. Má 19 vrstiev [74]. Vstup do siete je farebný obrázok o veľkosti 224 × 224 pixelov. 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 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. [45] [65]

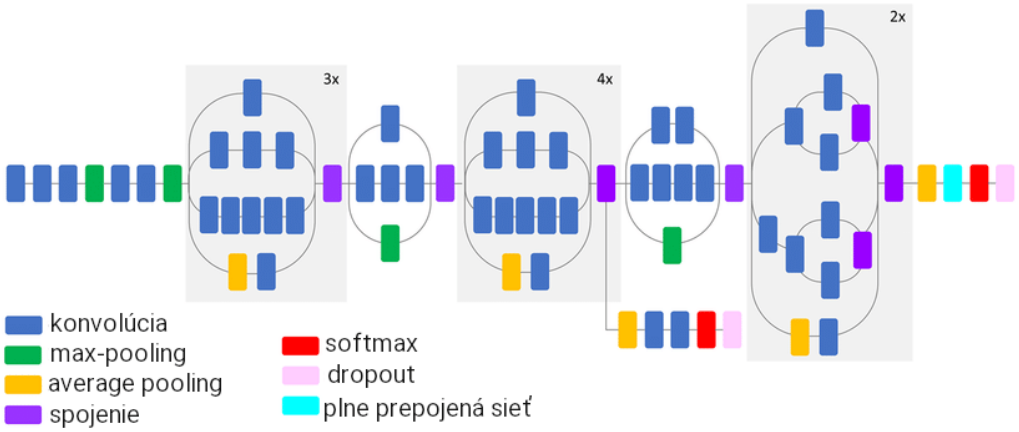


Obrázok 3.9: Architektúra VGG19 modelu. Upravené, zdroj: [11].

3.10.3 Inceptionv3 (GoogLeNet)

Inceptionv3 je konvolučná neurónová sieť, ktorá má hĺbku 50 vrstiev a je vytvorená spoločnosťou Google [67]. 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 komplexity pre parametre. 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

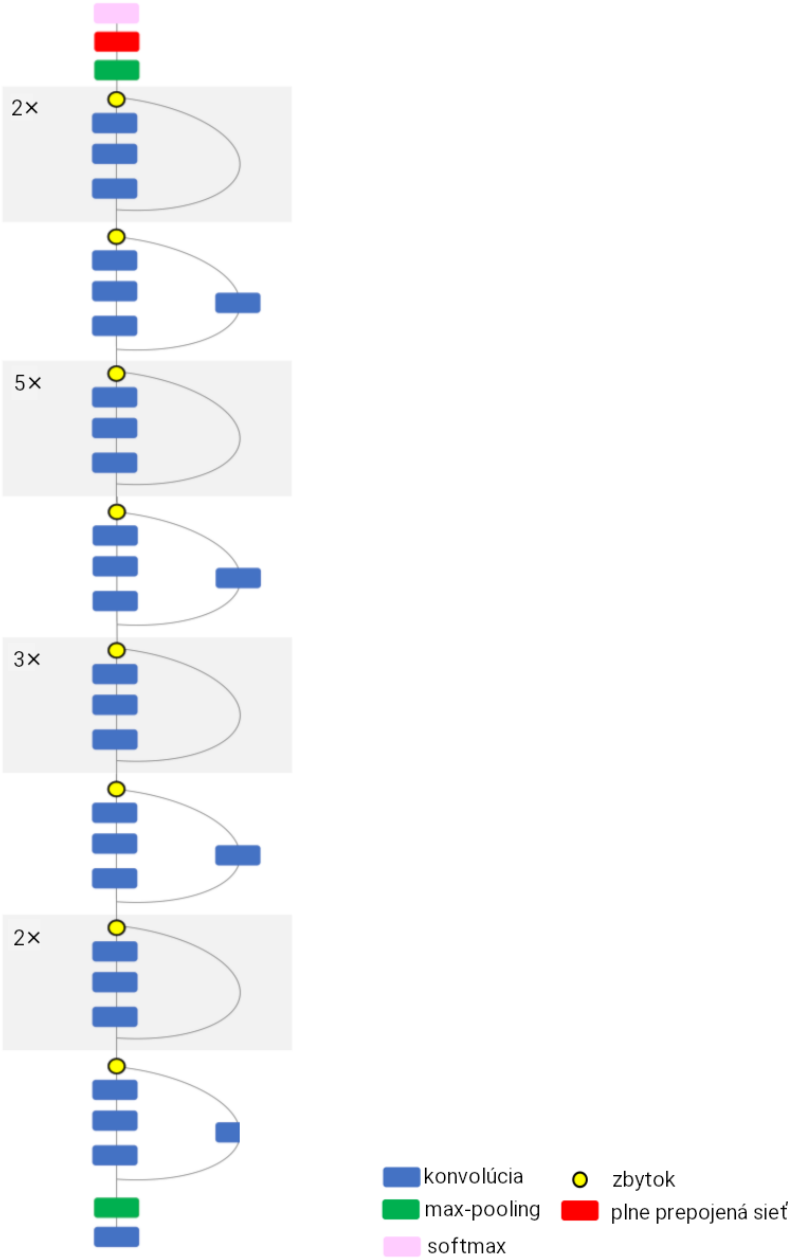
jenoduchá operácia a zabráni sa vytvoreniu úzkemu bodu. Model bol trénovaný na farebných obrázkoch s rozlíšením 299x299 pixelov. Architektúra je znázornená na obrázku 3.10. [67] [68]



Obrázok 3.10: Architektúra Inceptionv3 modelu. Upravené, zdroj: [46].

3.10.4 ResNet50

ResNet50 je konvolučná neurónová sieť, ktorá má hĺbku 50 vrstiev a je vyvinutá spoločnosťou Microsoft [74]. 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 [36]. 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. [36]



Obrázok 3.11: Architektúra ResNet50 modelu. Upravené, zdroj: [46]

Kapitola 4

Návrh riešenia

Cieľom práce je vylepšiť a rozšíriť stávajúce riešenie konvolučnej neurónovej siete pre detekciu a klasifikáciu ochorení odtlačku prsta, ktorá bola vyvinutá v bakalárskej práci [70]. Stávajúce riešenie dokáže detekovať a klasifikovať dyshidrózu a bradavicu. Nové riešenie je rozšírené o nové dve choroby. Konkrétne sa jedná o psoriázu a ekzém. Zároveň nové riešenie bude rozšírené o poškodenie odtlačku prsta pri snímaní. Týmito poškodeniami sú tlak a vlhkosť. Dokopy to tvorí rozšírenie o 4 nové typy. Celkovo sieť bude schopná rozoznávať 6 rôznych typov. Pri tomto rozšírení je zároveň cieľ znížiť pamäťovú náročnosť a čas potrebný pre detekciu a klasifikáciu. Posledné vylepšenie je vytvoriť nový klasifikátor, trénovací proces a tvorbu databázy tak, aby bolo ľahko rozšíriteľné o ďalšie ochorenia alebo poškodenia, bez veľkého zásahu používateľa. Z týchto dôvodov je postup návrhu nasledovný:

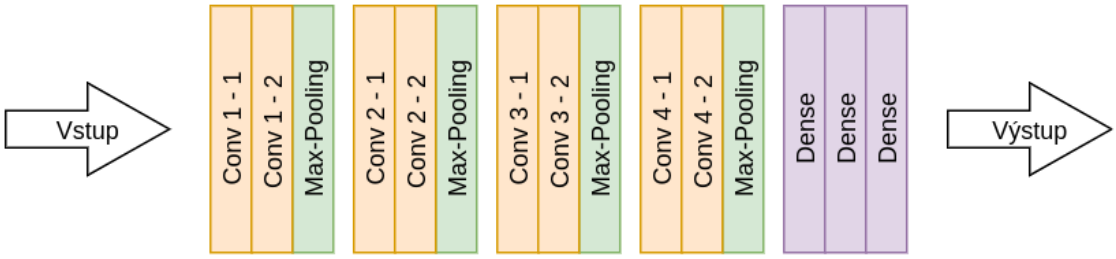
- Popis stávajúceho riešenia.
- Návrh sady vylepšení.
- Návrh databázy pre ochorenia a poškodenie pri snímaní.
- Preskúmať a navrhnúť predspracovanie pre vstup do konvolučnej siete.
- Preskúmať zapracovanie do súčasného klasifikátora alebo vytvorenie nového klasifikátora pre choroby a pre poškodenia pri snímaní zvlášť.

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 [32]. Knižnica je napísaná v jazyku Python. Toto riešenie používa internú databázu odtlačkov prstov skupiny STRaDe. Ďalšie databázy, ktoré riešenie používa je NIST SD4 DB [71] a syntetické odtlačky prstov. Na nasledujúcom obrázku 4.1 je popísaná architektúra súčasného riešenia.

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. [70] Nevýhodou modelu VGG16 je, že zaberá 528 MB [63]. Model



Obrázok 4.1: Model súčasného riešenia. Zdroj: [70]

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. [70] 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. [70]

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 [70] 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 [70]. Popis tejto aktualizácie je v podkapitole 3.5.2. Tento typ optimalizácie je vybraný v bakalárskej práci [70] 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. [70] 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. [70]

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. [70]

4.2 Návrh sady vylepšení

Návrh sady vylepšení spočíva v štyroch základných oblastiach. Prvou oblasťou bude vymenený predtrénovaný model. Následne sa rozoberú pohľady na aktivačnú funkciu. Predposlednou oblasťou bude rozobratá zmena optimalizátora a nakoniec bude popis spracovania.

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 [74]. 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 [74].

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 [70].

4.2.2 Zmena aktivačnej funkcie

V podkapitole 4.1.2 je popísaný typ používanej funkcie v aktuálnom riešení [70]. 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 [70].

4.2.3 Zmena optimalizátora a spracovania

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. Sieť v bakalárskej práci využíva pri spracovaní obrázka do siete rozmazanie *medianBlur*. Následne je použité adaptívne prahovanie. Nové riešenie plánuje dať ako rozmazanie *GaussianBlur* a adaptívne prahovanie zmeň za binárne prahovanie spojené s OTSU prahovaním.

4.2.4 Zhrnutie

V následujúcom zhrnutí je popísané, aké zmeny budú vykonané.

- Odlišná trénovacia databáza.
- Zmenené rozmazanie a prahovanie pri spracovaní obrázku.
- Trénovacia databáza s použitím Gáborovho filtra a bez použitia.
- Trénovanie s modelom z bakalárskej práce, s modelom Inceptionv3 alebo ResNet50.
- Trénovanie s aktivačnou funkciou ReLu, leaky ReLu alebo tanh.
- Zmena implementácie vytvorenia databázy, trénovania, predikcie a klasifikácie a nakoniec testovanie siete.

4.3 Návrh systému uloženia a spracovania dát

Kapitola sa bude zaoberať modulárnym riešením pre poškodenia a choroby odtlačkov prstov. Ďalej sa kapitola bude zaoberať, ako sú dáta uložené na disku. Následne bude v kapitole popísaný návrh konfiguračného súboru pre algoritmus vytvorenia databázy. Poslednou vecou, ktorou sa bude kapitola zaoberať, je návrh algoritmu pre tvorbu databázy.

4.3.1 Modulárny sytém databázy

Táto podkapitola popisuje vytvorenie modulárneho systému databázy. Ide o vytvorenie jednotného konfiguračného súboru, ktorý sa bude používať pri tvorbe databázy, pri trénovaní, predikcii a klasifikácií, a testovaní siete. Tento konfiguračný súbor je zobrazený na algoritme 4.1.

Z tohto algoritmu je vidieť, že v konfiguračnom súbore sú dve polia reťazcov. Prvé pole *damages* sú všetky ochorenia alebo poškodenia odtlačku prsta pri snímaní vrátane reťazca zdravého odtlačku prsta. V poli *damages_test* sú reťazce bez popisu zdravého odtlačku prsta. Toto pole slúži pre vytvorenie slovníka v jazyku Python pre testovanie konvolučnej neurónovej siete.

Následne bude obsahovať objekt *data*, ktorý bude obsahovať objekty. Názvy jednotlivých objektov budú pomenované podľa daného poškodenia pri snímaní alebo choroby. V týchto objektoch bude definované meno ochorenia alebo ich skratka, ktorá bude použitá

v algoritmoch. Nasledujúcim údajom bude poradové číslo triedy pri učiacom a klasifikačnom procese. Posledný údaj je farba, ktorá bude použitá pri vykresľovaní do obrázka, kde sa nachádza choroba alebo poškodenie. Vďaka tomuto konfiguračnému súboru je možné jednoducho rozšíriť sieť o ďalšie ochorenia alebo poškodenia odtlačku prsta.

```
{
        "damages": [
        "clear",
        "dyshidrosis"
        ],
        "damages_test":[
        "dyshidrosis"
        ]
        "data": {
                "dyshidrosis": {
                        "damage_name": "dyshidrosis",
                        "damage_class": 1,
                        "damage_color": "#aa8800"
                }
        }
}
```

Algoritmus 4.1: Konfiguračný súbor pre poškodenia odtlačkov prstov.

4.3.2 Návrh konfiguračného súboru pre tvorbu databázy

Tento návrh vychádza z toho, aby používateľ nemusel náročne upravovať zdrojový kód, keď bude potrebovať pridať ďalšie ochorenie alebo poškodenie pri snímaní. Na to mu bude stačiť nahrať dané odtlačky do priečinka a upraví konfiguračný súbor. Tento spôsob nastavenia parametrov ponúka veľmi efektívnu škálovateľnosť a malé úsilie používateľa pri rozšírení databázy. Návrh predpokladá s dvoma konfiguračnými súbormi. Prvý konfiguračný súbor bude všeobecný pre celé riešenie, ktorý je popísaný v kapitole 4.3.1.

Druhý konfiguračný súbor bude pre samotné generovanie množín. Ukážka tohto konfiguračného súboru je na nasledujúcom algoritme 4.2.

```
{
        "is_remove_dirs": false,
        "create_database": {
                "create_label": false,
                "source_database_path": "data/PrepareDatabase/clear",
                "destination_database_path": "data/Database",
                "class_type": "clear",
                "training_length": 7000,
                "validation_length": 7000,
                "prediction_length": 25,
                "gabor_filter": 4,
                "gabor_angle": 30,
                "fingerprint_damages_config": "config/damage_fingerprint_config.json"
        }
}
```

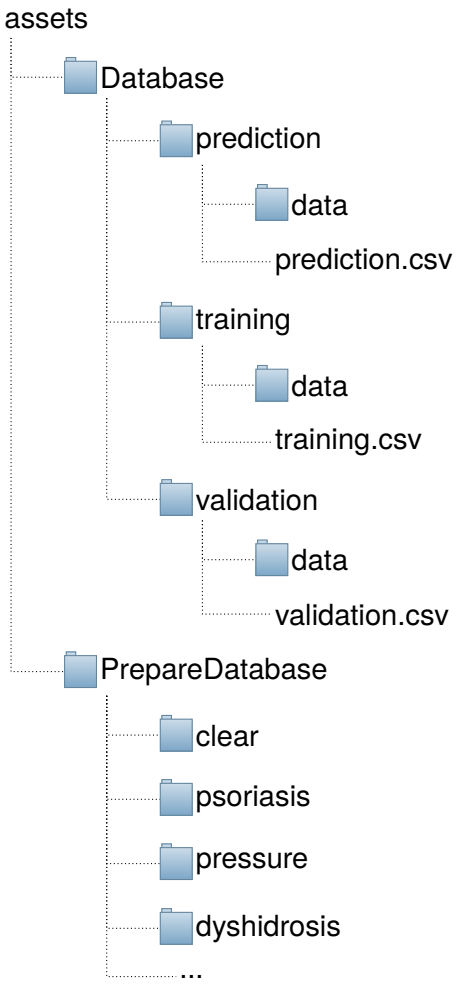
Algoritmus 4.2: Konfiguračný súbor pre vytvorenie databázy

Prvý parameter je *is_remove_dirs*, ktorý hovorí, či sa majú odstrániť staré data. Ďalší parameter je *create_database*. Tento parameter je objekt, ktorý má parametre pre algoritmus tvorby databázy. Parameter *create_label* popisuje, či sa má vytvoriť anotácia nad zdravými odtlačkami prstov. Parameter *source_database_path* popisuje zdrojovú cestu pre tvorbu databázy. Parameter *destination_database_path* popisuje cieľovú databázu pre orezané vzorky. Parameter *class_type* popisuje, ktorá trieda ochorenia alebo poškodenia sa

má vytvoriť. Parameter *training_length* popisuje počet vzorkov pre trénovací proces. Parameter *validation_length* popisuje počet vzorkov pre validačný proces. Parameter *prediction_length* popisuje počet obrázkov pre predikciu a klasifikáciu. Parameter *gabor_filter* popisuje, koľkokrát sa má zavolať Gáborov filter. Parameter *gabor_angle* určuje uhol pre Gáborov filter. Posledný parameter *fingerprint_damages_config* popisuje cestu k prvému konfiguračnému súboru.

4.3.3 Návrh štruktúry databázy

Pred samotnou implementáciou je potrebné navrhnúť štruktúru databázy. Táto štruktúra bude použitá ako pri vytváraní databázy pre trénovanie, tak aj pri samotnom trénovacom a testovacom procese. Návrh štruktúry databázy je znázornený na obrázku 4.2.



Obrázok 4.2: Štruktúra databázy na disku.

Z obrázku je definovaná primárna štruktúra, ktorá sa bude používať pri implementácii. V priečinku *PrepareDatabase* budú v jednotlivých priečinkoch prichystané choroby alebo poškodenia s anotáciami. Anotácie sú uložené v JSON (*JavaScript Object Notation*) súbore. Tento JSON obsahuje pole, v ktorom každý prvok má informácie o názvu poškodenia a dva body. Pomocou týchto bodov sa vytvorí obdĺžnik, kde sa nachádza dané poškodenie. Zároveň používateľ dostane možnosť si definovať, kde bude chcieť vytvoriť databázu. Z obrázku

je vidieť, že koreňový priečinok pre vytvorenie databázy je *assets*. Používateľ pri učiacom procese bude mať možnosť nadefinovať koreňový priečinok pre trénovanie. Pre trénovanie je koreňový priečinok *Database*.

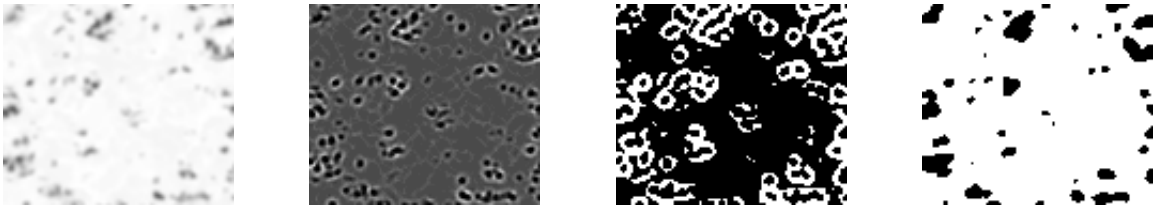
4.3.4 Návrh spracovania odtlačkov prstov

Prvou fázou pri návrhu spracovania odtlačkov prstov bolo vygenerovanie odtlačkov prstov pomocou nástroju SFinGe. Tento nástroj vygeneroval, čo najviac reálne vypadajúce nasnímané odtlačky prstov. Do týchto odtlačkov prstov boli vygenerované pomocou poškodzovača ochorenia dyshidróza, bradavica a ekzém. Následne boli pridané poškodenia odtlačkov prstov pri snímaní. Tieto poškodenia sa generovali pomocou poškodzovača z diplomovej práce [66]. V obidvoch prípadoch sa vytvorili aj anotácie k chorobám a poškodeniam. Nakoniec boli pridané reálne odtlačky prstov s ochoreniami dyshidrózou, bradavicou a psoriázou. Pri týchto odtlačkoch prstov prišlo k manuálnemu označeniu choroby a vytvoreniu anotácií.

Následuje vytvorenenie nového algoritmu pre vytvorenie databázy pre trénovanie, validáciu trénovania a klasifikáciu. Inšpiráciou pre tento algoritmus je generátor množín *TLMaker.py*, ktorý slúžil v bakalárskej práci [70] na vygenerovanie datasetu pri učení siete.

Spracovanie odtlačkov prstov pred trénovacím procesom je nasledovný. S *PrepareDatabase* sa budú brať jednotlivé priečinky s obrázkami odtlačkov prstov s anotáciami. Obrázky, ktoré nebudú obsahovať sprievodné anotácie, budú vylúčené, aby to neovplyvnilo databázu. Princípom je, že sa z anotácii spracuje a určí oblasť, kde sa nachádza ochorenie. Pre zdravé oblasti sa tiež vytvoria anotácie, ktoré sú dôležité. Implementácia anotácii zdravých oblastí je popísaná v kapitole 5.4.2.

Po určení oblasti sa označí, vyreže, odstráni sa okolie, zvýšia sa kontúry. Po tomto procese sa použije Gáborov filter s nastavenými parametrami. Po aplikovaní Gáborovho filtra je použitý Gausov filter pre rozmazanie. Nakoniec sa použije binárne prahovanie s OTSU prahovaním. OTSU prahovanie je založené na zhlukovacej metóde. Na nasledujúcom obrázku 4.3 sú zobrazené jednotlivé fázy úpravy obrázka.Po tejto fáze sa obrázky upravia na vstupnú vrstvu a uložia sa. Vďaka uloženiu do dočasných súborov nám vylepší rýchlosť učenia neurónovej siete a zároveň tento proces sa môže paralelizovať pre jednotlivé adresáre a znížiť tak fázu tvorby databázy. Tieto dočasné súbory sa môžu odstránit po skončení trénovacej fáze. Na obrázku 4.4 sú zobrazené príklady vygenerovaných ochorení a poškodení pri snímaní.



(a) Orezaná oblasť.

(b) Aplikovaný Gáborov filter.

(c) Aplikácia rozmazania a prahovania.

(d) Rozmazanie a prahovanie bez Gáborovho filtra.

Obrázok 4.3: Výrez vygenerovaného ochorenia dyshidrózy po úpravách do datasetu.

(a) Bradavica.

(b) Dyshidróza.

(c) Ekzém.

(d) Vlhkosť.

(e) Tlak.

Obrázok 4.4: Príklad rôznych vygenerovaných chorôb a poškodení pomocou poškodzovačov. Zdroj: syntetické odtlačky prstov s rôznymi poškodzovačmi.

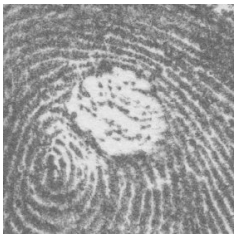
4.4 Návrh trénovacej, validačnej a testovacej množiny

Trénovacia množina pre ochorenia a poškodenia je nevyhnutnou časťou pri návrhu neurónovej siete. Pri trénovacej množine treba rozoznávať dva typy trénovacej množiny. Jedna množina dát sa používa priamo na trénovanie neurónovej siete a druhá množina sa pri trénovaní používa na validáciu trénovacieho procesu neurónovej siete. Pri tvorbe trénovacej množiny sa využívajú rôzne zdroje. Jednotlivé zdroje budú popísané v nasledujúcich podkapitolách, kde sa preberú trénovacie množiny pre ochorenia a poškodenia pri snímaní.

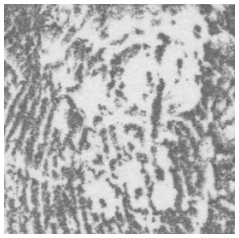
4.4.1 Trénovacia a validačná množina

Trénovacia a validačná množina pre ochorenia bude zostavená z dvoch druhov odtlačkov prstov. Prvým druhom budú syntetické odtlačky prstov. Do syntetických odtlačkov prstov, sa budú generovať ochorenia pomocou poškodzovačov, ktoré sú vytvorené v rámci skupiny STRaDe. Druhý typ odtlačku prstov budú reálne odtlačky odtlačky prstov, ktorých je len malé množstvo. U poškodení pri snímaní táto množina bude vychádzať len so syntetických odtlačkov prstov. Do týchto odtlačkov budú vygenerované poškodenia tlak a vlhkosť. Táto databáza bude poskytnutá výskumnou skupinou STRaDe. Zo syntetických a reálnych odtlačkov sa vytvorí ešte množina pre zdravé odtlačky. Táto množina slúži na to, aby vylepšila detekciu a klasifikáciu a zároveň neurónová sieť bola schopná rozpoznať zdravý odtlačok prsta. Označovanie zdravých odtlačkov prstov je jedna z množností skriptu pri vytváraní databázy.

Pri príprave trénovacej a validačnej množiny je veľmi dôležité pokrytie čo najväčšieho množstva ochorení, zdravých oblastí alebo iných poškodení ako je napríklad bežné defekty

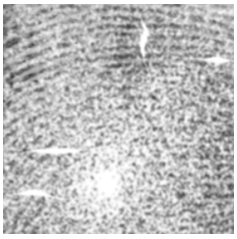


(a) Bradavica.



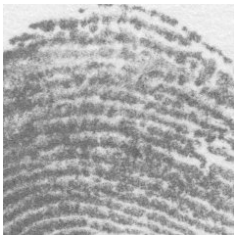
(b) Dyshidróza.

(c) Psoriáza.



(d) Ekzém.

(e) Iné poškodenie.



(f) Zdravá oblasť

(g) Vlhkosť.

(h) Tlak.

Obrázok 4.5: Príklad vstupných výrezov. Zdroj: Interná databáza skupiny STRaDe. Vygenerovaný odtlačok zo SFinGe a upravený generátorom z diplomovej práce [66].

pri snímaní. Na nasledujúcom obrázku 4.5 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 4.5. 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. Poškodzovače sú popísané v kapitole 2.8.4.

4.4.2 Množina pre testovanie

Pre testovanie kvality natrénovaného klasifikátora je nutné vytvoriť množinu na testovanie. Táto množina sa skladá podobne ako trénovacia množina zo syntetických odtlačkov a reálnych odtlačkov prstov, ktorá obsahuje odtlačky bez poškodenia, poškodenia pri snímaní a poškodenia chorobami. Táto množina je disjuktná od trénovacej a validačnej množiny. To znamená pri testovaní kvality rozpoznávania sa nepoužíva žiadny odtlačok prsta, ktorý bol použitý ako trénovací alebo validačný vzor. Túto množinu tvorí skopírovaný odtlačok prsta s anotáciou, kde sa nachádza ochorenie. Klasifikátor si načíta celý odtlačok prsta a vyrezáva oblasť z odtlačku, ktorú následne upraví pre vstup ako to je pri vytvorení trénovacej databázy. Tento úpravený výrez ide do modelu na predikciu a klasifikáciu.

4.5 Inovácie detekcie a klasifikácie o nové triedy poškodenia

V rámci práce bude vytvorená nová klasifikácia a detekcia pre poškodenia odtlačku prsta. Poškodenia sa rozdeľujú na choroby a poškodenia pri snímaní. Pozorovanou chorobou sú bradavica, ekzém, psoriáza a dyshidróza. Pozorovaným poškodením odtlačku prsta pri snímaní je tlak a vlhkosť. Tieto poškodenia sú popísané v podkapitole 2.6. Existujú dva prístupy k vytvoreniu rozšíreného klasifikátora.

- Vytvorenie nového klasifikátora.
- Detekciu a klasifikáciu pre poškodenia zakomponovať do vylepšenej konvolučnej siete.

4.5.1 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 použije 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 siete je vytvorenie obyčajného rozhodovacieho algoritmu, ktorý dostane informácie zo sietí a následne rozhodne. Táto možnosť nebude jednoducho implementovaná pomocou navrhnutého konfiguračného súboru.

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 jednovrstvovej 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 viactriedna 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.

Diplomová práca bude aplikovať rožšírenie do vylepšeného klasifikátora. Toto rozhodnutie je z vyššie uvedených lepšie. Keď sa rozšíri klasifikátor o ďalšie ochorenia, tak nemusíme riešiť veci ako, ktorý klasifikátor má väčšiu šancu na úspech. Zamedzí sa aj problém, že klasifikátor nebude mať toľko infomácií o rôznych poškodeniach, a tým pádom môže skreslovať do akej triedy to patrí.

Rovnako ako návrh konfiguračného súboru pre tvorbu databázy, učiaci proces, tak klasifikátor, bude používať konfiguračný súbor. Tento konfiguračný súbor je zobrazený na

nasledujúcom algoritme 4.3. Z konfiguračného súboru je vidieť, že klasifikátor bude využívať ďalší konfiguračný súbor pre poškodenia. V tomto konfiguračnom súbore sú infomácie o poškodeniach odtlačkov prstov, ktorý je popísaný v podkapitole 4.3.2. Vďaka tomuto bude klasifikátor, ako ostatné časti nezávislé na zásahu používateľa do zdrojového kódu. Bude mu stačiť ponastavovať pár vecí v konfiguračnom súbore a môže klasifikovať na nové poškodenia odtlačkov prstov.

```
{
        "model_directory": "assets/saved_models/",
        "model_name": "some_name",
        "predict_file": null,
        "predict_dir": "assets/Database/prediction/data/",
        "output_directory": "assets/prediction_by_cnn/",
        "is_enabled_marking": true,
        "threshold_warts": 0.5761,
        "thresholds_dyshidoris": 0.5761,
        "thresholds_psoriasis": 0.5761,
        "thresholds_pressure": 0.5761,
        "thresholds_moisture": 0.5761
}
```

Algoritmus 4.3: Konfiguračný súbor pre predikciu a klasifikáciu

4.5.3 Výstup detekcie poškodenia odtlačkov prstov

Pre výstup z detekcie sa bude používať upravené implementované riešenie z bakalárskej práce [70]. Toto riešenie implementuje prechod celého obrázka, kde na konci vytvorí výstupný súbor vo formáte JSON. V tomto súbore budú zaznačené oblasti jednotlivých poškodení, ktoré klasifikátor bude detekovať. Súbor JSON obsahuje nasledujúce údaje. Prvým údajom je názov predikovaného súboru. Ďalší údaj sú informácie o obrázku. Tento údaj tvorí pole o troch celých čisiel. Prvé číslo hovorí ovýške obrázku. Druhé číslo hovorí o šírke obrázku. Posledné číslo popisuje koľko má kanálov v obrázku. Posledný údaj je pole s nájdenými chorobami alebo poškodeniami pri snímaní. Každý prvok sa skladá z dvoch bodov, aby bolo možné vykresliť obdĺžnik do obrázka, kde sa nachádza ochorenie alebo poškodenie. Bod tvoria dve desatinné čísla, ktoré zapisujú bod v pixeloch. Ďalej sa u každého prvku nachádza typ ochorenia alebo poškodenia. Posledný údaj v tomto prvku je pravdepodobnosť s akou sieť určila ochorenie alebo poškodenie.

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 podkapitola 5.3 bude rozoberať implementáciu prepisu z knižnice Keras do knižnice PyTorch. Ďalšou podkapitolou 5.4 budú popísané implementácie metód pre vytvorenie databázy. Konkrétne implementácia predspracovania pomocou Gáborovho filtra, implementácia anotácii zdravých odtlačkov prstov a nakoniec implementácia vytvorenia databázy. Podkapitola 5.5 popisuje trénovanie konvolučnej neurónovej siete. Táto podkapitola rozoberie implementáciu trénovania, ktorá bude využívať návrh sadu vylepšení, ktoré sú v podkapitole 4.2. Predposledná podkapitola 5.6 popíše implementáciu klasifikátora. V poslednej podkapitole 5.7 bude popísaná implementácia prípravy testovania.

5.1 Možnosti implementácie konvolučnej siete

Táto podkapitola sa bude zaoberať API Keras, ktorá bola použitá v bakalárskej práci [70]. 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 v mnohých článkoch 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. Nad uloženým modelom 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 v mnohých článkoch označovaný ako knižnica. Podľa oficiálnych stránok [24] 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. [24]

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. [24]

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*** [24]

5.1.3 Rozdiely

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 [54] 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 funguje stabilnejšie ako Keras. Framework PyTorch bol vybratý pre diplomovú prácu. Dôvody pre výber sú rýchlejšie učenie, stabilný framework 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 1.9.0)* knižnica PyTorch, ktorá sa používa na trénovanie a pre správu neurónovej siete.
- *torchvision (verzia 0.10.0)* podporná knižnica k *torch*, na pracovanie s obrázkami a transformácie pred vložením obrázka do siete.
- *opencv-python (verzia 4.5.5.62)* OpenCV knižnica, ktorá sa používa pri pracovaní s obrázkami.

- *numpy (verzia 1.20.2)* knižnica NumPy sa používa na prácu s maticami.
- *pandas (verzia 1.4.0)* Pandas je knižnica, ktorá sa v diplomovej práci využíva na vytvorenie anotácie a načítanie do datasetu pre knižnicu PyTorch.
- *pillow (verzia 9.0.1)* 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 popísané cloudové riešenie MetaCentrum.

5.2.1 MetaCentrum VO

Jedná sa o virtuálnu organizáciu, ktorá združuje akademických pracovníkov, vedcov, študentov, akademických zamestnancov alebo vedecké inštitúcie po celej Českej republike. [53]

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.

```
#!/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 5.1: Dávkový súbor pre učenie v knižnici PyTorch.

Algortimus 5.1 zobrazuje PBS dávku pre trénovanie neurónovej konvolučnej siete z diplomovej práce. Z tohto algoritmu je vidieť, že na Metacentru sa používajú oficiálne kontajnery pre učenie neurónových sietí s použitou knižnicou PyTorch. Kontajnery obsahujú virtuálne rozbehnutý operačný systém Ubuntu s danou verziou knižnice Pytorch. Vytvorené sú od spoločnosti NVIDIA. Z algoritmu je ďalej vidieť, že sa používa ešte jeden skript, v ktorom je spustenie python skriptu s konfiguráciou, a je zobrazený na algoritme 5.2. Dávkové súbory sú vytvorené ešte pre vytvorenie databázy a klasifikáciu pomocou konvolučnej neurónovej siete.

```
#!/bin/bash

cd /storage/brno3-cerit/home/xvican02/relu
python3 learn_cnn_main.py -c config/learn_cnn/learning_cnn.json
```

Algoritmus 5.2: Skript, ktorý spúšťa učenie konvolučnej neurónovej siete.

5.3 Prepis modelu z Keras do PyTorch

Táto podkapitola sa bude zaoberať prepisom modelu z Keras do PyTorch. Tento model je z riešenia, ktoré sa rozširuje a vylepšuje. 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. Nasledujúce odstavce popisujú dôležité zmeny, ktoré museli byť vykonané. Prvou zásadnou zmenou je, že PyTorch využíva triedny prístup k vytvoreniu modelu. Znamená to, že model je napísaný v triede. V Keras je celý model napísany vo funkcii.

2D konvolučná vrstva sa v knižnici Keras definuje tak, ako je to zobrazené v prílohe B. 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. Algoritmus pre PyTorch je zobrazený v prílohe C. V knižnici PyTorch sa konvolučná sieť skladá v metóde *forward*. Ďalším rozdielom je prepis *max-pooling* vrstvy. V Keras je to funkcia s dvoma parametrami a v PyTorch sú dva parametre, ktoré sú dvojice. V knižnici Keras sa na zjednodušenie siete používa *Flatten* vrstva. 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. *Dropout* vrstva sa prepíše veľmi ľahko. Sú to veľmi podobné zápisy až na zloženie pomocou ReLu aktivačnej funkcie.

Najzložitejšie sa prepisuje *Dense* vrstva, ktorá celú sieť zjednodušuje do jedného čísla. Implementácia Dense vrstvy v knižnici Keras je jednoduchá. Zavolá sa *Dense* s dvoma parametrami. 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 predchádzajúcou výstupnou hodnotou. Druhý parameter je výstupná hodnota.

Vo funkcii pre vytvorenie modelu sa v knižnici Keras nakoniec kompiluje model so stratovou funkciou a optimalizátorom. To sa v knižnici PyTorch robí až pri tvorení algoritmu učenia. Implementácia stávajúceho modelu je rozšírená o vylepšenie aktivačnej funkcie. Pre toto rozšírenie sa musel model upraviť tak, aby bolo možné vymeniť funkciu pomocou parametra. Musela byť upravená metóda*forward*, ktorá najprv model implementovala priamo. Metóda má v sebe tri podmienky, ktoré určujú model s danou aktivačnou funkciou.

5.4 Implementácia metód súvisiacich s tvorbou databázy

Podkapitola sa bude zaoberať implementáciou metód súvisiacich s tvorbou databázy. Konkrétne sa jedná o popísanie implementácie predspracovania pomocou Gáborovho filtra, implementácia prípravy anotácií čistých odtlačkov prsta a implementácia vytvorenia databázy.

5.4.1 Implementácia predspracovania

V podkapitole 4.3.4 sa rozoberalo, že v implementácii bude použitý Gáborov filter. Jeho implementácia vychádza z rovnice pre 2D Gáborovu funkciu [27]. Táto funkcia je zobrazená v rovnici 5.1.

$$f(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp(-\frac{x^{'2} + \gamma^2 y^{'2}}{2\sigma^2})exp(i(2\pi\frac{x^{'}}{\lambda} + \psi)) \tag{5.1}$$

Reálna časť je zobrazená na rovnici 5.2

$$f(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp(-\frac{x^{'2} + \gamma^2 y^{'2}}{2\sigma^2})cos(2\pi\frac{x^{'}}{\lambda} + \psi) \tag{5.2}$$

Imaginárna časť je zobrazená na rovnici 5.3.

$$f(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp(-\frac{x^{'2} + \gamma^2 y^{'2}}{2\sigma^2})sin(2\pi\frac{x^{'}}{\lambda} + \psi) \tag{5.3}$$

Pre funkcie je nutné ešte zadefinovať $x^{'}$ a $y^{'}$. Rovnica pre $x^{'}$ je $x^{'} = xcos\theta + ysin\theta$. Rovnica pre $y^{'}$ je $y^{'} = -xsin\theta + ycos\theta$. V rovniciach jednotlivé premenné prezentujú nasledujúce:

- λ označuje vlnovú dĺžku sínusového faktoru.
- θ označuje orientáciu normály k rovnobežným pruhom Gáborovej funkcie.
- ψ označuje fázový posun.
- σ označuje smerodatnú odchýlku.
- γ označuje priestorový pomer strán.
- x je šírka obrázka.
- y je výška obrázka.

Gáborov filter je implementovaný ako trieda. Konštruktor má dva povinné parametre. 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. Nad inštanciou triedy sa zavolá metóda *image_after_gabor*, ktorá má jeden parameter. Tento parameter je obrázok do Gáborovho filtra.

5.4.2 Príprava anotácie zdravých odtlačkov prsta

Príprava anotácií zdravých odtlačkov prstov bolo nutné implementovať z dôvodu predprípravy komplet celej databázy. Anotácia je tiež ako Gáborov filter implementovaná ako trieda. Konštruktor má dva parametre. Prvý parameter je zdrojová databáza a druhý parameter je cieľová databáza. Zdrojová databáza sa odkazuje na priečinok s čistými odtlačkami bez ochorenia. Cieľová databáza je priečinok, kde sa uloží obrázok s anotáciami s čistými odtlačkami.

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*. Tieto súradnice sa uložia do JSON súboru a obrázok sa prekopíruje. Toto anotovanie je zobrazené na algoritme 5.3

```
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 5.3: Anotovanie zdravých odtlačkov prstov

5.4.3 Implementácia vytvorenia databázy

Implementácia vychádza z návrhu spracovania odtlačkov prstov, ktorá je popísaná v podkapitole 4.3.4 a z návrhu konfiguračného súboru pre tvorbu databáz, ktorá je popísaná v podkapitole 4.3.2. V tejto podkapitole bolo zmienené, že databáza sa vytvorí pred trénovaním siete. Táto zmena je vykonaná kvôli optimalizácii pri učení siete. Hlavný skript sa volá *create_database_main.py*, v ktorom je zakomponovaná logika parsovania konfiguračného súboru a zavolania triedy *CreateDatabase*, kde je implementovaná celá funkcionalita.

Volanie hlavného algoritmu na vytvorenie databázy sa volá metódou *invoke_option*. Táto hlavná metóda zodpovedá za rozhodnutie, čo sa vykoná. Jedná sa o funkcionalitu odstránenie priečinkov, vytvorenie anotácie, alebo samotné vytvorenie databázy danej kategórie. Vytvorenie databázy spočíva v niekoľkých krokoch. Oreže sa obrázok pre trénovaciu a validačnú databázu. Následne sa použije na takto orezaný obrázok Gáborov filter, ktorý bol popísaný v podkapitole 5.4.1. Následne sa na obrázok aplikuje rozmazanie a nakoniec sa použije binárne prahovanie s OTSU prahovaním. Štruktúra databázy je implementovaná z návrhu štruktúry databázy, ktorá je popísaná v podkapitole 4.3.3. Pri ukladaní obrázkov sa zapíše do jednotlivých csv súborov pre trénovanie, validáciu a predikciu relatívna cesta k obrázku a jeho trieda v číselnej podobe.

5.5 Implementácia trénovania

Nasledujúca podkapitola bude rozoberať implementáciu trénovania. V tejto podkapitole budú popísané, ako sú v implementácii zakomponované jednotlivé vylepšenia z podkapitoly 4.2. Trénovanie je implementované v *learn_cnn.py*. V tomto súbore je trieda, ktorá sa stará o celé trénovanie. Konštruktor má jeden povinný parameter a to *logger*, ktorým sa nastavuje vlastné logovanie udalostí v skripte. Trieda má tri verejné metódy. Druhá verejná metóda sa volá *load_configuration()*. Táto metóda má načítať konfiguračný súbor do premennej, odkiaľ sa budú brať informácie pre trénovanie. Posledná verejná metóda je *learn()*. Métoda, ktorá volá privátne metódy na učenie siete. 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.

5.5.1 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. Toto načítanie je implementované v triede *FingerPrintDataset*. Táto trieda má ako predka *Dataset*. Trieda musí zdediť túto triedu z dôvodu, že PyTorch pracuje s metódami s tejto triedy. Konštruktor má dva povinné parametre a jeden voliteľný. Prvým parametrom je priečinok, kde sa nachádza csv súbor. Druhý parameter je názov csv súboru. Posledný parameter definuje transformácie nad obrázkom.

Inicializácia trénovacej a validačnej databázy je vytvorená funkciou *___create_loader*. Funkcia vygeneruje inštanciu pre načítanie trénovacej a validačnej databázy. Funkcia má dva parametre. Prvým parametrom je cesta a druhý parameter je názov súboru v csv formáte. Tento súbor obsahuje relatívne cesty k vyrezaným obrázkom a ich triedu v číselnej podobe. Táto funkcia ešte vytvára trénovaciu alebo validačnú databázu pre zmenu modelu na predtrénovaný model.

Typy transformácií obrázku je definovaná pomocou lambda funkcie, ktorá je zobrazená na algoritme 5.4. Lambda funkcia má jediný parameter, a to veľkosť obrázka, ktorý je rozdielny pre model z bakalárskej práce [70], 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 5.4: Transformácia pomocou lambda funkcie

5.5.2 Trénovanie

Pred samotným trénovaním sa musia nastaviť parametre stratovej funkcii, a aký optimalizátor bude použitý. Aby sa mohla sieť učiť na grafickej karte, tak je potrebné zistiť, či sa nachádza na stroji grafická karta. Ak áno, tak sa sieť prepne na grafickú kartu.

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 stav modelu, ktorý je rezprezentovaný slovníkom v *Pythone* s definovanými váhami, epocha a úspešnosť. Algoritmus je implementovaný tak, že do súboru uloží len najlepší model. Učiaci proces je zobrazený na dvoch algoritmoch. Prvý algoritmus 5.5 zobrazuje prvý cyklus *for*. V ňom je vidieť, že validačný proces je vytiahnutý do samostatnej metódy. Podobne aj uloženie najlepšieho modelu je vytiahnutý do samostatnej metódy.

```
def ___learning_proces(self, train_dataloader, validation_dataloader, optimizer, loss_function):
        min_valid_loss = np.inf
        ...
        for epoch in range(self.___config.epochs()):
                number_images = 0
                train_loss = 0.0
                self.___cnn_model.train()
                for batch_idx, (data, labels) in enumerate(train_dataloader, 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.___config.batch_size()

                        if batch_idx % self.___config.batch_size() == 0 and batch_idx != 0:
                                min_valid_loss = self.___validation_proces(
                                        batch_idx, data, epoch, loss_function, min_valid_loss,
                                        number_images, train_loss, validation_dataloader
                                )

                        if self.___config.training_steps() == batch_idx:
                                break

        self.___save_trained_model_to_file()
```

Algoritmus 5.5: Trénovací proces.

Druhý algoritmus je časťou učiaceho algoritmu, ktorý spúšťa validáciu nad databázou a naučeným modelom. Algoritmus má za úlohu validovať natrénovaný model. V prvom kroku sa musí prepnúť model z trénovacieho módu do módu, kde model vykonáva predikciu. Celý algoritmus beži v cykle *for*. V cykle je podmienka, ktorá môže ukončiť validačný proces skôr ako je celá databáza. V cykle sa testuje sieť v určitých dávkach vzorkov, z ktorých sa vypočíta, koľko obrázkov je označených správne a koľko je zle. Následne sa vypočíta validačná odchýlka a presnosť daného modelu. Po skončení cyklu sa model uloží do premennej a algoritmus vráti novú vypočítanú validačnú chybu pokiaľ sa zvýšila presnosť. V opačnom prípade posiela starú validačnú chybu. Po trénovacom procese sa uloží model do súboru. Zároveň s modelom sa uložia informácie o modeli v JSON súbore. Tieto informácie sú použité pre načítanie modelu zo súboru. Na nasledujúcom algoritme 5.6 je zobrazený JSON súbor.

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

Algoritmus 5.6: Informácie o uloženom modeli.

5.6 Implementácia klasifikátora

Implementácia klasifikátora je inšpirovaná z bakalárskej práce [70]. Inšpiruje sa hlavne algoritmom kĺzavého okna, ktorý sa používa na detekciu a klasifikáciu. Implementácia klasifikátora je podobne naimplementovaná ako implementácia učiaceho procesu. Prvým krokom je vytvorená obálka v skripte, ktorý sa nazýva *predict_cnn_main.py*. Tento skript spúšťa parsovanie argumentu na získanie cesty ku konfiguračnému súboru. Tento konfiguračný súbor je popísaný v podkapitole 4.5.2. V hlavnom skripte sa volá inštancia triedy *PredictCNN*. Konštruktor triedy má jeden parameter. Tento parameter je vlastná implementácia ladiacich výpisov. Následne trieda *PredictCNN* obsahuje tri verejné metódy. Prvá metóda sa volá *update_configuration_path()*, ktorá má jeden parameter. Parametrom je cesta ku konfiguračnému súboru. Ďalšou verejnou metódou je *load_configuration()*. Táto metóda má za úlohu načítať konfiguračný súbor do vnútornej reprezentácie pre skript. Posledná verejná metóda je *predict()*. Táto metóda spúšťa detekciu a klasifikáciu.

Detekcia a klasifikácia je možná ako pre samotný súbor, tak aj pre celý priečinok. Rozhodovanie o tom, či sa bude detekovať a klasifikovať priečinok alebo samotný súbor sa rozhoduje vo verejnej metóde *predict()*. Rozhodovanie je vytvorené pomocou *if-elif-else* konštruktu, ktorým sa v *Pythone* imlementuje konštrukt *case* z jazyka *C*. Prvá podmienka hovorí o tom, či používateľ nastavil priečinok pre preidkciu. Ak áno, tak sa predikuje a klasifikuje celý priečinok. Ak nie je priečinok nastavený, kontroluje sa nastavenie súboru. Ak je súbor nastavený, tak sa predikuje súbor. V opačnom prípade sa vypíše do logu varovanie, že sa nič nepredikovalo.

Nasledujúci algoritmus 5.7 zobrazuje samotný algoritmus detekcie a klasifikácie pomocou kĺzavého okna. Z algoritmu je vidieť, že algoritmus končí, keď dosiahne hodnotu posunu kĺzavého okna. Tento parameter sa nastavuje v konfiguračnom súbore. Podobne ako učiaci proces, tak aj detekcia a klasifikácia funguje generickým spôsobom. Nasledujúce podkapitoly popisujú jednotlivé generické algoritmy.

```
while sliding_window < self.___config.end_size_sliding_window():
        predict_height = 0
        predict_width = 0
        while True:
                ...

                if predict_width + sliding_window >= width:
                        predict_width = 0
                        predict_height += step
                else:
                        predict_width += step

                if predict_height + sliding_window >= height:
                        break

                if predict_width > width:
                        predict_width = width - 1

                if thresh is None:
                        continue

                probabilities = self.___predict_with_image_model(
                        sliding_window, image, self.___cnn_model, predict_width, predict_height)
                ...
                if self.___is_greater_than_first(probabilities):
                        index_list = self.___get_damage_from_probabilities_and_threshold(
                                self.___load_fingerprint_damage.get_list_fingerprint_damages(),
                                    probabilities)

                        if index_list == self.___NOT_VALID_INDEX:
                                continue
                        else:
                          damage = self.___load_fingerprint_−damage.get_list_fingerprint_damages()[index_list]
                          hits_frames.append(self.___get_frame(
                                  damage, sliding_window, probabilities[index_list], predict_width, predict_height) )

                        step = self.___update_min_size_step(int(step / 2), self.___MIN_SIZE_STEP)
                        hit += 1
                  else:
                        step = self.___update_max_size_step(step * 2, self.___config.max_step())

        sliding_window += self.___config.sliding_window_size()
```

Algoritmus 5.7: Detekcia a klasifikácia pomocou kĺzavého okna.

5.6.1 Generické algoritmy pre predikciu a klasifikáciu

Použitie nižšie generických algoritmov je zobrazené v algoritme 5.7. **Generický algoritmus detekcie z modelu** popisuje získanie pravdepodobností, ktorá trieda sa nachádza na výreze, ktorý ide do modelu. Jedná sa o metódu *___predict_with_image_model*. Algoritmus má päť parametrov. Prvým parametrom je aktuálne posunutie kĺzavého okna. Ďalši parameter je aktuálny obrázok. Následne je natrénovaný model a posledné dva parametre sú šírka a výška pre predikciu. Metóda vracia pole pravdepodobností, ktorá trieda a aké ochorene sa na výreze nachádza.

Ďalším generickým algoritmom je **algoritmus pre kontrolu pravdepodobností**. Jedná sa o metódu *___is_greater_than_first*. Algoritmus učenia a aj klasifikácie je naimplementovaný tak, že ako prvú triedu očakáva čistý odtlačok prsta. Z tohto dôvodu sa kontroluje, či pravdepodobnosť v poli je väčšia než prvý prvok, ktorý je zdravý odtlačok prsta. Tento algoritmus vracia logickú hodnotu pravda alebo nepravda. Pravdu vráti, ak index je rozdielny od nula. V opačnom prípade vráti nepravdu.

Posledný algoritmus je **generický algoritmus pre určenie poškodenia odtlačku prsta.** Jedná sa o metódu *___get_damage_from_probabilities_and_threshold*. Algoritmus používa list s definovanými poškodeniami, ktorý je definovaný v podkapitole 4.3.1. Algoritmus funguje tak, že prechádza pravdepodobnosti jednu po druhej. Vyberie pravdepodobnosť a skontroluje, či je najväčšia pravdepodobnosť. Ak je, tak sa skontroluje ešte prah pre danú chorobu. Ak je hodnota väčšia ako prah, tak sa vráti index s daným poškodením odtlačku prsta. V opačnom prípade sa vráti -1.

5.7 Implementácia testovania

Implementácia testovania je rovnako ako implementácia trénovania a implementácia klasifikátora úplne nová a skript je vytvorený pre testovanie rôzneho počtu tried pri trénovaní a klasifikácií. Implementácia testovania preberá myšlienky testov ako je test plochy a test správneho určenia z riešenia, ktoré je rozširované. Pred implementáciou testovania budú popísané tieto testy.

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 neoznačilo. Druhá oblasť určuje koľko plochy je označenej mimo anotácie. 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. [70]

Test správneho určenia je druh testu, ktorý skúma tiež ako test plochy dve oblasti. Prvá oblasť je aká je úspešná sieť. Druhá oblasť testuje, koľko 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 vypíše percentuálne ako sa darilo. [70]

Po definovaní týchto testov sa môže popísať implementácia testovania. Rovnako ako trénovanie a predikcia bude testovanie implementované v hlavnom skripte, ktorý sa volá *test_cnn_main.py*. V tomto skripte bude rovnako implementované parsovanie pre získanie konfiguračného súboru pre testovanie. Samotné testovanie je implementované v triede *TestCNN* v súbore *test_cnn.py*. Konštruktor triedy je jeden parameter. Tento parameter slúži na vypisovanie vlastných ladiacich výpisov do konzoly alebo do súboru. Trieda má tri verejné metódy. Prvá metóda *update_configuration_path()* má jeden parameter. V parametri sa ukrýva cesta ku konfiguračnému súboru, kde sú zadefinované všetky potrebné veci pre testovanie. Nasledujúcou verejnou metódou je *load_configuration()*. Táto metóda načíta konfiguráciu do vnútornej reprezentácie skriptu. Posledná verejná metóda je *test()*. V tejto metóde sa volá samotné testovanie po predikcii. Výstup z testu správneho určenia sú JSON súbory pre jednotlivé odtlačky, v ktorých je počet správnej triedy, nesprávnej triedy, falošnej pozitivity a celkový počet určení. Po testovaní je vytvorený jeden súhrnný JSON súbor, kde sú zobrazené výsledky pre celý priečinok. Podobne je to pre test správneho určenia. Zmenou sú údaje, ktoré sa v JSON súbore ukladajú. Týmito údajmi sú správne označená plocha, zle označená plocha, plocha bez označenia a posledný údaj podiel označenej plochy mimo anotácií. Po tomto sa zavolá ešte skript *hit_area_check_percentage.py*, ktorý tieto údaje prevedie do percent a uloží to do posledného JSON súboru.

Kapitola 6

Testovanie a výsledky

Kapitola sa zaoberá testovaním a zhodnotením výsledkov. Testovanie prebieha nad konvolučnými neurónovými sieťami, ktoré používajú tri rôzne aktivačné funkcie. Konkrétne sú to *ReLu*, *leaky ReLu* a*tanh*. Tieto tri konvolučné siete boli trénované nad databázou A a databázou A6, kde vzniklo šesť mutácii natrénovaných konvolučných sieti. Testovanie je rozdelené do podkapitol. Prvá podkapitola bude porovnávať veľkosť natrénovaného modelu z diplomovej práce a z bakalárskej práce [70]. Následne sa porovná nad rovnakou databázou rýchlosť detekcie. Po týchto testoch a porovnaniach sa prejde na samotné testovanie plochy a správneho určenia. 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 [71] a reálne odtlačky prstov. Nad touto Databázou bol použitý Gáborov filter s počtom krokov nula a s posunom v každom kroku o 30 stupňov. Zloženie databázy je 15000 zdravých vzoriek, 15000 s dyshidrózou, 15000 s bradavicami, 150 s psoriáziou. Tento počet je nízky, kvôli použitiu iba reálnych odtlačkov. 15000 vzoriek s ekzémom, 15000 vzoriek s rôznym tlakom a 15000 s vlhkosťou.

- *databáza A6* obsahuje syntetické odtlačky z SFinGe, čiastočne NIST_DB4 [71] a reálne odtlačky prstov. Nad touto Databázou bol použitý Gáborov filter s počtom krokov šesť a s posunom v každom kroku o 30 stupňov. Zloženie databázy je 15000 zdravých vzoriek, 15000 s dyshidrózou, 15000 s bradavicami, 150 s psoriáziou. Tento počet je nízky, kvôli použitiu iba reálnych odtlačkov. 15000 vzoriek s ekzémom, 15000 vzoriek s rôznym tlakom a 15000 s vlhkosťou.

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

- *databáza AB* - testovacia databáza s 37 zdravými odtlačkami.

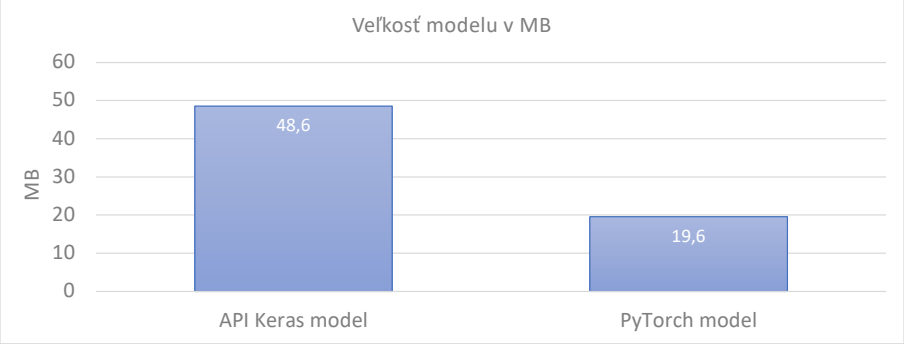
- *databáza B* - testovacia databáza s 10 zdravými odtlačkami, 10 odtlačkov s chorobou dyshidróza a 10 odtlačkov s chorobou bradavice. Tieto odtlačky sú zmesou syntatických odtlačkov, NIST_DB4 a reálne odtlačky prstov. Databáza je podmnožinou databázy C, kvôli porovnaniu novej siete so sieťou z bakalárskej práce.

- *databáza C* - testovacia databáza s 10 zdravými odtlačkami, 10 odtlačkov s chorobou dyshidróza a 10 odtlačkov s chorobou bradavice. Ďalej sú to 3 odtlačky s chorobou

psoriáza, 10 odtlačkov s ekzémom, 10 odtlačkov s poškodením pri snímaní tlakom. Posledných 10 odtlačkov prstov majú poškodenie pri snímaní spôsobené vlhkosťou. Tieto odtlačky sú zmesou syntatických odtlačkov, NIST_DB4 a reálne odtlačky prstov.

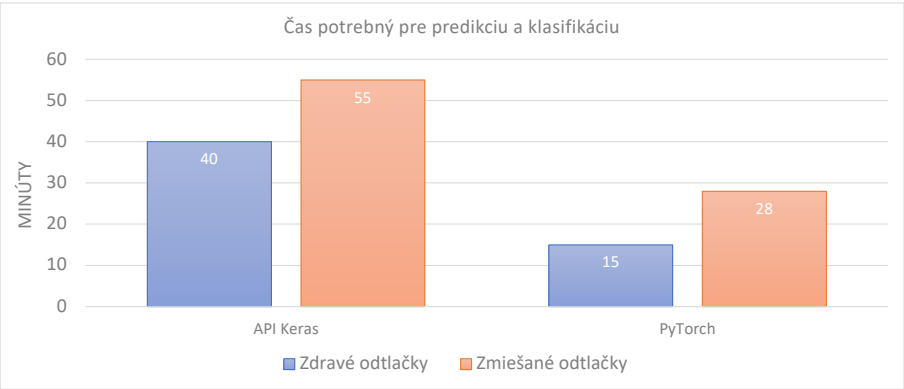
6.1 Testovanie zmeny API Keras na PyTorch

Jedným z prvých testov, ktoré sa dajú spraviť pri zmenení API Keras na Pytorch je test veľkosti modelu. Test sa zameriava koľko pamäte je potrebné na uloženie modelu, ktorý je natrénovaný pomocou knižnice Keras a koľko pamäte zaberá model naučený pomocou PyTorch. Nasledujúci graf 6.1 ukazuje koľko pamäte v megabajtoch je potrebné pre modely. Na vodorovnej časti grafu máme modely a na vertikálnej osi máme veľkosť v MB. Z grafu je vidieť, že model natrénovaný pomocou API Keras má 48,6 MB. Model trénovaný pomocou PyTorch má 19,6 MB. Model natrénovaný pomocou PyTorch je po zaokrúhlení na dve desatinné miesta o 59,67 % menší než model trénovaný pomocou knižnice Keras.



Obrázok 6.1: Veľkosť modelov v MB.

Ďalší všeobecný test, ktorý sa dá spraviť je test rýchlosti predikcie a klasifikácie. Test testuje rýchlosť predikcie a klasifikácie na obidvoch databázach. Výsledky sú zobrazené na grafe 6.2. Zvislá časť na grafe označuje počet minút strávených na predikciu a klasifikáciu. Horizontálna časť grafu je uložená do dvoch klastrov. Prvý kluster zobrazuje počet strávených minút pri predikci a klasifikácii pomocou modelu API keras. V tomto klastri je najprv zobrazený počet minút pre predikciu zdravých odtlačkov a potom zmiešaných. Druhý klaster zobrazuje to isté, ale pre model trénovaný pomocou PyTorch. Z grafu je vidieť, že čas



Obrázok 6.2: Čas strávený pri predkcii a klasifikácii.

strávený čistými odtlačkami je pri modelu s API keras 40 minút a pri Pytorch je to 15 minút. Pri predikcii zdravých odtlačkov prstov je vidieť, že sa dokázalo pomocou modelu PyTorch znížiť čas predikce a klasifikácie o 62,5 %. Pri zmiešaných odtlačkoch API Keras strávil 55 minút predikciou a klasifikáciou a PyTorch model 28 minút. Pri zmiešaných odtlačkov je časová úspora predikcie a klasifikácie znížená po zaokrúhlení na dve desatinné miesta o 49,1 %.

6.2 Testovanie zmeny modelu

Pre účely 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. 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.

6.3 Testovanie na určenie poškodenia

Kapitola sa zameriava na testovanie určenia typu choroby alebo poškodenia odtlačkov prstov. Pri tomto teste sa vychádza z implementácie testovania, ktoré je popísané v kapitole 5.7. Kapitola rozoberie v nasledujúcich podkapitolách testovanie podľa testovacej databázy a trénovacej databázy. Prvá podkapitola 6.3.1 rozoberie testovanie nad databázou AB a trénovacou databázou A. Druhá podkapitola 6.3.2 sa bude zameriavať na testovaciu databázu AB a nad trénovacou databázou A6. Nasledujúca podkapitola 6.3.3 zhodnotí testovanie nad testovacou databázou C a trénovacou databázou A. Štvrtá podkapitola 6.3.4, popíše testovanie testovacej databázy C a trénovacej databázy A6. Predposledná podkapitola 6.3.5 rozoberie testovanie nad databázou B a trénovacou databázou A. Posledná podkapitola 6.3.6 popíše testovanie nad databázou B a trénovacou databázou A6.

6.3.1 Testovacia databáza AB a trénovacia databáza A

Pri testovaní tejto sady databáz sa zistilo, že falošná pozitivita 100 % vyšla len v dvoch prípadoch poškodenia zo šiestich nad konvolučnými sieťami *leaky ReLu*, *ReLu* a *tanh*. Týmito poškodeniami sú choroby dyshidróza a ekzém. Pri ostatných typoch chorôb a poškodení je falošná pozitivita nulová. Na nasledujúcom obrázku 6.3 budú zobrazené odtlačky prstov a konvolučné neurónové siete, ktoré označili dyshidrózu a ekzém na zdravých odtlačkoch prstoch. Z obrázka 6.3 je vidieť červenou farbou dyshidrózu. Cyan farbou je vidieť ochorenie ekzém, ktoré siete označili nad čistými odtlačkami prstov.

(a) Konvolučná sieť s aktivačnou funkciou leaky ReLu.

(b) Konvolučná sieť s aktivačnou funkciou ReLu.

(c) Konvolučná sieť s aktivačnou funkciou tanh.

Obrázok 6.3: Príklad falošnej pozitivity nad zdravými odtlačkami prstov s trénovacou databázou A.

6.3.2 Testovacia databáza AB a trénovacia databáza A6

Testovanie nad databázou AB a trénovacou databázou A6 nie je tak úspešné ako pri prvom teste. Konvolučné neurónové siete s aktivačnou funkciou *leaky ReLu* a *ReLu* majú falošnú pozitivitu 100 % u piatich zo šiestich skúmaných tried poškodenia odtlačku prsta. Jedná sa o dyshidrózu, bradavice, ekzém, tlak a vlhkosť. U neurónovej sieti s aktivačnou funkciou *tanh* sú to štyri triedy zo šiestich skúmaných. Konkrétne sa jedná o dyshidrózu, bradavide, ekzém a tlak. Na nasledujúcom obrázku 6.4 budú zobrazené odtlačky prstov, na ktorých sieť označila falošne nejakú popísanú chorobu. Pre pochopenie tohto obrázka je nutné poznať, ktoré farby patria ktorej chorobe. Bradavice majú modrú farbu, ekzém má cyan farbu, dyshidróza má červenú, tlak má žltú farbu a vlhkosť má purpurovú farbu.

(a) Konvolučná sieť s aktivačnou funkciou leaky ReLu.

(b) Konvolučná sieť s aktivačnou funkciou ReLu.

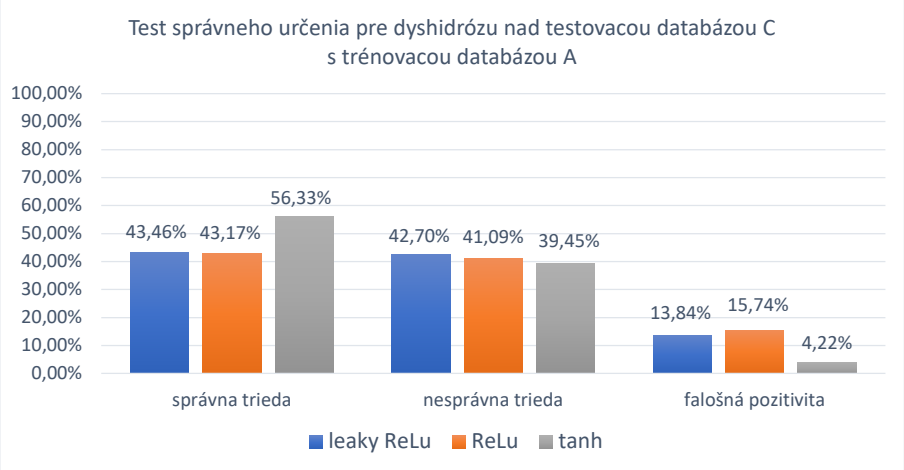
(c) Konvolučná sieť s aktivačnou funkciou tanh.

Obrázok 6.4: Príklad falošnej pozitivity nad zdravými odtlačkami prstov s trénovacou databázou A6.

6.3.3 Testovacia databáza C a trénovacia databáza A

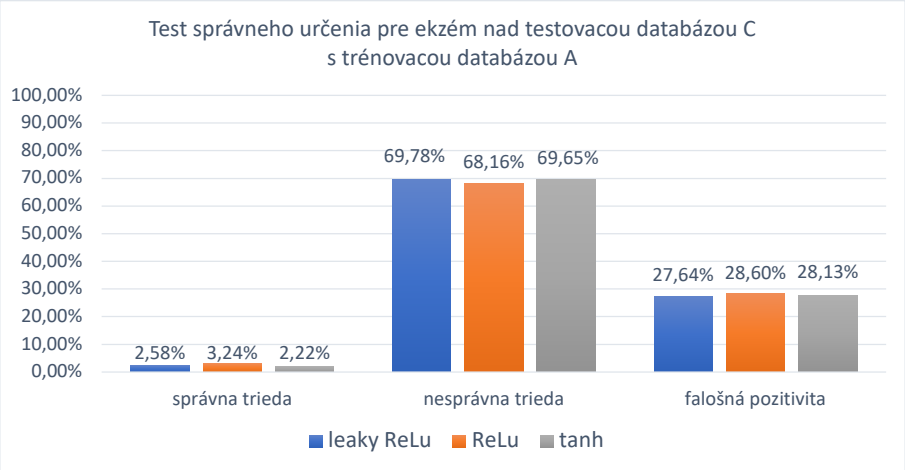
Kapitola sa bude zaoberať výsledkami nad testovacou databázou C trénovacou databázou A. Výsledky budú popísané v grafoch pre jednotlivé ochorenia, aby bolo na grafe vidieť ako sa darí jednotlivým konvolučným neurónovým sieťam.

Prvým ochorením je **dyshidróza**. Na nasledujúcom obrázku 6.5 je zobrazený graf, ktorý popisuje ako sa darí určiť správna trieda, nesprávna trieda a falošná pozitivita. Z tohto grafu je vidieť, že najlepšie sa darilo sieti s aktivačnou funkciou *tanh*. Táto sieť určila správne 56,33 %. Následuje za ňou sieť s *leaky ReLu*. Označila správnu triedu v 43,46 % prípadoch. Najhoršie sa darilo sieti s *ReLu*. Táto sieť označila správnu triedu v 43,17 % prípadoch. Pri označení nesprávnej triedy sa najlepšie darilo sieti s *tanh* funkciou. Táto sieť označila 39,45 % nesprávnej triedy. Sieť s aktivačnou funkciou *leaky ReLu* označila 42,7 % nesprávnej triedy. Sieť s *ReLu* funkciou označila nesprávnu triedu v 41,09 % prípadoch. Najmenej falošnej pozitivity má sieť s *tanh* funkciou. Označila len 4,22 %. Následuje sieť s *leaky ReLu*, ktorá označila falošnú pozitivitu u 13,84 % prípadov. Posledná bola sieť s *ReLu* funkciou. Táto sieť falošne pozitívne označila 15,74 % prípadov.



Obrázok 6.5: Test správneho určenia pre dyshidrózu nad testovacou databázou C a trénovacou A.

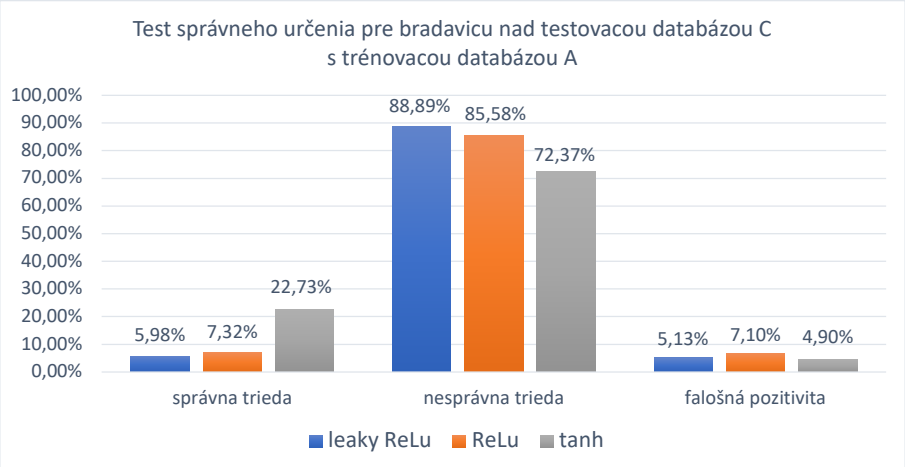
Druhou chorobou je **ekzém**. Výsledky sú zobrazené formou grafu na obrázku 6.6. Z tohto grafu vyplýva, že správnu triedu najlepšie určila sieť s *ReLu* funkciou. Označila správnu triedu v 3,24 % prípadoch. Za ňou je sieť s *leaky ReLu* funkciou. Táto sieť označila správnu triedu v 2,58 %. Najhoršie sa darilo sieti s funkciou *tanh*. Táto sieť označila správne triedu v 2,22 % prípadoch. Pri nesprávnej triede sa najlepšie darilo sieti s funkciou *ReLu*. Sieť nesprávnu triedu určila v 68,16 %. Sieť s funkciou *tanh* označila nesprávnu triedu 69,65 %. Najhoršia bola sieť s *leaky ReLu*. Táto sieť označila 69,78 % nesprávnej triedy. Najmenej označenú falošnú pozitivitu má sieť s *leaky ReLu*. Falošne pozitívnu triedu označila v 27,64 % prípadoch. Druhá je sieť s *tanh*. Táto sieť falošne pozitívnu triedu označila v 28,13 %. Posledná je sieť s *ReLu* funkciou. Falošnú pozitivitu klasifikovala v 28,6 %.



Obrázok 6.6: Test správneho určenia pre ekzém nad testovacou databázou C a trénovacou A.

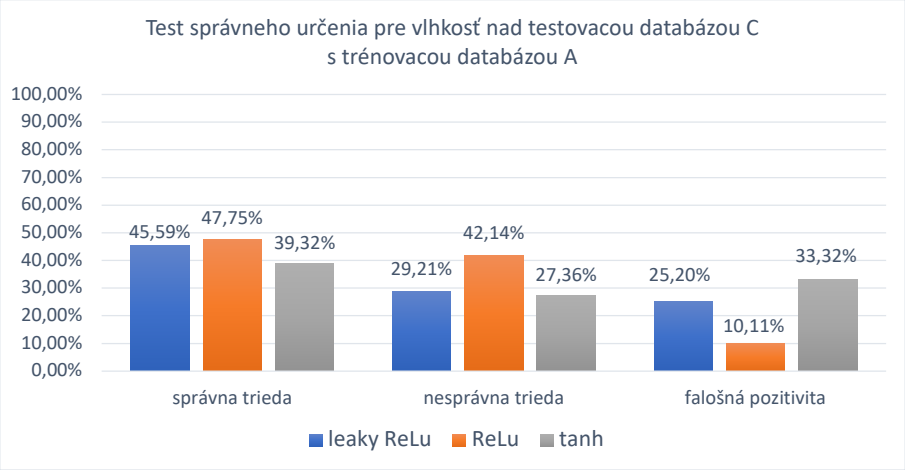
Ďalšou sledovanou chorobou je **psoriáza**. Táto choroba má vo všetkých prípadoch 0 %. Znamená to, že pri tak nízkych počtoch odtlačkov prstov, sieť nebola schopná vôbec rozpoznať správnu triedu. Dobré je, že keď odtlačok prsta nebude obsahovať psoriázu tak ju vôbec neoznačí ako falošne pozitívnu, alebo nesprávnu triedu.

Bradavica je posledné sledované ochorenie. Aj u tohto ochorenia bude zobrazený graf na obrázku 6.7. Pri určení správnej triedy sa najlepšie darilo sieti s *tanh*. Táto sieť označila správnu triedu v 22,73 % prípadoch. Nasleduje sieť s *ReLu* . Táto sieť označila správnu triedu v 7,32 %. Posledná je sieť s funkciou *leaky ReLu*. Označenie správnej triedy tvorí 5,98 %. Pri nesprávnej triede sa najlepšie darilo sieti s *tanh*. Táto sieť nesprávnu triedu určila v 72,37 %. Za ňou nasleduje sieť s *ReLu*. Táto sieť nesprávnu triedu klasifikovala v 85,58 %. Posledná je sieť s *leaky ReLu*. Klasifikovala nesprávnu triedu v 88,89 %. U sieti s *tanh* funkciou bola falošná pozitivita 4,9 %. Falošná pozitivita u sieti s *leaky ReLu* je 5,13 %. A sieť s *ReLu* klasifikovala falošnú pozitivitu u 7,1 % prípadov.



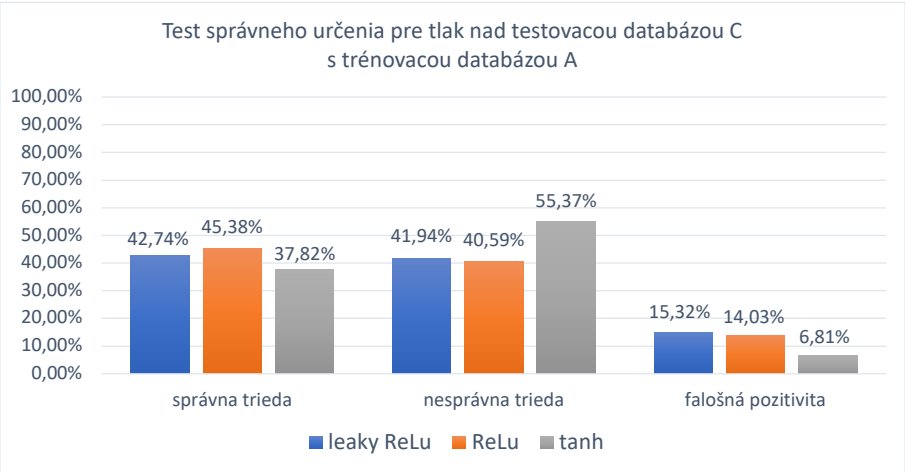
Obrázok 6.7: Test správneho určenia pre bradavicu nad testovacou databázou C a trénovacou A.

Prvým poškodením pri snímaní, ktorým sa test zaoberá je **vlhkosť**. Rovnako ako u chorôb, tak aj pri poškodeniach budú výsledky zobrazené na obrázku s grafom. Tento graf je zobrazený na obrázku 6.8. Z grafu vyplýva, že sa najlepšie darilo sieti s *ReLu* určiť správnu triedu. U tejto sieti to bolo 47,75 %. Následuje za ňou sieť s *leaky ReLu*. Táto sieť správnu triedu klasifikovala v 45,59 % prípadoch. Najhoršie sa darilo sieti s *tanh* funkciou. Táto sieť klasifikovala 39,32 % správnej triedy. Nesprávnu triedu klasifikovala sieť s *leaky ReLu* v 29,21 % prípadoch. 42,14 % nesprávnej triedy klasifikovala sieť s *ReLu*. Sieť s *tanh* označila nesprávnu triedu v 27,36 % prípadoch. U falošnej pozitivity sa najlepšie darilo sieti s *ReLu*. Táto sieť označila iba 10,11 % falošne pozitívnu triedu. U sieti s *leaky ReLu* to je 25,2 %. Sieť s *tanh* označila falošne pozitívne 33,32 %.



Obrázok 6.8: Test správneho určenia pre vlhkosť nad testovacou databázou C a trénovacou databázou A.

Posledným skúmaným poškodením pri snímaní je **tlak**. Výsledky s testovania sú zobrazené na obrázku 6.9 vo forme grafu. Z tohto obrázka vyplýva, že najlepšie správnu triedu určila sieť s *ReLu*. U tejto sieti to je 45,38 %. Sieť s *leaky ReLu* klasifikovala správnu triedu v 42,74 % prípadoch. Klasifikovanie správnej triedy u sieti s *tanh* je 37,82 %. Najmenej



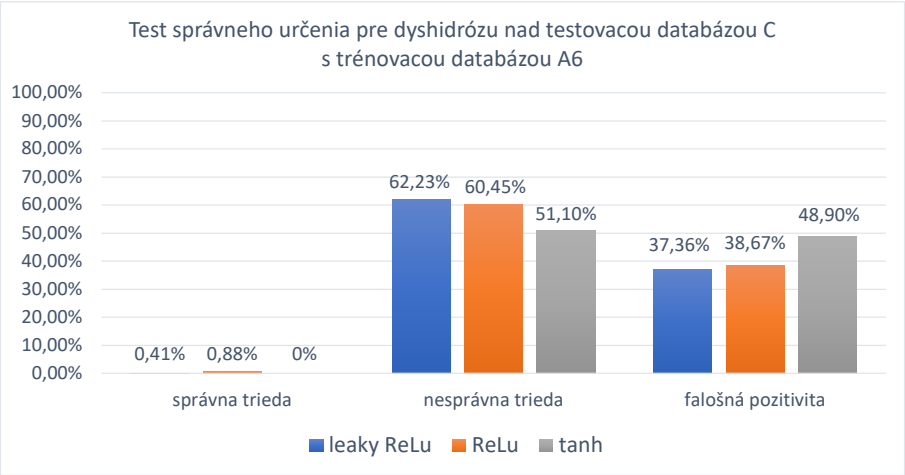
Obrázok 6.9: Test správneho určenia pre tlak nad testovacou databázou C a trénovacou A.

označenej nesprávnej triedy má sieť s *ReLu*. U tejto sieti to činí 40,59 %. Druhá je sieť s *leaky ReLu*. U tejto sieti to je 41,94 %. Posledná je sieť s *tanh*. Nesprávna trieda je klasifikovaná v 55,37 % prípadoch. U falošnej pozitivite sa najlepšie darilo sieti s *tanh*. U tejto sieti to činí 6,81 %. Za ňou je sieť s *ReLu*. Táto sieť klasifikovala falošnú pozitivitu u 14,03 % prípadov. Najhoršie je na tom sieť s *leaky ReLu*. U tejto sieti to činí 15,32 %.

6.3.4 Testovacia databáza C a trénovacia databáza A6

Testovanie bude popísané pre každé ochorenia poškodenie pri snímaní zvlášť, aby bolo možné popísať ako sa darilo alebo nedarilo jednotlivým sieťam. Testovanie prebiehalo nad testovacou databázou C a trénovacou databázou A6.

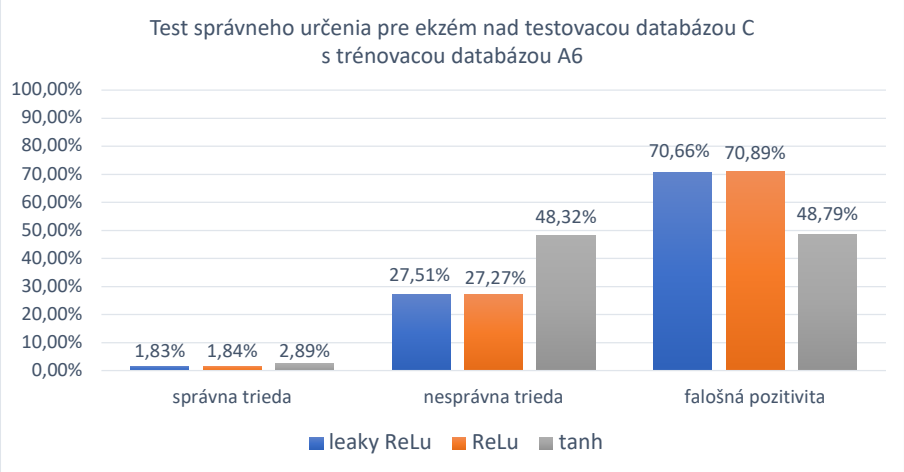
Prvé ochorenie, ktoré sa skúma v teste je **dyshidróza**. Výsledky z testovania sú zobrazené na grafe v obrázku 6.10. Z grafu je vidieť, že sieť s aktivačnou funkciou *leaky ReLu* určila správnu triedu v 0,41 % prípadoch. Správna trieda u sieti s *ReLu* je 0,88 %. U sieti s *tanh* sa nepodarilo vôbec určiť správne triedu. Najmenej nesprávnej triedy klasifikovala sieť s *tanh*, konkrétne 51,1 %. Za ňou nasleduje sieť s aktivačnou funkciou *ReLu*, ktorá klasifikovala 60,45 % nesprávnej triedy. Najhoršie sa viedlo sieti s *leaky ReLu*. U tejto sieti nesprávna trieda bola v 62,23 % prípadoch. Posledný údaj, ktorý je vidieť z grafu, tak je falošná pozitivita. Tá je najmenšia u sieti s *leaky ReLu*. Konkrétne to je 37,36 %. Druhá je sieť s *ReLu*. Táto sieť označila falošnou pozitivitou u 38,67 % prípadov. Najhoršie dopadla sieť s *tanh*. Klasifikovala 48,9 % ako falošne pozitívne.



Obrázok 6.10: Test správneho určenia pre dyshidrózu nad testovacou databázou C a trénovacou A6.

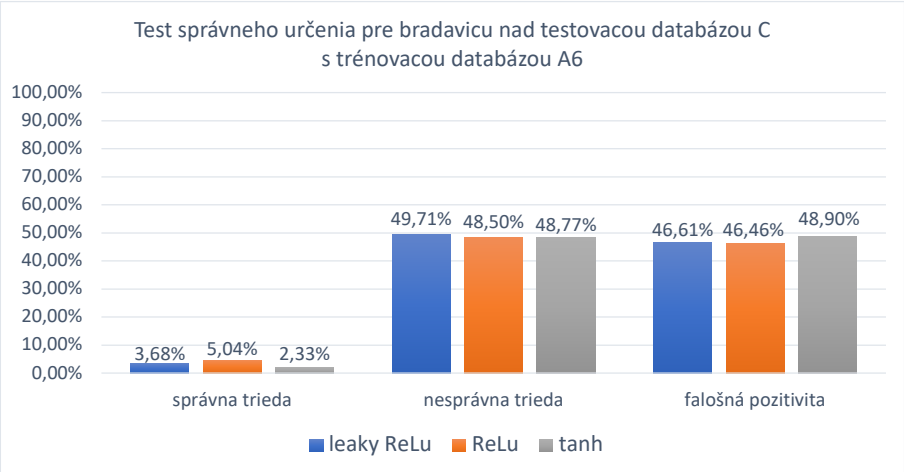
Druhé sledované ochorenie je **ekzém**. Výsledky sú zobrazené formou grafu na obrázku 6.11 Z tohto grafu vyplýva, že správnu triedu najlepšie určila sieť s *tanh* funkciou. Sieť klasifikovala správnu triedu v 2,89 % prípadoch. Za ňou nasleduje sieť s *ReLu* funkciou. Sieť klasifikovala správne 1,84 %. Najhoršie sa darilo sieti s funkciou *leaky ReLu*. Táto sieť označila správne triedu v 1,83 % prípadoch. Pri nesprávnej triede sa najlepšie darí sieti s funkciou *ReLu*. Sieť nesprávnu triedu určila v 27,27 %. Sieť s funkciou *leaky ReLu* označila nesprávnu triedu 27,51 %. Najhoršia bola sieť s *tanh*. Táto sieť označila 48,32 % nesprávnej triedy. Najmenej označenu falošnú pozitivitu má sieť s *tanh*. Falošne pozitívnu triedu označila v 48,79 % prípadoch. Druhá je sieť s *leaky ReLu*. Táto sieť falošne

pozitívnu triedu označila v 70,66 %. Posledná je sieť s *ReLu* funkciou. Falošnú pozitivitu klasifikovala v 70,89 %.



Obrázok 6.11: Test správneho určenia pre ekzém nad testovacou databázou C a trénovacou A6.

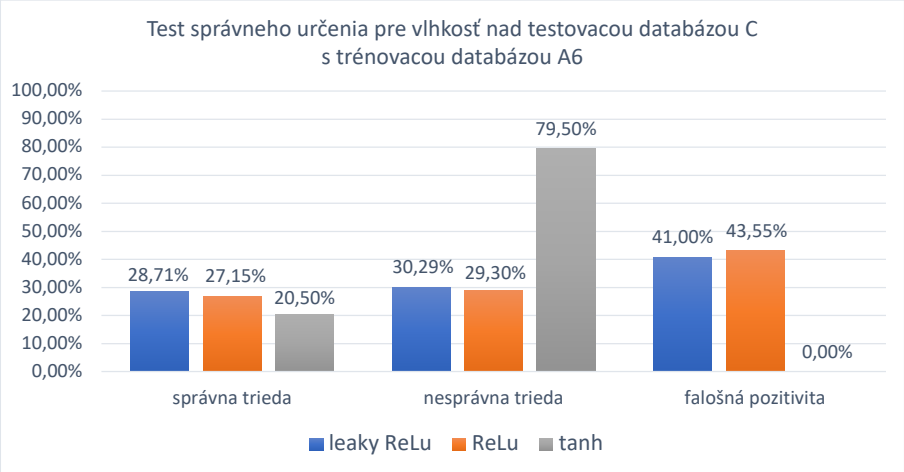
Ďalšou sledovanou chorobou je **bradavica**. Výsledky testovania sú zobrazené rovnako ako u predošlých ochorení na grafe v obrázku 6.12. Pri určení správnej triedy sa najlepšie darilo sieti s *ReLu*. Správnu triedu sieť označila v 5,04 %. Druhá bola sieť s *leaky ReLu*. Táto sieť označila správnu triedu v 3,68 %. Posledná je sieť s funkciou *tanh*. Označenie správnej triedy tvorí 2,33 %. Pri nesprávnej triede sa najlepšie darilo sieti s *ReLu*. Táto sieť nesprávnu triedu určila v 48,5 %. Za ňou nasleduje sieť s *tanh*. Táto sieť nesprávnu triedu klasifikovala v 48,77 %. Posledná je sieť s *leaky ReLu*. Klasifikovala nesprávnu triedu v 49,71 %. U sieti s *tanh* funkciou bola falošná pozitivita 48,9 %. Falošná pozitivita u sieti s *leaky ReLu* je 46,61 %. A sieť s *ReLu* klasifikovala falošnú pozitivitu u 46,46 % prípadov.



Obrázok 6.12: Test správneho určenia pre bradavicu nad testovacou databázou C a trénovacou A6.

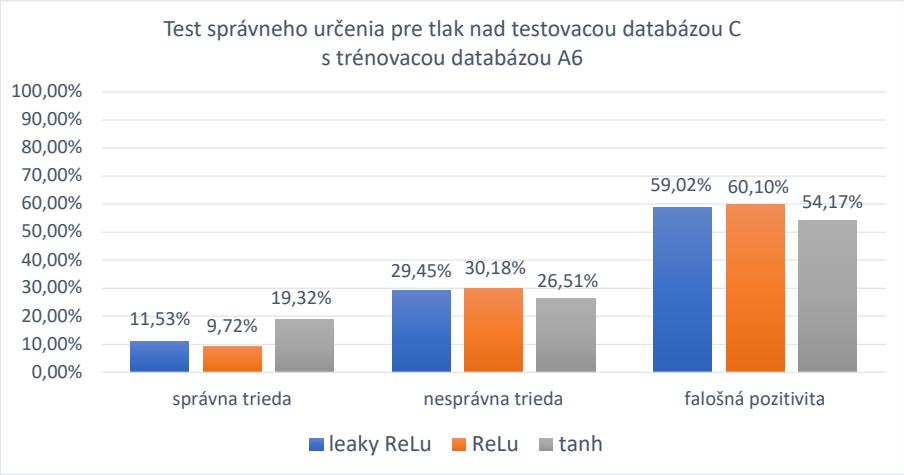
Poslednou sledovanou chorobou je **psoriáza**. Táto choroba má vo všetkých prípadoch 0 %. Znamená to, že pri tak nízkych počtoch odtlačkov prstov, sieť nebola schopná vôbec

rozpoznať správnu triedu. Dobré je, že keď odtlačok prsta nebude obsahovať psoriázu, tak ju vôbec neoznači ako falošne pozitívnu, alebo nesprávnu triedu. Prvým poškodením pri snímaní, ktorým sa test zaoberá je **vlhkosť**. Podobne ako u chorôb, tak aj pri poškodeniach budú výsledky zobrazené na grafe v obrázku 6.13. Z grafu je možné vidieť, že správnu triedu sa najlepšie podarilo klasifikovať sieti s *leaky ReLu*. Konkrétne 28,71 % úspešne klasifikovala ako správnu triedu. Druhá je sieť s *ReLu*. U tejto sieti to bolo 27,15 %. Najhoršie sa darilo sieti s *tanh*. Sieť klasifikovala správnu triedu v 20,5 % prípadoch. Nesprávnu triedu klasifikovala sieť s *tanh* v 79,5 % prípadoch. 30,29 % nesprávnej triedy klasifikovala sieť s *leaky ReLu*. Sieť s *ReLu* označila nesprávnu triedu v 29,3 % prípadoch. U falošnej pozitivity sa najlepšie darilo sieti s *tanh*. Táto sieť označila 0 % falošne pozitívnu triedu. U sieti s *leaky ReLu* to je 41 %. Sieť s *ReLu* označila falošne pozitívne 43,55 %.



Obrázok 6.13: Test správneho určenia pre vlhkosť nad testovacou databázou C a trénovacou A6.

Posledným skúmaným poškodením pri snímaní je **tlak**. Výsledky s testovania sú zobrazené na obrázku 6.14 vo forme grafu. Z tohto grafu vyplýva, že najlepšie správnu triedu určila sieť s *tanh*. U tejto sieti to je 19,32 %. Sieť s *leaky ReLu* klasifikovala správnu triedu v 11,53 % prípadoch. Klasifikovanie správnej triedy u sieti s *ReLu* je 9,72 %. Najmenej označenej nesprávnej triedy má sieť s *tanh*. U tejto sieti to činí 26,51 %. Druhá je sieť s *leaky ReLu*. U tejto sieti to je 29,45 %. Posledná je sieť s *ReLu*. Nesprávna trieda je klasifikovaná v 30,18 % prípadoch. U falošnej pozitivite sa najlepšie darilo sieti s *tanh*. U tejto sieti to činí 54,17 %. Za ňou je sieť s *leaky ReLu*. Táto sieť klasifikovala falošnú pozitivitu v 59,02 % prípadoch. Najhoršie je na tom sieť s *ReLu*. U tejto sieti to činí 60,1 %.

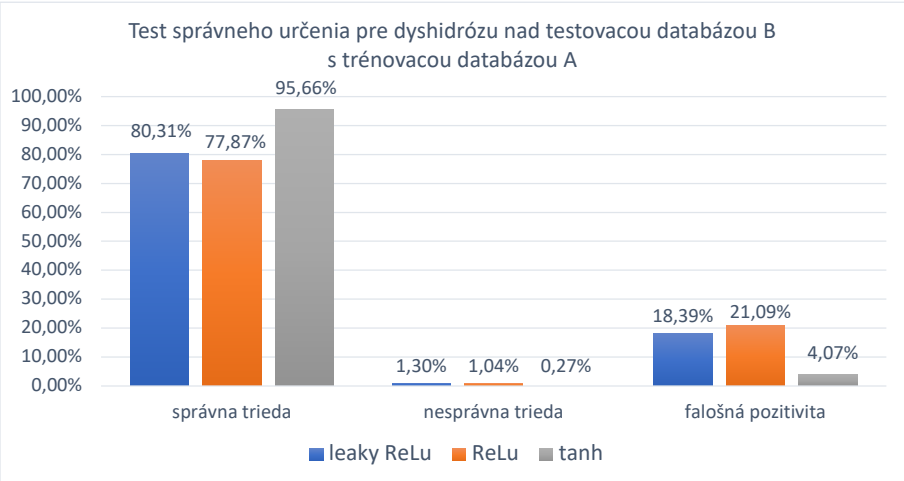


Obrázok 6.14: Test správneho určenia pre tlak nad testovacou databázou C a trénovacou A6.

6.3.5 Testovacia databáza B a trénovacia databáza A

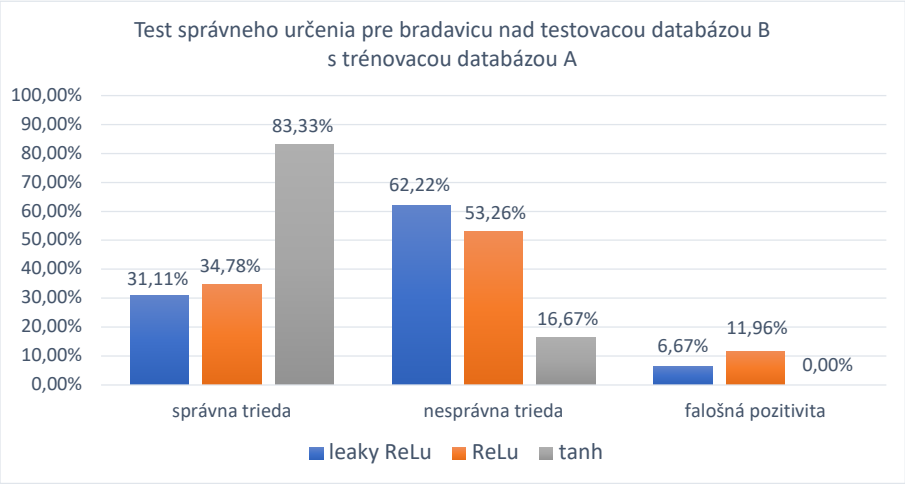
Táto podkapitola sa bude zaoberať testovaním nad testovacou databázou AB a trénovacou databázou A. Toto testovanie je potrebné pre porovnanie novej siete so sieťou z bakalárskej práce. Podobne ako v predchadzajúcich podkapitolách budú rozobraté choroby a to dyshidróza a bradavica. Tieto choroby vie rozoznať sieť z bakalárskej siete.

Prvé výsledky z testovania budú popísané pre chorobu *dyshidrózu*. Tie sú zobrazené na grafe v obrázku 6.15. Sieť s *leaky relu* klasifikovala správnu triedu v 80,31 % prípadoch. V 77,87 % klasifikovala sieť s *relu* ako správnu triedu. U sieti s *tanh* správna trieda činila 95,66 %. Nesprávna trieda bola najmenej určena u sieti s *tanh*. U tejto sieti to bolo 0,27 %. Následuje za ňou sieť s *relu*, ktorá nesprávnu triedu označila v 1,04 % prípadoch. Posledná je sieť s *leaky relu*. U tejto sieti to činilo 1,3 %. Falošná pozitivita u sieti s *leaky relu* činila 18,39 %. Sieť s *relu* aktivačnou funkciou označila u 21,09 % prípadov falošnú pozitivitu. 4,07 % klasifikovala sieť s *tanh* falošnú pozitivitu.



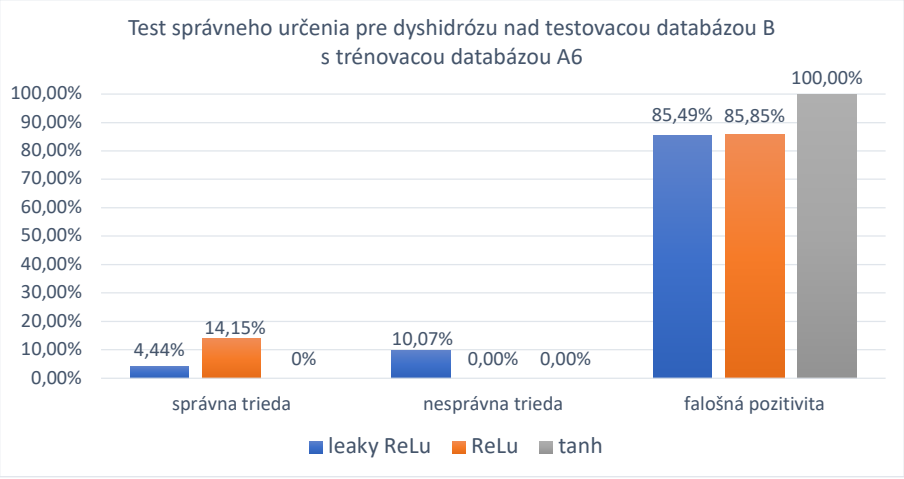
Obrázok 6.15: Test správneho určenia pre dyshidrózu nad testovacou databázou B a trénovacou A.

Druhou chorobou je **bradavica**. Výsledky sú zobrazené na obrázku 6.16. Na obrázku je vidieť graf, z ktorého sa dajú vyčítať nasledujúce informácie. Klasifikovanie správnej triedy u sieti s *relu* činilo 34,78 %. U sieti s *leaky relu* to bolo 31,11 %. Sieť s *tanh* klasifikovala správnu triedu v 83,33 % prípadoch. Sieť s aktivačnou funkciou *leaky relu* klasifikovala nesprávnu triedu u 62,22 % prípadov. Sieť s *relu* označila nesprávnu triedu v 53,26 % prípadoch. Sieť s *tanh* klasifikovala nesprávnu triedu v 16,67 % prípadoch. Najmenej falošnej pozitivity mala sieť s *tanh*. Táto sieť falošne pozitívne klasifikovala bradavicu v 0 % prípadoch. Následuje sieť s *leaky relu*. Táto sieť označila falošnou pozitivitou 6,67 % prípadov. Najhoršia bola sieť s *relu*. Táto sieť klasifikovala bradavicu v 11,96 % prípadoch označila ako falošne pozitívnu triedu.



Obrázok 6.16: Test správneho určenia pre bradavicu nad testovacou databázou B a trénovacou A.

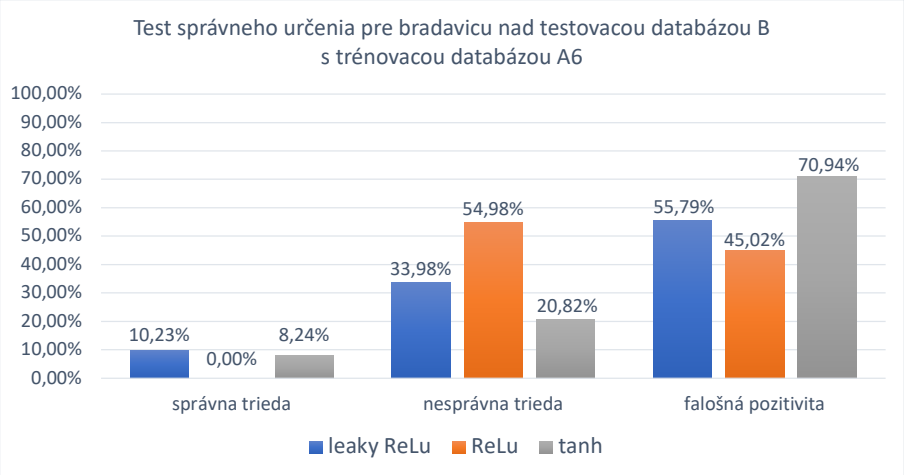
6.3.6 Testovacia databáza B a trénovacia databáza A6



Obrázok 6.17: Test správneho určenia pre dyshidrózu nad testovacou databázou B a trénovacou A6.

Podobne ako predchadzajúca podkapitola, tak aj táto sa bude zaoberať iba chorobou dyshidróza a bradavica. Prvé budú výsledky pre ochorenie **dyshidrózu** v grafe na obrázku 6.17. Sieť s *leaky ReLu* klasifikovala správnu triedu v 4,44 % prípadoch. 14,15 % klasifikovala sieť s *ReLu* ako správnu triedu. U sieti s *tanh* správna trieda činila 0 %. Nesprávna trieda bola najmenej určena u sieti s *ReLu*. U tejto sieti to bolo 0 %. Následuje za ňou sieť s *tanh*, ktorá nesprávnu triedu označila v 0 % prípadoch. Posledná je sieť s *leaky ReLu*. U tejto sieti to činilo 10,07 %. Falošná pozitivita u sieti s *leaky ReLu* činila 85,49 %. Sieť s *ReLu* aktivačnou funkciou označila u 85,85 % prípadov falošnú pozitivitu. 100 % klasifikovala sieť s *tanh* falošnú pozitivitu.u dyshidrózy.

Druhou chorobou je **bradavica**. Výsledky sú zobrazené na obrázku 6.18. Na obrázku je



Obrázok 6.18: Test správneho určenia pre bradavicu nad testovacou databázou B a trénovacou A6.

vidieť graf, z ktorého sa dajú vyčítať nasledujúce informácie. Klasifikovanie správnej triedy u sieti s *ReLu* činilo 0 %. U sieti s *leaky ReLu* to bolo 10,23 %. Sieť s *tanh* klasifikovala správnu triedu v 8,24 % prípadoch. Sieť s aktivačnou funkciou *leaky ReLu* klasifikovala nesprávnu triedu u 33,98 % prípadov. Sieť s *ReLu* označila nesprávnu triedu v 54,98 % prípadoch. Sieť s *tanh* klasifikovala nesprávnu triedu v 20,82 % prípadoch. Najmenej falošnej pozitivity mala sieť s *ReLu*. Táto sieť falošne pozitívne klasifikovala bradavicu v 45,02 % prípadoch. Následuje sieť s *leaky ReLu*. Táto sieť označila falošnou pozitivitou 55,79 % prípadov. Najhoršia bola sieť s *tanh*. Táto sieť klasifikovala bradavicu v 70,94 % prípadoch.

6.4 Testovanie na určenie plochy

Kapitola sa zameriava na testovanie s akou úspešnosťou sa podarilo označiť správne plochu choroby alebo poškodenia nad danými odtlačkami prstov. Pri tomto teste sa vychádza z implementácie testovania, ktoré je popísané v kapitole 5.7. Kapitola rozoberie v nasledujúcich šiestich podkapitolách testovanie podľa testovacej databázy a trénovacej databázy. Prvá podkapitola 6.4.1 rozoberie testovanie nad databázou AB a trénovacou databázou A. Druhá podkapitola 6.4.2 sa bude zameriavať na testovaciu databázu AB a nad trénovacou databázou A6. Nasledujúca podkapitola 6.4.3 zhodnotí testovanie nad testovacou databázou C a trénovacou databázou A. Štvrtá podkapitola 6.4.4, popíše testovanie testovacej databázy C a trénovacej databázy A6. Predposledná podkapitola 6.4.5 rozoberie testovanie

nad databázou B a trénovacou databázou A. Posledná podkapitola 6.4.6 popíše testovanie nad databázou B a trénovacou databázou A6.

6.4.1 Testovacia databáza AB a trénovacia databáza A

Podkapitola sa zameriava na výsledky testovania plochy nad testovacou databázou AB a trénovacou databázou A. Podkapitola rozoberie určenie plochy z pohľadu chôrob, aby sa dalo porovnávať medzi jednotlivými konvolučnými sieťami a ich aktivačnými funkciami. Kapitola bude rozoberať iba ochorenie dyshidrózu a ekzém. Ostatné ochorenia a poškodenia sa nad skúmanou databázou zachovali správne a neoznačili žiadnu plochu.

Výsledky pre ochorenie **dyshidrózu** je vidieť na tabuľke 6.1. Z tejto tabuľky je zrejmé, že najmenej plochy označenej mimo anotácie, má konvolučná sieť s aktivačnou funkciou *tanh*. Táto sieť označila iba 3,24 % z celkovej zdravej plochy odtlačkov prstov. Za ňou nasleduje sieť s aktivačnou funkciou *ReLu*, ktorá označila 14,59 % plochy. Najhoršie dopadla sieť s aktivačnou funkciou *leaky ReLu*. Táto sieť označila až 19,01 % z celkovej plochy zdravých odtlačkov prstov.

	Leaky ReLu	ReLu	Tanh
Podiel označenej plochy mimo anotácií	19,01 %	14,59 %	3,24 %

Tabuľka 6.1: Test plochy pre ochorenie dyshidrózu, testovacia databáza AB, sieť trénovaná s databázou A.

Rovnako sú zobrazené výsledky pre ochorenie **ekzém**. Tieto výsledky sú zobrazené v tabuľke 6.2. Z tejto tabuľky je vidieť, že pre ochorenie ekzém, opäť zvíťazila sieť s aktivačnou funkciou *tanh*. Pri chorobe ekzém podiel označenej plochy mimo anotácií činil len 2,16 %. Sieť s aktivačnou funkciou ReLu označila len o 0,35 % plochy viac ako sieť s aktivačnou funkciou *tanh*. Sieť s *leaky ReLu* označila oproti sieti s *tanh* o 0,6 % plochy viac.

	Leaky ReLu	ReLu	Tanh
Podiel označenej plochy mimo anotácií	2,76 %	2,51 %	2,16 %

Tabuľka 6.2: Test plochy pre ochorenie ekzém, testovacia databáza AB, sieť trénovaná s databázou A.

6.4.2 Testovacia databáza AB a trénovacia databáza A6

Podobne ako podkapitola 6.4.1 aj táto podkapitola bude rozdelená na popis výsledkov testovania rozdelených nad jednotlivými chorobami. V popise bude chýbať psoriáza, pretože sa siete zachovali správne.

Prvý výsledok je pre ochorenie **dyshidrózu** a je zobrazená v tabuľke 6.3. Z tejto tabuľky je vidieť, že najlepšie si viedla sieť s aktivačnou funkciou *tanh*. Táto sieť označila iba 6,67 % plochy. Druhé miesto obsadila sieť s aktivačnou funkciou *leaky ReLu*, ktorá označila 12,31 %. Posledná bola sieť s funkciou *ReLu*. Táto sieť označila dyshidrózu na celkovej ploche zdravých odtlačkov prstov 15,67 %.

Druhou pozorovanou chorobou je **bradavica**. Výsledky z nej sú zobrazené v tabuľke 6.4. Z tabuľky je vidieť, že opäť sa najlepšie darí sieti s aktivačnou funkciou *tanh*. Táto sieť označila 24,06 % plochy. Za ňou nasleduje sieť s funkciou *leaky ReLu*, ktorá označila 34,68 % plochy. Posledná je opäť sieť s funkciou *ReLu*. Táto sieť označila 35,27 % plochy.

	Leaky ReLu	ReLu	Tanh
Podiel označenej plochy mimo anotácií	12,31 %	15,67 %	6,67 %

Tabuľka 6.3: Test plochy pre ochorenie dyshidrózu, testovacia databáza AB, sieť trénovaná s databázou A6.

	Leaky ReLu	ReLu	Tanh
Podiel označenej plochy mimo anotácií	34,68 %	35,27 %	24,06 %

Tabuľka 6.4: Test plochy pre ochorenie bradavice, testovacia databáza AB, sieť trénovaná s databázou A6.

Ďalšou pozorovanou chorobou je **ekzém**. Výsledky z tohto testu sú zobrazené v tabuľke 6.5. Z tabuľky je vidieť, že opäť sa najlepšie darilo sieti s aktivačnou funkciou *tanh*. Táto sieť označila 41,61 % plochy. Za ňou nasleduje sieť s funkciou *ReLu*, ktorá označila 50,39 % plochy. Posledná je sieť s funkciou *leaky ReLu*. Táto sieť označila 51,79 % plochy.

	Leaky ReLu	ReLu	Tanh
Podiel označenej plochy mimo anotácií	51,79 %	50,39 %	41,61 %

Tabuľka 6.5: Test plochy pre ochorenie ekzém, testovacia databáza AB, sieť trénovaná s databázou A6.

Predposledný test je pre poškodenie pri snímaní, konkrétne **tlak**. Výsledky pre poškodenie je zobrazené v tabuľke 6.6. Z tabuľky vyplýva, že najlepšie sa darilo sieti s *leaky ReLu* aktivačnou funkciou. Táto sieť označila 6,37 %. Za ňou sa nachádza sieť s *ReLu* aktivačnou funkciou a označenou plochou 6,49 %. Najhoršie sa darilo sieti s aktivačnou funkciou *tanh* je to prvý test, v ktorom nie je najlepšia. Táto sieť označila až 34,82 % plochy.

	Leaky ReLu	ReLu	Tanh
Podiel označenej plochy mimo anotácií	6,37 %	6,49 %	34,82 %

Tabuľka 6.6: Test plochy pre poškodenie pri snímaní tlakom, testovacia databáza AB, sieť trénovaná s databázou A6.

Posledný test sa zameriava na poškodenie pri snímaní spôsobené **vlhkosťou**. Tento test je zobrazený v tabuľke 6.6. Z tabuľky je vidieť, že sieť s aktivačnou funkciou *tanh* sa zachovala správne a teda má označené 0 %. Za touto sieťou nasleduje sieť s funkciou *ReLu* s 2,84 % označenej plochy. Posledná sieť bola s funkciou *leaky ReLu*. Táto sieť označila až 21,28 %.

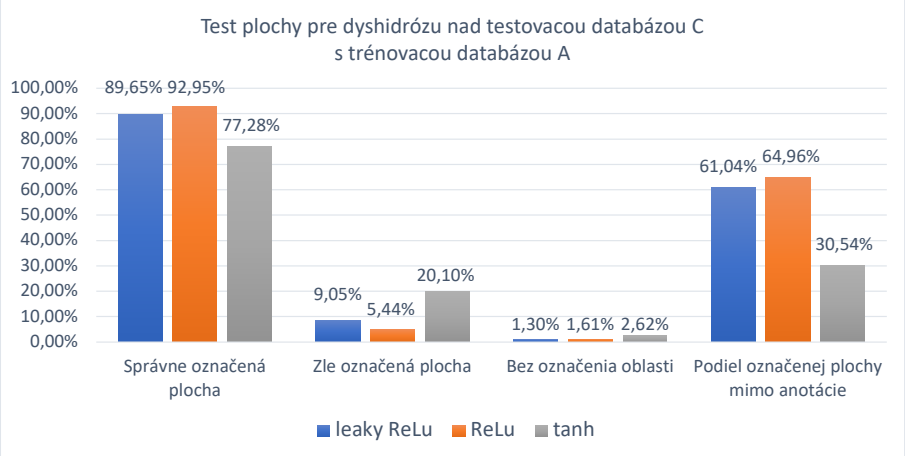
	Leaky ReLu	ReLu	Tanh
Podiel označenej plochy mimo anotácií	21,28 %	2,84 %	0 %

Tabuľka 6.7: Test plochy pre poškodenie pri snímaní vlhkosťou, testovacia databáza AB, sieť trénovaná s databázou A6.

6.4.3 Testovacia databáza C a trénovacia databáza A

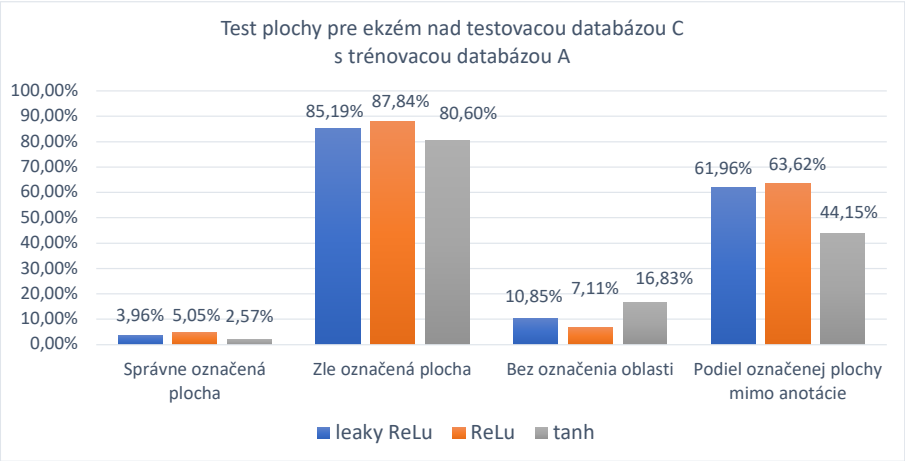
Podkapitola sa bude zameriavať na testovanie plochy nad testovacou databázou C a trénovacou databázou A. Výsledky budú znázornené pomocou grafu. Pri použití grafu je vhodné si popísať, čo je na grafe možné vidieť. Graf sa skladá zo štyroch klastrov, ktoré popisujú správne určenú plochu, nesprávne určenú plochu, neoznačenú plochu a označenú plochu mimo anotácie. V každom klastri sú vidieť, siete s troma rôznymi aktivačnými funkciami. Podkapitola bude popisovať jednotlivé výsledky pre dané choroby a poškodenia pri snímaní samostatne.

Prvé skúmané ochorenie je **dyshidróza**. Pre kopec údajov v grafe, bude popísaná najlepšia a najhoršia varianta. Test na plochu je znázornený na grafe v obrázku 6.19. Z grafu je vidieť, že najlepšie sa darilo určiť plochu pre sieť s aktivačnou funkciou *ReLu*. Táto sieť správne klasifikovala plochu v 92,95 % prípadoch. Najhoršie bola sieť s *tanh*. U tejto sieti to činilo u 77,28 % prípadov z celkovej plochy. Najmenej nesprávnej plochy klasifikovala sieť s *ReLu*. Táto sieť klasifikovala nesprávnu plochu iba v 5,44 %. Najviac nesprávnej označenej plochy mala sieť s *tanh*. Táto sieť označila nesprávne plochu v 20,10 % prípadoch plochy. Najmenej neoznačenej plochy mala sieť s *leaky ReLu*. Sieť neoznačila 1,3 % plochy. Najviac neoznačenej plochy mala sieť s *tanh*. Sieť neoznačila 2,62 % plochy, ktorá je ochorením dyshidróza. Najmenej plochy označenej mimo anotácii klasifikovala sieť s *tanh*. Táto sieť označila 30,54 % plochy mimo anotácie. Najviac to bolo u sieti s *ReLu*. Sieť označila plochu mimo anotáciu v 64,96 % plochy.



Obrázok 6.19: Test plochy pre dyshidrózu nad testovacou databázou C a trénovacou databázou A.

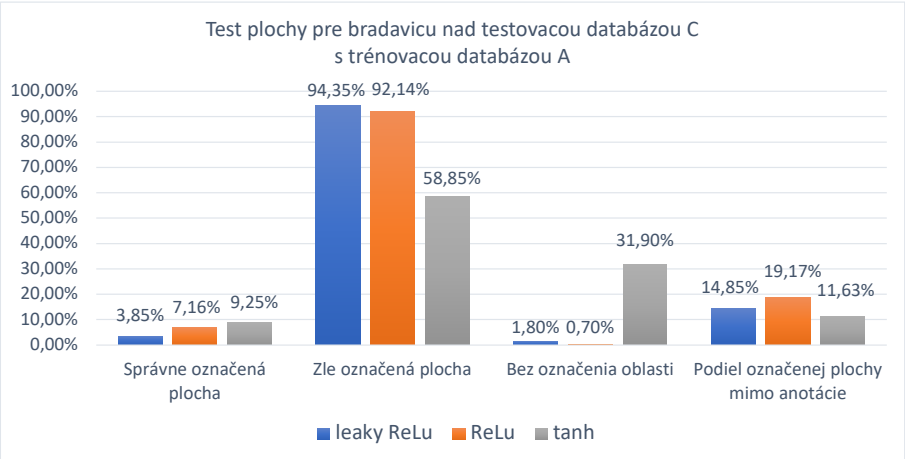
Ďalšou skúmanou chorobou je **ekzém**. Podobne ako pri dyshidróze, tak aj tu budú výsledky vo forme grafu, ktorý je vidieť na obrázku 6.20. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *ReLu*. Sieť správne označila 5,05 % plochy. Najmenej správne označenej plochy má sieť s *tanh*. Táto sieť správne označila iba 2,57 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *tanh*. Táto sieť zle označila 80,6 % plochy. Najviac zle označenej plochy má sieť s *ReLu*. Zle označenej plochy činilo 87,84 %. Najmenej neoznačenej plochy má sieť s *ReLu*. Neoznačená plocha činila 7,11 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 16,83 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť



Obrázok 6.20: Test plochy pre ekzém nad testovacou databázou C a trénovacou databázou A.

s *tanh*. Činilo to 44,15 % plochy. Najviac označenej plochy mimo anotácie má sieť s *ReLu*. Táto sieť označila mimo anotácie 63,62 % plochy.

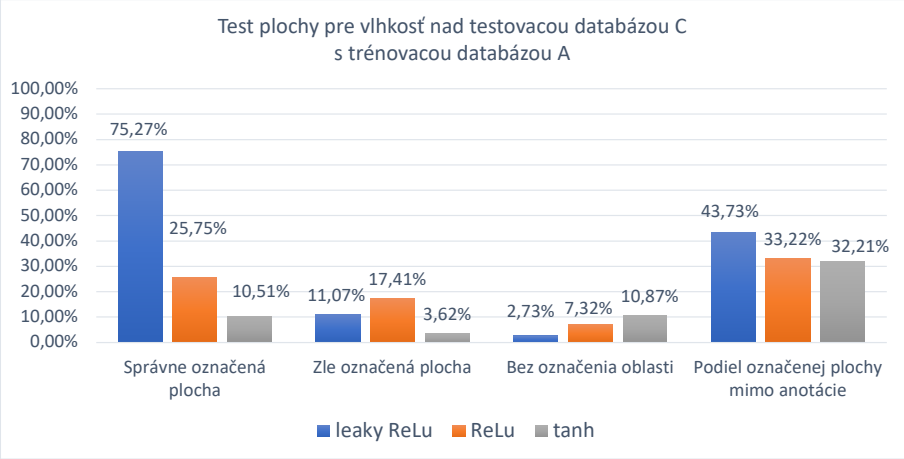
Predposlednou skúmanou chorobou je **bradavica**. Graf tohto výsledku je zobrazený na obrázku 6.21. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *tanh*. Sieť správne označila 9,25 % plochy. Najmenej správne označenej plochy má sieť s *leaky ReLu*. Táto sieť správne označila iba 3,85 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *tanh*. Táto sieť zle označila 58,85 % plochy. Najviac zle označenej plochy má sieť s *leaky ReLu*. Zle označenej plochy činilo 94,35 %. Najmenej neoznačenej plochy má sieť s *ReLu*. Neoznačená plocha činila 0,7 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 31,9% plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *tanh*. Činilo to 11,63 % plochy. Najviac označenej plochy mimo anotácie má sieť s *ReLu*. Táto sieť označila mimo anotácie 19,17 % plochy.



Obrázok 6.21: Test plochy pre bradavicu nad testovacou databázou C a trénovacou databázou A.

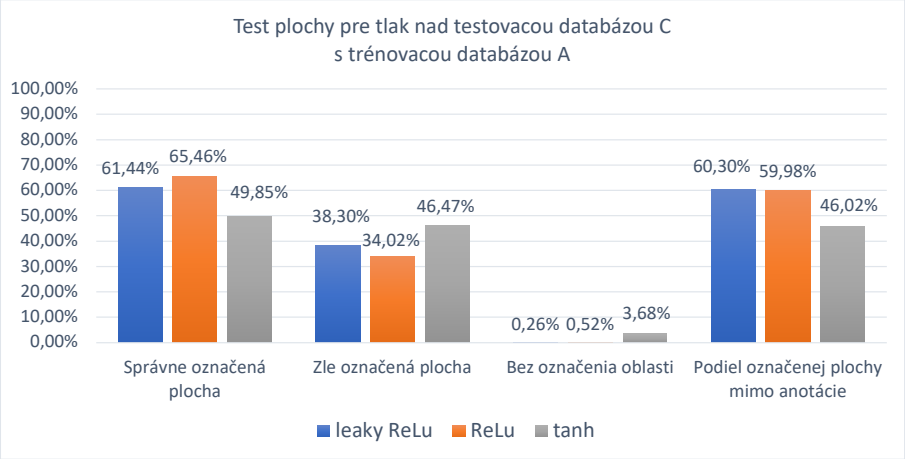
Posledná choroba je **psoriáza**. Táto choroba má iba jednu kolonku nenulovú, a to je neoznačnená plocha. Pri sieti s *leaky ReLu* to je 4,56 % plochy. Znamená to, že pri tejto sieti bola zvyšná plocha označená inými chorobami. U sieti s *ReLu* to bolo 3,05 %. A pri sieti s *tanh* to bolo 2,71 % plochy.

Prvé skúmané poškodenie pri snímaní je **vlhkosť**. Výsledky z testovania sú zobrazené na grafe, ktorý je zobrazený na obrázku 6.22. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *leaky ReLu*. Sieť správne označila 75,27 % plochy. Najmenej správne označenej plochy má sieť s *tanh*. Táto sieť správne označila 10,51 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *tanh*. Táto sieť zle označila 3,62 % plochy. Najviac zle označenej plochy má sieť s *ReLu*. Zle označenej plochy činilo 17,41 %. Najmenej neoznačenej plochy má sieť s *leaky ReLu*. Neoznačená plocha činila 2,73 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 10,87 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *tanh*. Podiel označenej plochy mimo anotácie bolo 32,21 % plochy. Najviac označenej plochy mimo anotácie má sieť s *leaky ReLu*. Táto sieť označila plochu mimo anotácie 43,73 % plochy.



Obrázok 6.22: Test plochy pre vlhkosť nad testovacou databázou C a trénovacou A.

Posledné skúmané poškodenie pri snímaní je **tlak**. Výsledky sú zobrazené v grafe na obrázku 6.23. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *ReLu*. Sieť správne označila 65,46 % plochy. Najmenej správne označenej plochy má sieť s *tanh*. Táto sieť správne označila iba 49,85 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *ReLu*. Táto sieť zle označila 34,02 % plochy. Najviac zle označenej plochy má sieť s *tanh*. Zle označenej plochy činilo 46,47 %. Najmenej neoznačenej plochy má sieť s *leaky ReLu*. Neoznačená plocha činila 0,26 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 3,68 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *tanh*. Činilo to 46,02 % plochy. Najviac neoznačnej plochy má sieť s *leaky ReLu*. Táto sieť označila plochu mimo anotácie v 60,3 % plochy.

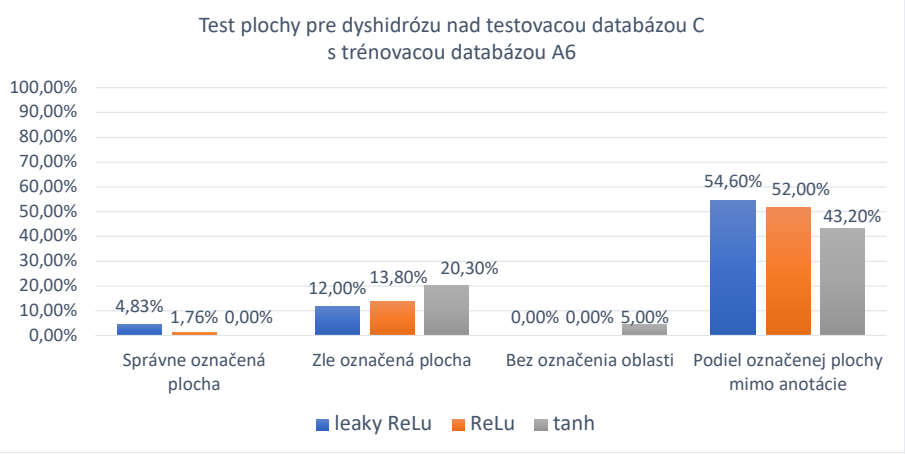


Obrázok 6.23: Test plochy pre vlhkosť nad testovacou databázou C a trénovacou databázou A.

6.4.4 Testovacia databáza C a trénovacia databáza A6

Podkapitola sa bude zameriavať na testovanie plochy nad testovacou databázou C a trénovacou databázou A6. Výsledky budú podobné, ako v predchádzajúcej podkapitole, znázornené pomocou grafu. Podkapitola bude popisovať jednotlivé výsledky pre dané choroby a poškodenia pri snímaní samostatne. Pre kopec údajov v grafe, bude popísaná najlepšia a najhoršia varianta.

Prvé skúmané ochorenie je **dyshidróza**. Test na plochu je znázornený na grafe v obrázku 6.24. Z grafu je vidieť, že najlepšie sa darilo určiť plochu pre sieť s aktivačnou funkciou *leaky ReLu*. Táto sieť správne klasifikovala plochu v 4,83 % prípadoch. Najhoršie bola sieť

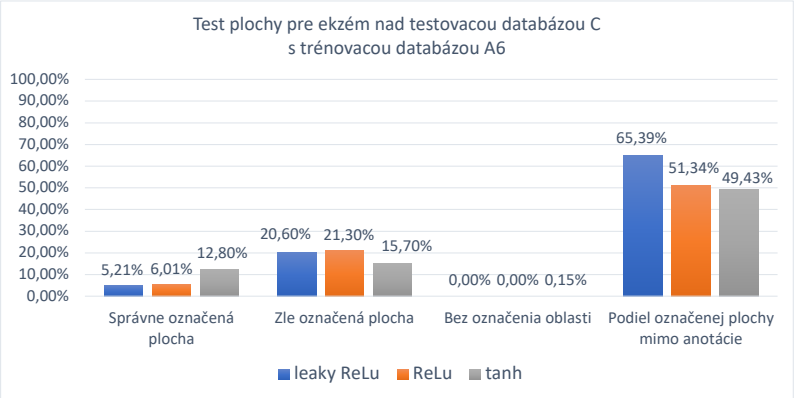


Obrázok 6.24: Test plochy pre dyshidrózu nad testovacou databázou C a trénovacou databázou A6.

s *tanh*. U tejto sieti to činilo 0 % prípadov z celkovej plochy. Najmenej nesprávnej plochy klasifikovala sieť s *leaky ReLu*. Táto sieť klasifikovala nesprávnu plochu iba v 12 %. Najviac nesprávnej označenej plochy mala sieť s *tanh*. Táto sieť označila nesprávne plochu v 20,30 % prípadoch plochy. Najmenej neoznačenej plochy mala sieť s *leaky ReLu*. Sieť neoznačila 0 %

plochy. Najviac neoznačenej plochy mala sieť s *tanh*. Sieť neoznačila 5 % plochy, ktorá je ochorením dyshidróza. Najmenej plochy označenej mimo anotácii klasifikovala sieť s *tanh*. Táto sieť označila 43,2 % plochy mimo anotácie. Najviac to bolo u sieti s *leaky ReLu*. Sieť označila plochu mimo anotáciu v 54,6 % plochy.

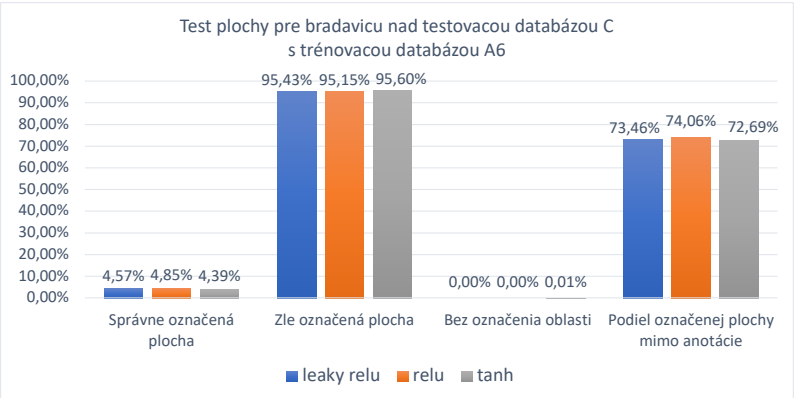
Ďalšou skúmanou chorobou je **ekzém**. Výsledky budú vo forme grafu, ktorý je vidieť na obrázku 6.25. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *tanh*. Sieť správne označila 12,8 % plochy. Najmenej správne označenej plochy má sieť s *leaky*



Obrázok 6.25: Test plochy pre ekzém nad testovacou databázou C a trénovacou databázou A6.

ReLu. Táto sieť správne označila 5,21 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *tanh*. Táto sieť zle označila 15,7 % plochy. Najviac zle označenej plochy má sieť s *ReLu*. Zle označenej plochy činilo 21,3 %. Najmenej neoznačenej plochy má sieť s *leaky ReLu*. Neoznačená plocha činila 0 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 0,15 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *tanh*. Konkrétne to bolo 49,43 % plochy. Najviac označenej plochy má sieť s *leaky ReLu*. Táto sieť označila mimo anotácie 65,39 % plochy.

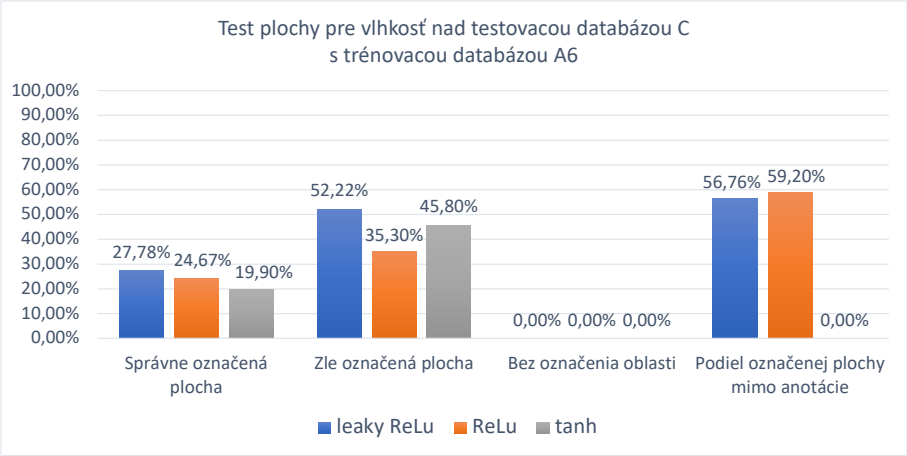
Predposlednou skúmanou chorobou je **bradavica**. Graf tohto výsledku je zobrazený na obrázku 6.26. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *ReLu*. Sieť



Obrázok 6.26: Test plochy pre bradavicu nad testovacou databázou C a trénovacou databázou A6.

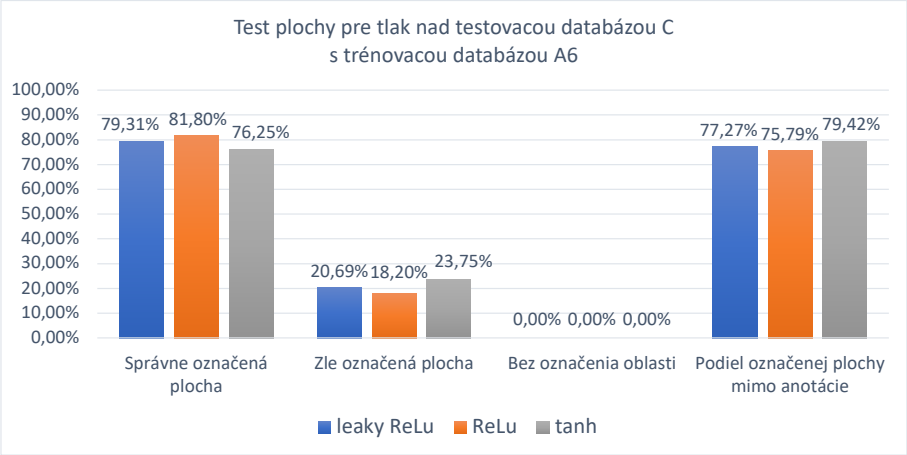
správne označila 4,85 % plochy. Najmenej správne označenej plochy má sieť s *tanh*. Táto sieť správne označila 4,39 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *ReLu*. Táto sieť zle označila 95,15 % plochy. Najviac zle označenej plochy má sieť s *tanh*. Zle označenej plochy činilo 95,6 %. Najmenej neoznačenej plochy má sieť s *leaky ReLu*. Neoznačená plocha činila 0 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 0,01 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *tanh*. Činilo to 72,69 % plochy. Najviac označenej plochy má sieť s *ReLu*. Táto sieť označila mimo anotácie 74,06 % plochy.

Posledná choroba je **psoriáza**. Všetky informácie má nulové. Hovorí to o tom, že sieť označila zlú chorobu na psoriáze Prvé skúmané poškodenie pri snímaní je **vlhkosť**. Výsledky z testovania sú zobrazené na grafe, ktorý je zobrazený na obrázku 6.27 Z grafu je vidieť, že najviac správne označenej plochy má sieť s *leaky ReLu*. Sieť správne označila 27,78 % plochy. Najmenej správne označenej plochy má sieť s *tanh*. Táto sieť správne označila 19,9 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *ReLu*. Táto sieť zle označila35,3 % plochy. Najviac zle označenej plochy má sieť s *leaky ReLu*. Zle označenej plochy činilo 52,22 %. Všetky tri siete mali plochu bez označenia 0 %. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *tanh*. Bolo to 0 % plochy. Najviac označnej plochy mimo anotáciu má sieť s *ReLu*. Táto sieť označila mimo anotácie 59,2 % plochy.



Obrázok 6.27: Test plochy pre vlhkosť nad testovacou databázou C a trénovacou databázou A6.

Posledné skúmané poškodenie pri snímaní je **tlak**. Výsledky sú zobrazené v grafe na obrázku 6.28. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *ReLu*. Sieť správne označila 81,8 % plochy. Najmenej správne označenej plochy má sieť s *tanh*. Táto sieť správne označila 76,25 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *ReLu*. Táto sieť zle označila 18,2 % plochy. Najviac zle označenej plochy má sieť s *tanh*. Zle označenej plochy činilo 23,75 %. Všetky tri siete majú 0 % bez označenej oblasti. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *ReLu*. Činilo to 75,79 % plochy. Najviac označnej plochy má sieť s *tanh*. Táto sieť označila mimo anotácie 79,42 % plochy.



Obrázok 6.28: Test plochy pre tlak nad testovacou databázou C a trénovacou databázou A6.

6.4.5 Testovacia databáza B a trénovacia databáza A

Podkapitola sa bude zameriavať na testovanie a výsledky nad testovacou databázou B a trénovacou databázou A. Pri tomto testovaní budú pozorované ochorenia dyshidróza a bradavica. Podkapitola bude rozoberať iba tieto choroby z dôvodu porovnania so sieťou z bakalárskej práce. Opäť výsledky budú zobrazené v grafe pre jednotlivé choroby.

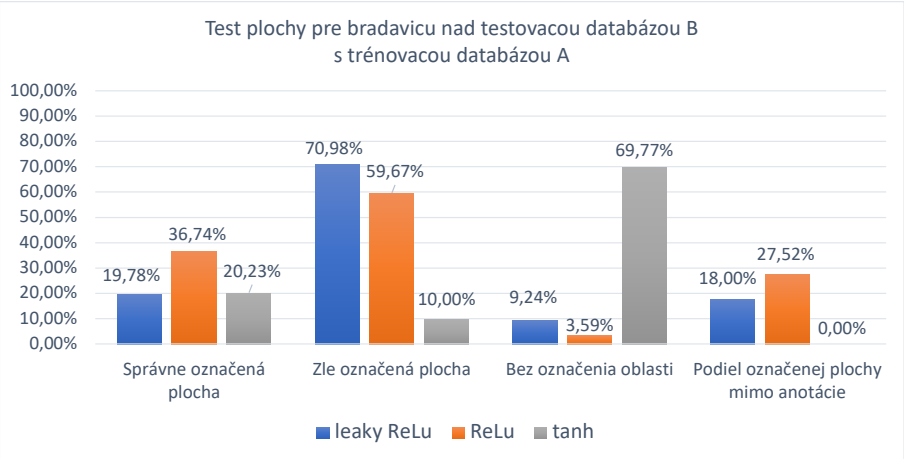
Prvou skúmanou chorobou je **dyshidróza**. Výsledky sú vo forme grafu, ktorý je vidieť na obrázku 6.29. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *ReLu*. Sieť

Obrázok 6.29: Test plochy pre dyshidrózu nad testovacou databázou B a trénovacou databázou A.

správne označila 97,62 % plochy. Najmenej správne označenej plochy má sieť s *tanh*. Táto sieť správne označila 95,72 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *ReLu*. Táto sieť zle označila 0,69 % plochy. Najviac zle označenej plochy má sieť s *leaky ReLu*. Zle označenej plochy činilo 1,42 %. Najmenej neoznačenej plochy má sieť s *leaky ReLu*. Neoznačená plocha činila 1,4 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 3,24 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *ReLu*. Podiel plochy bol 8,41 % . Najviac

označenej plochy mimo anotácie má sieť s *leaky ReLu*. Táto sieť označila mimo anotácie 55,18 % plochy.

Poslednou chorobou je **bradavica**. Graf tohto výsledku je zobrazený na obrázku 6.30. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *ReLu*. Sieť správne označila



Obrázok 6.30: Test plochy pre dyshidrózu nad testovacou databázou B a trénovacou databázou A.

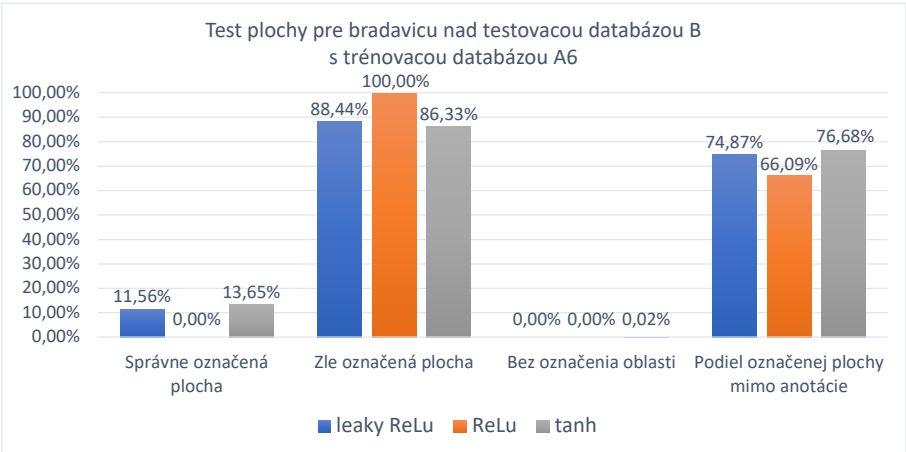
36,74 % plochy. Najmenej správne označenej plochy má sieť s *leaky ReLu*. Táto sieť správne označila 19,78 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *tanh*. Táto sieť zle označila 10 % plochy. Najviac zle označenej plochy má sieť s *leaky ReLu*. Zle označenej plochy činilo 70,98 %. Najmenej neoznačenej plochy má sieť s *ReLu*. Neoznačená plocha činila 3,59 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 69,77 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *tanh*. U tejto sieti to bolo 0 % plochy. Najviac označenej plochy mimo anotácie má sieť s *ReLu*. Táto sieť označila mimo anotácie 37,52 % plochy.

6.4.6 Testovacia databáza B a trénovacia databáza A6

Podkapitola sa bude zameriavať na testovanie a výsledky nad testovacou databázou B a trénovacou databázou A. Pri tomto testovaní budú pozorované ochorenia dyshidróza a bradavica. Prvou skúmanou chorobou je **dyshidróza**. Výsledky sú vo forme grafu, ktorý je vidieť na obrázku 6.31. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *ReLu*. Sieť správne označila 23,5 % plochy. Najmenej správne označenej plochy má sieť s *tanh*. Táto sieť správne označila 0 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *ReLu*. Táto sieť zle označila 0 % plochy. Najviac zle označenej plochy má sieť s *leaky ReLu*. Zle označenej plochy činilo 35,6 %. Najmenej neoznačenej plochy má sieť s *leaky ReLu*. Neoznačená plocha činila 0 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 3 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *leaky ReLu*. Podiel plochy bol 62,2 % . Najviac označenej plochy mimo anotácie má sieť s *ltanh*. Táto sieť označila mimo anotácie 77,32 % plochy.

Obrázok 6.31: Test plochy pre dyshidrózu nad testovacou databázou B a trénovacou databázou A6.

Poslednou chorobou je **bradavica**. Graf tohto výsledku je zobrazený na obrázku 6.32. Z grafu je vidieť, že najviac správne označenej plochy má sieť s *tanh*. Sieť správne označila 13,65 % plochy. Najmenej správne označenej plochy má sieť s *ReLu*. Táto sieť správne označila 0 % plochy. Najmenej zle označenej plochy má sieť s aktivačnou funkciou *tanh*. Táto sieť zle označila 86,33 % plochy. Najviac zle označenej plochy má sieť s *ReLu*. Zle označenej plochy činilo 100 %. Najmenej neoznačenej plochy má sieť s *ReLu*. Neoznačená plocha činila 0 %. Najviac neoznačenej plochy má sieť s *tanh*. Táto sieť neoznačila 0,02 % plochy. Posledný údaj je podiel označenej plochy mimo anotácie. Najmenej tejto plochy má označená sieť s *lReLu*. U tejto sieti to bolo 66,09 % plochy. Najviac označenej plochy mimo anotácie má sieť s *tanh*. Táto sieť označila mimo anotácie 76,68 % plochy.



Obrázok 6.32: Test plochy pre bradavicu nad testovacou databázou B a trénovacou databázou A6.

6.5 Testovanie siete z bakalárskej práce

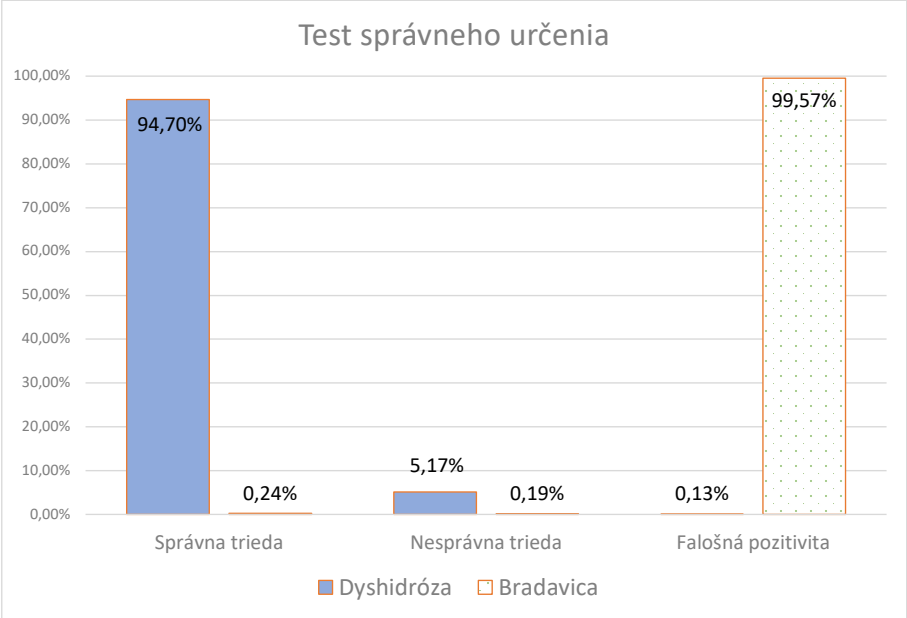
Podkapitola sa bude zaoberať testovaním siete z bakalárskej práce [70]. Neurónová sieť bude testovaná nad databázou AB a nad databázou B.

6.5.1 Test správneho určenia

Prvým testom bolo otestovať ako sa sieti darí pri zdravých odtlačkoch prstoch. Pri teste sa zistila 100 % falošná pozitivita u dyshidrózy, ale u bradavíc sa sieť zachovala výborne. Teda je falošná pozitivita 0 %. Znamená to, že na zdravých odtlačkoch sa aspoň raz detekovala dyshidróza. Nasledujúci obrázok 6.33 zobrazuje tri odtlačky prstov, ktoré boli čisté a sieť na nich označila dyshidrózu.

Obrázok 6.33: Príklad falošnej pozitivity nad zdravými odtlačkami prstov ochorenia dyshidrózy.

Druhý test prebiehal nad databázou B. Výsledky tohto testovania je možné vidieť na obrázku 6.34, na ktorom je graf ako sa darilo sieti správne, nesprávne alebo falošne pozitívne označiť chorobu. Z tohto grafu je vidieť, že správne určenie pre dyshidrózu je 94,7 %. Nesprávne sa podarilo určiť 5,17 %. Falošná pozitivita činila 0,13 %. Bradavicu sa podarilo správne určiť v 0,24 % prípadoch. Nesprávne sa podarilo určiť 0,19 %. U bradavíc je vysoká miera falošnej pozitivity, ktorá činila až 99,57 %. Z tohto druhého testu na správne určenie je zrejmé, že sieť veľmi presne vie určiť dyshidrózu. No pri bradaviciach je to veľmi veľký problém. Tam test správneho určenia neobsahuje ani 0,5 %. To znamená, že bradavicu nevie správne určiť, keď sa tam nachádza.



Obrázok 6.34: Test správneho určenia siete z bakalárskej práce nad databázou B.

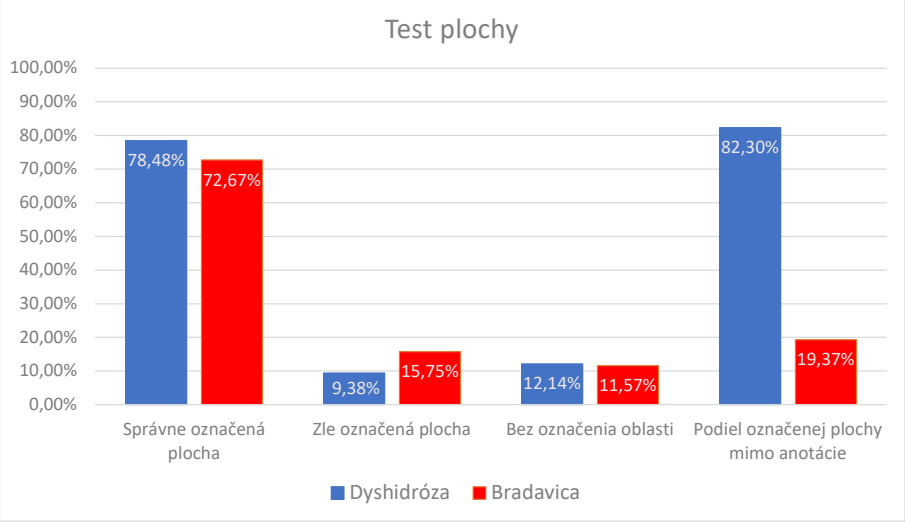
6.5.2 Test plochy

Ďalší test je test plochy. Pri testovaní siete nad databázou AB nás bude zaujímať iba podiel označenej oblasti mimo anotácie. Toto je zobrazené v tabuľke 6.8. U dyshidrózy to je 57,79 %. Podiel označenej plochy mimo anotácie pre bradavicou je 0 %. Podobne ako test správneho určenia choroby, tak aj test plochy je pre bradavicu výborný nad skúmanou databázou AB.

	Dyshidróza	Bradavice
Podiel označenej plochy mimo anotácie	57,79 %	0 %

Tabuľka 6.8: Test plochy, testovacia databáza AB, sieť z bakalárskej práce.

Posledný test je test plochy nad databázou B. Rovnako ako u teste správneho určenia aj toto je zobrazené na grafe, ktorý je vidieť na obrázku 6.35. Z tohto grafu je vidieť, že správne určiť dyshidrózu sa podarilo v 78,48 % prípadoch. Zle označená plocha činila v 9,38 % prípadoch. Bez označenia oblastí bolo 12,14 %. Tento údaj hovorí, koľko % plochy nebolo označenej, aj keď sa tam choroba nachádzala. Posledný dôležitý údaj je podiel označenej plochy mimo anotácie. Ten u dyshidrózy je 82,3 %. Z grafu je podobne vidieť, ako sa darilo sieti s bradavicou. U bradavice sa podarilo správne určiť 72,67 % plochy. Zle označená plocha činila 15,75 %. Bez označenia oblasti bolo 12,14 %. Pre bradavice je podiel označenej plochy mimo anotácie 19,37 %.



Obrázok 6.35: Test plochy siete z bakalárskej práce nad databázou B.

6.6 Zhodnotenie testovania

Podkapitola sa bude zaoberať vyhodnotením sady vylepšení, ktoré boli prevedené. Prvým vyhodnotením je zmenenie API Keras na PyTorch. Táto zmena priniesla veľkú úsporu uloženého modelu na disku. Model ušetrí o 59,67 % priestoru na disku. Zmena priniesla aj úsporu pri samotnej predikcii a klasifikácii. Pri našej testovacej databáze AB bola úspora času na predikciu a klasifikáciu o 62,5 %. Pri testovaní databázy B to bola časová úspora o 49,01 %.

Ďalšie vyhodnotenie bude zmena modelu na predtrénovaný model. Ako sa spomína v podkapitole 6.2, 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. Na nasledujúcich dvoch podkapitolách bude popísané zhodnotenie test správneho určenia a test plochy.

6.6.1 Test správneho určenia choroby alebo poškodenia

Prvé zhodnotenie bude testu správneho určenia nad trénovacou databázou A6. Výsledky testovania nad touto trénovacou databázou sú popísané v kapitolách 6.3.2, 6.3.6 a 6.3.4. Z týchto výsledkov je zrejmé, že použitie Gáborovho filtra neprinieslo vylepšenie. Použitie Gáborovho filtra prinieslo naopak zhoršenie. Nad zdravými odtlačkami našlo rôzne chroby a poškodenia, ktoré tam neboli. Záver z týchto testov je, že Gáborov filter nie je cesta k vylepšeniu.

Posledné zhodnotenie bude nad trénovacou databázou A. Toto zhodnotenie bude popísané zvlášť pre testovaciu databázu AB a testovaciu databázu C. Výsledky z testovania nad databázou AB sú popísané v podkapitole 6.3.1. Z tejto podkapitoly vyplýva, že nad čistými odtlačkami bola falošná pozitivita 100 % u dvoch chorôb. Konkrétne dyshidróza a ekzém. U ostatných chorobách a poškodeniach bola falošná pozitivita 0 %. Toto je veľmi pozitívne, že odtlačky prstov bez chorôb a poškodení zostanú neoznačené.

Po testovaní nad zdravými odtlačkami bolo prevedené testovanie nad testovacou databázou C. Tieto výsledky sú zobrazené v podkapitole 6.3.3. Výsledky testovania prebiehali nad jednotlivými chorobami. Prvým skúmaným ochorením je **dyshidróza**. Pri tomto ochorení sa najlepšie darilo sieti s aktivačnou funkciou *tanh*. Sieť označila správnu triedu v 56,33 % prípadoch. Nesprávnu triedu označila v 39,45 % prípadoch. U falošnej pozitivity to bolo 4,22 %. Nasledujúci obrázok 6.36 zobrazuje správne určenú triedu pre dyshidrózu.

Obrázok 6.36: Výsledok predikcie a klasifikácie dyshidrózy sieťou s aktivačnou funkciou tanh, testovacia databáza C, trénovacia databáza A.

Druhou skúmanou chorobou bol **ekzém**. U tejto choroby sa najlepšie darilo sieti s *ReLu*. Sieť označila správnu triedu v 3,24 % prípadoch. Nesprávnu triedu označila v 68,16 % prípadoch. U falošnej pozitivity to bolo 28,6 %. Falošnú pozitivitu mala nižšie sieť s *tanh*. Táto sieť označila falošne pozitívne 28,13 %. Ďalšou chorobou bola **psoriáza**. U tejto choroby všetky tri údaje boli 0 %. Znamená to, že sieť s tak malým počtom trénovacích vzorkov

nedokázala správne určiť psoriázu. Na druhej strane neoznačuje nesprávnu triedu ani falošnú pozitivitu, čo je z pohľadu nesprávnosti výborné. Poslednou skúmanou chorobou je bradavica. Najlepšie sa darilo sieti s *tanh*. Táto sieť označila správne 22,73 % prípadov. Nesprávnu triedu označila v 72,37 % prípadoch. Falošná pozitivita u tejto sieti bola 4,9 %. Prvým sledovaným poškodením pri snímaní je **vlhkosť**. U tohto poškodenia správnu triedu najlepšie určila sieť s *ReLu*. Táto sieť označila 47,75 % prípadov. Pri nesprávnej triede bola najhoršia a označila až 42,14 %. Najmenej klasifikovanú nesprávnu triedu má sieť s *tanh*. Táto sieť klasifikovala 27,36 % prípadov. Falošnú pozitivitu mala opäť najlepšiu sieť s *ReLu*. U tejto sieti to bolo 10,11 %. Posledné poškodenie pri snímaní je **tlak**. Najviac správnej triedy určila sieť s *ReLu*. Táto sieť označila 45,38 % prípadov. Zároveň určila najmenej nesprávnej triedy a to 40,59 % prípadov. Falošnú pozitivitu určila v 14,03 % prípadov, ale najmenej mala sieť s *tanh*. Táto sieť označila iba 6,81 % prípadov falošne pozitívne. Na nasledujúcom obrázku 6.37 bude zobrazená klasifikácia a predikcia pre tlak. Odtlačok prsta vľavo je odtlačok prsta s tlakom a na pravo je odtlačok predikovaný a klasifikovaný. Pre tlak je nastavená žltá farba.

Obrázok 6.37: Výsledok predikcie a klasifikácie pre tlak sieťou s aktivačnou funkciou relu, testovacia databáza C, trénovacia databáza A.

6.6.2 Test plochy choroby alebo poškodenia

Podobne ako test správneho určenia, tak aj test plochy bol rozdelený do podkapitol podľa trénovacej a testovacej databázy. Prvým testom plochy sú všetky databázy nad trénovacou databázou A6. Tieto testy sú popísané v kapitolách 6.4.2 6.4.4 a 6.4.6. Podobne ako pri teste správneho určenia, tak aj test plochy dopadol nad touto trénovacou databázou zle. Z tohto dôvodu, nie je dobre použiť Gáborov filter.

Druhý test plochy prebiehal nad trénovacou databázou A. Pri tejto databáze sú výsledky omnoho lepšie. Prvou skúmanou databázou je testovacia databáza AB. Testovanie je popísané v kapitole 6.4.1. Nad touto databázou boli označené len dve choroby. Konkrétne dyshidróza a ekzém. Najmenej podielu označenej plochy pre dyshidrózu mala sieť s *tanh*. Tento podiel činil 3,24 %. Pre chorobu ekzém je to tiež sieť s *tanh*. Označila 2,16 % podielu označenej plochy mimo anotácie. Nad zdravými odtlačkami jednoznačne vyhrála sieť s aktivačnou funkciou *tanh*.

Následuje zhodnotenie testovania nad databázou C.Testovanie je popísané v kapitole 6.4.3. Pre túto databázu budú popísane jednotlivé choroby zvlášť. Rovnako ako v predchadzajúcej podkapitole, tak aj tu sa budú popisovať najlepšie výsledky. Prvé skúmané ochorenie je **dyshidróza**. Najviac označenej správnej plochy a najmenej označenej zlej plochy mala

sieť s *ReLu*. Sieť správnu plochu označila 92,95 %. Zlej označenej plochy mala len 5,44 %. Bez označenia najmenej mala sieť s *leaky ReLu*. Táto sieť neoznačila 1,3 %. Tento údaj môže skresľovať fakt, že síce je najmenej bez označenia plochy, ale táto plocha môže byť označená inou chorobou. Dôležité údaje je správne označená plocha, zle označená plocha a podiel označenej plochy mimo anotácie. Ale je dobré vedieť aj podiel neoznačenej plochy. Najviac podielu plochy mimo anotácie má sieť s ReLu. Táto sieť označila 64,96 %. Najmenej označenej plochy mimo anotácie má sieť s tanh. Táto sieť označila 30,54 %.

Druhé skúmané ochorenie je **ekzém**. Najviac označenej správnej plochy mala sieť s *ReLu*. Táto sieť označila 5,05 %. Najmenej zlej označenej plochy mala sieť s *tanh*. Táto sieť klasifikovala 80,6 % zle označenej plochy. Bez označenia najmenej mala sieť s *ReLu*. Táto sieť neoznačila 7,11 %. Najviac podielu plochy mimo anotácie má sieť s ReLu. Táto sieť označila 63,62 %. Najmenej označenej plochy mimo anotácie má sieť s tanh. Táto sieť označila 44,15 %. Ďalším skúmaným ochorením je **bradavica**. Najviac označenej správnej plochy mala sieť s *tanh*. Táto sieť označila 9,25 %. Najmenej zlej označenej plochy mala sieť s *tanh*. Táto sieť klasifikovala 58,85 % zle označenej plochy. Bez označenia najmenej mala sieť s *ReLu*. Táto sieť neoznačila 0,7 %. Najviac podielu plochy mimo anotácie má sieť s ReLu. Táto sieť označila 19,17 %. Najmenej označenej plochy mimo anotácie má sieť s tanh. Táto sieť označila 11,63 %. Posledná choroba je **psoriáza**. Táto choroba má iba jednu kolonku nenulovú a to je neoznačnená plocha. Pri sieti s *leaky ReLu* to je 4,56 % plochy. Znamená to, že pri tejto sieti bola zvyšná plocha označená inými chorobami. U sieti s *ReLu* to bolo 3,05 %. A pri sieti s *tanh* to bolo 2,71 % plochy.

Prvým skúmaným poškodením pri snímaní je **vlhkosť**. Najviac označenej správnej plochy mala sieť s *leaky ReLu*. Táto sieť označila 75,27 %. Najmenej zlej označenej plochy mala sieť s *leaky ReLu*. Označila zle iba 2,73 %. Bez označenia najmenej mala sieť s *leaky ReLu*. Táto sieť neoznačila 2,73 %. Najviac podielu plochy mimo anotácie má sieť s leaky ReLu. Táto sieť označila 43,73 %. Najmenej označenej plochy mimo anotácie má sieť s tanh. Táto sieť označila 32,21 %. Posledným skúmaným poškodením pri snímaní je **tlak**. Najviac označenej správnej plochy mala sieť s *ReLu*. Táto sieť označila 61,44 %. Najmenej zlej označenej plochy mala sieť s *ReLu*. Táto sieť označila 34,02 %. Bez označenia najmenej mala sieť s *leaky ReLu*. Táto sieť neoznačila 0,26 %. Najviac podielu plochy mimo anotácie má sieť s ReLu. Táto sieť označila 60,3 %. Najmenej označenej plochy mimo anotácie má sieť s tanh. Táto sieť označila 46,02 %.

6.6.3 Porovnanie so sieťou z bakalárskej práce

Posledné zhodnotenie je porovnanie novej neurónovej siete a siete z bakalárskej siete. Pri tomto porovnaní sa vychádza z podkapitoly 6.3.1 6.3.5 6.4.1 6.4.5 6.5.1 6.5.2. Podobne ako v predchadzajúcich podkapitolách, tak testovanie nebude nad trénovacou databázou A6. Ukázalo sa, že Gáborov filter nepomáha zlepšiť riešenie, ale ho kazí. Z podkapitol sa vybrala sieť s aktivačnou funkciou *ReLu*. Síce má horšie výsledky nad zdravými odtlačkami, ale má najviac označenej plochy pre dyshidrózu a bradavicu. U bradavice sieť s *tanh*, má síce určenie správnej triedy v 83,33 %, no má najmenej správne označenej plochy.

Prvým zhodnotením bude nad databázou AB. Nová neurónová sieť ako aj sieť z bakalárskej práce označili **dyshidrózu** falošne pozitívnu. U **bradavíc** naštastie u obidvoch sieťach bola falošná pozitivita 0 %. Nová sieť ešte označila falošne pozitívnu chorobu ekzém. Táto choroba je rozšírením stávajúcej siete. No dôležitejšie je podiel označenej plochy mimo anotácie. Nová sieť s funkciou *ReLu* označila 19,01 % plochy ako dyshidrózu. Pri sieti s funkciou *tanh* to bolo len 3,24 % plochy. Sieť z bakalárskej siete mala 57,79 % plochy.

Pri použití sieti s *tanh* sa podiel označenej plochy mimo anotácie zmenšil o 54,49 %. Pri použití sieti s *ReLu* to je o 38.78 % označenej plochy menej. Nad databázou AB došlo k výraznému zlepšeniu sieti. Výsledok z predikcie a klasifikácie nad databázou AB je možné vidieť na obrázku 6.38.U bakalárskej práce je dyshidróza modrou farbou a u sieti s *ReLu* je dyshidróza označená červenou farbou.

(a) Konvolučná sieť s ReLu.

(b) Konvolučná sieť s tanh.

(c) Sieť z bakalárskej práce.

Obrázok 6.38: Príklad falošnej pozitivity nad zdravými odtlačkami prstov.

Posledným zhodnotením je testovanie nad databázou B. Pri tomto vyhodnotení sa bude porovnávať nová sieť s aktivačnou funkciou *ReLu* a *tanh*. Prvou chorobou je **dyshidróza**. Sieť z bakalárskej práce správne určila triedu dyshidrózy v 94,7 % prípadoch. Nová sieť s *ReLu* označila správnu triedu v 77,87 % prípadoch. U novej sieti s *tanh* klasifikovala správnu triedu v 95,66 % prípadoch. Nesprávne sa podarilo určiť 5,17 % pomocou siete z bakalárskej práce. U tejto sieti falošná pozitivita činila 0,13 %. Nesprávne sa podarilo určiť 1,04 % pomocou siete *ReLu*. Nesprávne sa podarilo určiť 0,27 % pomocou siete *tanh*. U sieti s *ReLu* falošná pozitivita činila 21,09 %. Pri sieti s *tanh* bola falošná pozitivita 4,07 %. Nová sieť s *ReLu* správne určila plochu v 97,62 %. Sieť s *tanh* označila správne plochu v 95,72 %. Sieť s bakalárskej práce označila správnu plochu v 78,48 %. Nová sieť s *ReLu* nesprávne určila plochu v 0,69 %. Sieť s *tanh* označila nesprávne plochu v 1,04 %. Sieť z bakalárskej práce označila nesprávnu plochu v 9,38 %. Nová sieť s *ReLu* vôbec neoznačila plochu v 1,69 %. Sieť s *tanh* vôbec neoznačila plochu v 3,24 %. Sieť z bakalárskej práce vôbec neoznačila plochu v 12,14 %. Nová sieť s *ReLu* označila plochu mimo anotácií v 8,41 %. Sieť s *tanh* označila plochu mimo anotácií v 24,49 %. Sieť z bakalárskej práce označila plochu mimo anotácií v 82,3 %. Nasledujúci obrázok 6.39 zobrazuje predikciu a klasifikáciu dyshidrózy. U bakalárskej práce je dyshidróza modrou farbou a u novej sieti s červenou farbou.

Posledná choroba je **bradavica**. Sieť z bakalárskej práce správne určila triedu v 0,24 % prípadoch. Nová sieť s *ReLu* označila správnu triedu v 34,78 % prípadoch. U novej sieti s *tanh* klasifikovala správnu triedu v 83,33 % prípadoch. Nesprávne sa podarilo určiť 0,19 % pomocou siete z bakalárskej práce. U tejto sieti falošná pozitivita činila 99,57 %. Nesprávne sa podarilo určiť 53,26 % pomocou siete *ReLu*. Nesprávne sa podarilo určiť 16,67 % pomocou siete *tanh*. U sieti s *ReLu* falošná pozitivita činila 11,96 %. Pri sieti s *tanh* bola falošná pozitivita 0 %. Nová sieť s *ReLu* správne určila plochu v 36,74 %. Sieť s *tanh* označila správne plochu v 20,23 %. Sieť z bakalárskej práce označila správnu plochu v 72,67 %. Nová sieť s *ReLu* nesprávne určila plochu v 59,67 %. Sieť s *tanh* označila nesprávne plochu v 10 %. Sieť z bakalárskej práce označila nesprávnu plochu v 15,75 %. Nová sieť s *ReLu* vôbec

(a) Konvolučná sieť s ReLu.

(b) Konvolučná sieť s tanh.

(c) Sieť z bakalárskej práce.

Obrázok 6.39: Príklad predikcie a klasifikácie dyshidrózy.

neoznačila plochu v 3,59 %. Sieť s *tanh* vôbec neoznačila plochu v 69,77 %. Sieť z bakalárskej práce vôbec neoznačila plochu v 12,14 %. Nová sieť s *ReLu* označila plochu mimo anotácií v 27,52 %. Sieť s *tanh* označila plochu mimo anotácií v 0 %. Sieť z bakalárskej práce označila plochu mimo anotácií v 19,37 %. Na obrázku 6.40 je zobrazená detekcia a klasifikácia pre bradavicu. U bakalárskej práce je bradavica červenou farbou a u novej sieti s červenou farbou.je dyshidróza, cyan farbou ekzém a modrou bradavica.

(a) Konvolučná sieť s ReLu.

(b) Konvolučná sieť s tanh.

(c) Sieť z bakalárskej práce.

Obrázok 6.40: Príklad predikcie a klasifikácie bradavice.

6.6.4 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 a 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 vylepšenie a rozšírenie predikcie a klasifikácie poškodenia odtlačku prsta pomocou konvolučných neurónových sietí. Práca nadväzuje na bakalársku prácu [70]. Najskôr ako sa mohlo začať nadväzovať na túto prácu, bolo potrebné naštudovať danú problematiku. Prvou problematikou je naštudovanie samotnej biometrie a pojmu odtlačku prstu. Pri odtlačkoch prstov 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 vylepšením a rozšírením 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 [70], pridanie predspracovanie obrázku pomocou Gáborovho filtra. Následne bolo zmenené rozmazanie na Gausovo a bolo použité binárne prahovanie s OTSU prahovaním. Zároveň boli vytvorené všeobecné nové algoritmy pre tvorbu databázy, trénovania, predikcie a klasifikácie, a neposlednom rade pre testovanie siete. Použité odtlačky prstov boli NIST DB4, syntetické odtlačky vygenerované pomocou SFinGe a reálne odtlačky prstov. Tieto odtlačky tvorili novú vygenerovanú databázu pre trénovanie, predikciu a klasifikáciu. Konkrétne sa jedná o ochorenia dyshidróza, bradavica, ekzém a psoriáza. Zároveň boli pridané odtlačky prstov pre poškodenia pri snímaní, ktoré sú tlak a vlhkosť. Dokopy je to rozšírenie o štyri nové typy rozpoznania v konvolučnej neurónovej sieti.

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 sade 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 predspracovanie obrázka pomocou Gáborovho filtra, zmenené rozmazanie a prahovanie. 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 implementovali nové všeobecné algoritmy pre tvorbu databázy. Následne sa implementovali nové všeobecné algoritmy pre trénovanie, kde sa implementovala zmena modelu na predtrénovaný. Pri tomto sa upravovali prvé konvolučné vrstvy, aby obrázok mohol byť poslaný na vstup v odtieňoch sivej. Zároveň bola implementovaná zmena aktivačnej funkcie. Pri týchto

zmenách sa implementovalo aj rozšírenie o nové typy ochorení a poškodení pri snímaní. Trénovací proces je o niečo zložitejší na implementáciu ako v API Keras.Táto implementácia uľahčuje používateľovi spustenie algoritmov. Nepotrebuje vypisovať jednotlivé parametre do argumentov príkazového riadku, ako to bolo pri bakalárskej práci. Nové algoritmy majú len jeden parameter. Týmto parametrom je cesta ku konfiguračnému súboru, v ktorom sú nastavené všetky potrebné parametre. Následne sa implementoval klasifikátor. Tento klasifikátor bol inšpirovaný bakalárskou prácou. Z pôvodného klasifikátora zostala len základná myšlienka kĺzavého okna pre klasifikátor. Zvyšok celý klasifikátor je prerobený na všeobecné použitie pomocou konfiguračného súboru. Myšlienky testovania ako test plochy a test správneho určenia je prebratý z bakalárskej práce, no opäť zostala len táto myšlienka. Implementácia je prerobená tak, aby bolo možné testovať klasifikátor, ktorý sa dokáže naučiť rôzne ochorenia alebo poškodenia. Po tomto všetkom sa mohlo prejsť k trénovaniu a vyhodnoteniu.

Prvé vyhodnotenie je zmena API Keras na PyTorch. Táto zmena priniesla úsporu uloženého modelu o 59,87 % priestoru na disku. Ušetril sa aj čas pri predikcii a klasifikácii. Pri predikcii a klasifikácii testovanej databázy so zdravými odtlačkami prstov bola úspora času o 62,5 %. Pri testovaní predikcie a klasifikácie databázy so zmiešanými odtlačkami prstov bola úspora času o 49,01 %. Druhé vyhodnotenie je zmena modelu na niektorý z predtrénovaných modelov. Táto zmena dopadla neúspechom. pretože všetky predtrénované modely sú trénované nad farebnými obrázkami. Bola vyskúšaná zmena prvej konvolučnej vrstvy, aby na vstup mohol byť privedený obrázok v odtieňoch sivej. Ďalším vyhodnotením je trénovanie siete s použitým Gáborovým filtrom. Toto vylepšenie neprinieslo zlepšenie., naopak výsledky boli zlé. Posledné dve vyhodnotenia sú nad sieťou, ktorá bola trénovaná bez Gáborovho filtra. Najskôr sa testovalo nad zdravými odtlačkami prstov a potom na zmiešaných odtlačkoch prstov. Pri teste nad zdravými odtlačkami prstov. všetký siete mali falošnú pozitivitu u dyshidrózy a ekzému. No najlepšie sa darilo sieti s aktivačnou funkciou *tanh*. Táto sieť označila plochu pre dyshidrózu len v 3,24 %. Pri ekzéme to bolo 2,16 %. Pri teste so zmiešanými odtlačkami prstov bola najlepšia sieť s aktivačnou funkciou *ReLu*. Tieto výsledky sú zobrazené pre test správneho určenia v tabuľke 7.1. Pre test plochy sú zobrazené v nasledujúcej tabuľke 7.2.

	Správna trieda	Nesprávna trieda	Falošná pozitivita
Dyshidróza	43,17 %	41,09 %	15,74 %
Ekzém	3,24 %	68,16 %	28,60 %
Psoriáza	0 %	0 %	0 %
Bradavica	7,32 %	85,58 %	7,10 %
Vlhkosť	47,75 %	42,14 %	10,11 %
Tlak	45,38 %	40,59 %	14,03 %

Tabuľka 7.1: Test správneho určenia pre sieť s *ReLu*. Testovacia databáza C, sieť trénovaná s databázou A.

	Správne označená plocha	Zle označená plocha	Bez označenia oblasti	Plocha mimo anotácie
Dyshidróza	92,95 %	5,44 %	1,61 %	64,96 %
Ekzém	5,05 %	87,84 %	7,11 %	63,62 %
Psoriáza	0 %	0 %	3,05 %	0 %
Bradavica	7,16 %	92,14 %	0,70%	19,17 %
Vlhkosť	25,75 %	17,41 %	7,32 %	33,22 %
Tlak	65,46 %	34,02 %	0,52 %	59,98 %

Tabuľka 7.2: Test plochy pre sieť s *ReLu*. Testovacia databáza C, sieť trénovaná s databázou A.

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 kol. *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. 740 s. 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] EUC. *Atopický ekzém – příčiny, příznaky a léčba* [online]. EUC [cit. 2022-04-12]. Dostupné z: https://euc.cz/clanky-a-novinky/clanky/atopicky-ekzem-priciny-priznaky-a-lecba.

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

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] GALTON, F. *Finger Prints*. Macmillian and CO. and New York, 1892. 216 s. Dostupné z: http://www.biometricbits.com/Galton-Fingerprints-1892.pdf.

[31] GOODFELLOW, I., BENGIO, Y. a COURVILLE, A. *Deep Learning*. MIT Press, 2016. Dostupné z: http://www.deeplearningbook.org.

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

[33] HABIF, T. P. *Clinical dermatology: a color guide to diagnosis and therapy*. 4th edition. 2004. 1024 s. ISBN 0-323-01319-8.

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

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

[36] 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.

[37] 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.

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

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

[40] JAIN, A. K., FLYNN, P. a ROSS, A. A. *Handbook of Biometrics*. Springer US, 2008. ISBN 978-0-387-71040-2.

[41] 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.

[42] 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.

[43] 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/.

[44] 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.

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

[46] 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.

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

[48] 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.

[49] 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.

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

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

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

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

[54] 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/.

[55] 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/.

[56] 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/.

[57] 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/.

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

[59] Munakata, T. *Fundamentals of the New Artificial Intelligence: Neural, Evolutionary, Fuzzy and More (Texts in Computer Science)*. 2nd ed. Springer, 2009. 267 s. ISBN 978-1-84628-838-8.

[60] 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.

[61] 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/.

[62] 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.

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

[64] SHARMA, S. *Epoch vs Batch Size vs Iterations* [online], 23. septembra 2017. Dostupné z: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

[65] 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.

[66] Svoradová, V. *Pokročilé generování projevů poškození do syntetických otisků prstů*. Brno, CZ, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/24014/.

[67] 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.

[68] 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.

[69] 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&.

[70] 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/.

[71] 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.

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

[73] 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.

[74] Yalçin, 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.

[75] 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.

[76] 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/.

[77] Štork, J. *Dermatovenerologie.* 2. vyd. Praha: Galén, 2013. 502 s. 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 FingerprintCnnModifiedModel(nn.Module):
  def ___init___(self, number_of_outputs, activation_function):
    super(FingerprintCnnModifiedModel, self).___init___()

   self.number_of_outputs = number_of_outputs
   self.activation_function = activation_function
   self.conv1 = nn.Conv2d(1, 64, (3, 3))
   self.conv2 = nn.Conv2d(64, 64, (3, 3))
   self.relu = nn.ReLU()
   self.leaky_relu = nn.LeakyReLU()
   self.tanh = nn.Tanh()
   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_more_diseases = nn.Linear(768, self.number_of_outputs)
   self.linear_one_disease = nn.Linear(768, 2)

  def forward(self, model):
    if self.activation_function == ActivationFunctionEnum.DEFAULT_RELU:
      return self.___get_model_with_relu(model)
    if self.activation_function == ActivationFunctionEnum.LEAKY_RELU:
      return self.___get_model_with_leaky_relu(model)
   if self.activation_function == ActivationFunctionEnum.TANH:
      return self.___get_model_with_tanh(model)

  def ___get_model_with_relu(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)
```

```
    model = self.softmax(self.linear_more_diseases(model))
    return model

  def ___get_model_with_leaky_relu(self, model):
    model = self.leaky_relu(self.conv1(model))
    model = self.leaky_relu(self.conv2(model))
    model = self.pool(model)
    model = self.leaky_relu(self.conv3(model))
    model = self.leaky_relu(self.conv4(model))
    model = self.pool(model)
    model = self.leaky_relu(self.conv5(model))
    model = self.leaky_relu(self.conv6(model))
    model = self.pool(model)
    model = self.leaky_relu(self.conv6(model))
    model = self.leaky_relu(self.conv6(model))
    model = self.pool(model)
    model = model.view(model.size(0), -1)
    model = self.leaky_relu(self.linear1(model))
    model = self.dropout(model)
    model = self.leaky_relu(self.linear2(model))
    model = self.dropout(model)
    model = self.softmax(self.linear_more_diseases(model))
    return model

  def ___get_model_with_tanh(self, model):
    model = self.tanh(self.conv1(model))
    model = self.tanh(self.conv2(model))
    model = self.pool(model)
    model = self.tanh(self.conv3(model))
    model = self.tanh(self.conv4(model))
    model = self.pool(model)
    model = self.tanh(self.conv5(model))
    model = self.tanh(self.conv6(model))
    model = self.pool(model)
    model = self.tanh(self.conv6(model))
    model = self.tanh(self.conv6(model))
    model = self.pool(model)
    model = model.view(model.size(0), -1)
    model = self.tanh(self.linear1(model))
    model = self.dropout(model)
    model = self.tanh(self.linear2(model))
    model = self.dropout(model)
    model = self.softmax(self.linear_more_diseases(model))
    return model
```

Algoritmus C.1: model v PyTorch.