VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE PRO ROZPOZNÁNÍ OSOB PODLE OBLIČEJE

APPLICATION FOR RECOGNITION OF PEOPLE BY FACE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JAKUB SVOBODA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ GOLDMANN**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Svoboda Jakub, Bc.**
Program: Informační technologie
Obor: Inteligentní systémy
Název: **Aplikace pro rozpoznání osob podle obličeje**
Application for Recognition of People by Face
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s problematikou identifikace osob na základě snímků obličeje z kamerových systému. Zjistěte, jakými nedostatky trpí moderní systémy pro rozpoznávání osob podle obličeje.
2. Nastudujte, jaké algoritmy strojového učení se používají pro rozpoznávání osob podle obličeje. Sumarizujte informace o moderních algoritmech pro rozpoznávání osob.
3. Navrhněte řešení, které bude provádět rozpoznávání osob podle obličeje. Předpokládejte, že obličej může být zaznamenám v různých pozicích.
4. Navržené řešení implementujte v programovacím jazyce Python a vytvořte k němu jednoduché uživatelské rozhraní.
5. Výsledné řešení otestujte a proveďte experimenty zaměřené na zhodnocení úspěšnosti identifikace.

Literatura:

- MASI, Iacopo, et al. Deep face recognition: A survey. In: *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*. IEEE, 2018. p. 471-478.
- SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. p. 815-823.
- SUN, Yi, et al. Deep learning face representation by joint identification-verification. In: *Advances in neural information processing systems*. 2014. p. 1988-1996.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

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

Vedoucí práce: **Goldmann Tomáš, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Identifikace osob se v posledních letech stává jedním z klíčových způsobů extrakce informací z obrazových dat. Tato práce je zaměřena na identifikaci osob podle snímků obličeje za využití neuronových sítí. Většina konvenčních algoritmů vykazuje vysokou míru nepřesností, zatímco využitím neuronových sítí lze dosáhnout výrazně robustnějších výsledků. V této práci jsme na základě znalostí získaných studiem problematiky vytvořili architekturu neuronové sítě, která je schopná provádět úlohu identifikace a verifikace osob podle obličeje. Tuto architekturu a proces trénování jsme následně vylepšili na základě mnoha experimentů. Výsledný model dosáhl přesnosti srovnatelné se současnými *state–of–the–art* modely. Dále jsme vytvořili demonstrační aplikaci pro vizualizaci výsledků a správu databáze osob. Znalosti získané z této práce mohou být použity pro vylepšení současných algoritmů pro identifikaci osob nebo ke zpřesnění modelů řešících podobné úlohy.

Abstract

Person identification has in the recent years gained notoriety as one of the most powerful ways of extracting information from image data. This thesis is focused on the task of human identification from facial photographs. To solve this task, we employ algorithms based on neural networks, which produce more robust results than traditional algorithms. In this thesis, we studied the common approaches for solving this problem and based on the gathered knowledge we created an architecture of a neural network trained to tackle the task of human identification and verification based on facial photographs. We have then further improved the model architecture and the training process by performing various experiments and observing the results. The final model has reached an accuracy comparable to other *state–of–the–art* models. Furthermore, we created a desktop application to demonstrate the results visually and to enable easier manipulation with the identity database. The knowledge gathered in this thesis can be used for improvements of current identification models or models modified for solving similar tasks.

Klíčová slova

Rozpoznání obličeje, počítačové vidění, strojové učení, neuronové sítě, umělá inteligence.

Keywords

Facial recognition, computer vision, machine learning, neural network, artificial intelligence.

Citace

SVOBODA, Jakub. *Aplikace pro rozpoznání osob podle obličeje*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Goldmann

Aplikace pro rozpoznání osob podle obličeje

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Tomáše Goldmanna. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jakub Svoboda
10. května 2021

Poděkování

Rád bych poděkoval panu Ing. Tomášovi Goldmannovi za cenné rady a věcné připomínky, které mi poskytl při tvorbě této diplomové práce.

Obsah

Kapitola 1

Úvod

Systém pro rozpoznání obličeje je algoritmus schopný porovnat digitální snímky s předpřipravenou databází a ověřit identitu dané osoby. Tato technologie nachází své využití u bezpečnostních systémů. Kamerový systém doplněný o chytrou identifikaci osob je schopný sledovat osoby přítomné v dané budově a je schopen rozpoznat potenciální narušitele. Může také sbírat statistická data, ze kterých je možné dále analyzovat chování osob v daném prostředí. Využití identifikačních systémů se projevuje také u mobilních technologií, kde podobné systémy slouží jako biometrická verifikace uživatele.

Tato práce je zaměřena na algoritmy, které jsou schopny rozpoznávat osoby na základě snímků obličeje. Ideální systém by byl schopný bezchybně nalézt hledanou osobu v databázi, nicméně současné systémy stále obsahují určitou míru chybovosti. Tyto algoritmy ale mohou být využity pro prvotní průchod databází, který by byl pro člověka příliš zdlouhavý. Zatímco tradiční algoritmy využívané pro tyto účely byly často náchylné na kvalitu pořízených snímků, řešení založená na neuronových sítích si v posledních letech díky větší robustnosti vydobyla nepostradatelnou pozici v oblasti počítačového vidění.

V rámci této práce byla vytvořena architektura neuronové sítě, která převádí vstupní fotografie obličeje na vektory reprezentující identitu dané osoby. Navržený způsob trénování této neuronové sítě zaručuje, že výsledné vektory rozdílných osob budou více rozdílné, než vektory stejné osoby. Tohoto lze pak využít pro přímé porovnání dvou identit při procesu rozpoznání osoby.

Rozdělení kapitol je následující. V kapitole 2 je popsán teoretický základ k počítačovému vidění. Podkapitola 2.2 se pak podrobněji zaměřuje na neuronové sítě. Jsou zde popsány typy úloh, které lze neuronovými sítěmi řešit, je zde také popsán matematický základ neuronových sítí. Další sekce se zabývají vrstvami neuronových sítí a procesem jejich učení. Podkapitola 2.3 je věnována současným *state–of–the–art* metodám počítačového vidění. Podrobněji jsou zde popsány současné konvoluční neuronové sítě. Jsou zde také charakterizovány metody zaměřující se na identifikaci osob v obraze.

V kapitole 3 je popsáno řešení problému. Podkapitola 3.1 rozebírá existující datasety, které jsou vhodné pro trénování algoritmů na identifikaci osob podle obličeje. Celkem jsou zde shrnuty informace k osmi datasetům. Jsou zde také popsány podmínky, za kterých byly fotografie pořízeny, velikosti jednotlivých datasetů, rozlišení fotografií, barevný model snímků a počet různých identit v datasetu. Následně je popsán zvolený způsob trénování modelu, způsob extrakce příznaků z fotografií a architektura samotné neuronové sítě. V další podkapitole je také popsáno uživatelské rozhraní vytvořené demonstrační aplikace.

V kapitole 4 jsou podrobněji popsány experimenty, které byly provedeny pro zlepšení přesnosti algoritmu a jejich vliv na chybovost algoritmu. Je zde také podrobněji popsán způsob testování a srovnání s ostatními algoritmy řešícími stejnou úlohu.

Kapitola 2

Shrnutí dosavadního stavu

Proces identifikace osob se skládá z několika dílčích úloh. V první řadě jsou obrazová data zaznamenána v digitální podobě videokamerou nebo fotoaparátem. Získané snímky pak mohou být zpracovávány přímo na koncovém zařízení (*edge device*) a výsledky analýzy odeslány na vzdálený server. Alternativním postupem je odeslání nezpracovaného obrazu rovnou na server, proces zpracování a analýzy obrazu pak proběhne až zde. Výhodou zpracování dat přímo na koncovém zařízení je především snížení požadavků na přenos dat, kdy objem zpracovaných informací je obecně výrazně menší, než samotný nezpracovaný záznam. Naopak zpracování obrazu na vzdáleném serveru přináší výhody v podobě menší limitace výpočetních zdrojů. Koncová zařízení (typicky Nvidia Jetson, Raspberry Pi a podobné) nemusí mít dostatečný výpočetní výkon pro zpracování obrazu. U systémů využívajících neuronové sítě je také často limitující velikost paměti grafického čipu.

Při samotné analýze obrazových dat může být snímek předzpracován. Často se jedná o normalizaci jasu fotografie nebo opravu zkreslení obrazu způsobeném čočkou kamery. Na předzpracovaných snímcích je následně provedena lokalizace hledaného objektu, v tomto případě lidského obličeje. Mezi nejznámější algoritmy na lokalizaci obličeje patří algoritmus Viola a Jones [52] a Histogram orientovaných gradientů (HOG) [5]. Oba algoritmy byly v dnešní době překonány systémy založenými na neuronových sítích, které dosahují výrazně lepší přesnosti. Stále však mohou najít uplatnění na zařízeních s omezeným výkonem. Po lokalizaci následuje extrakce příznaků, které identifikují danou osobu. Tyto příznaky jsou následně porovnány s dříve nasnímanými příznaky a dle podobnosti je určena identita. Jednotlivá stádia celého procesu a algoritmy pro ně využívané jsou podrobněji popsány v následujících podkapitolách.

2.1 Biometrie

Identifikace osoby spadá do oblasti biometrie. Jejím cílem je ustanovit vztah mezi osobou a její identitou [22]. Identifikace osoby obvykle probíhá jednou ze tří základních metod [22]:

- Podle informací, které osoba zná.
- Podle objektů, které vlastní.
- Podle vzhledu osoby.

Do první kategorie spadají osobní hesla, piny nebo kryptografické klíče. Do druhé kategorie se řadí identifikační karty, fyzické klíče, řidičské průkazy nebo i osobní telefon. Třetí oblast

využívá vzhledu nebo unikátního chování dané osoby. Tomuto typu identifikace se také říká biometrická identifikace.

Biometrická identifikace přináší některé výhody z hlediska bezpečnosti. Zatímco hesla mohou být vyzrazena nebo uhádnuta a fyzické identifikátory mohou být ztraceny nebo ukradeny, biometrické údaje jsou pevně spjaty s danou osobou. Díky tomu jsou biometrické systémy alespoň teoreticky méně náchylné na zneužití. Následující informace o biometrických systémech byly čerpány z [22].

Biometrický systém se stará o měření fyzických vlastností osoby, typicky se může jednat o otisky prstů, sken sítnice, záznam hlasu, ručně psaný podpis nebo charakteristiku chůze. Biometrický systém provádí dva typy úkonů. Prvním z nich je získání identifikačních příznaků jedince a jejich zapsání do databáze identit. Druhým úkonem je porovnání příznaků osoby s touto databází.

Biometrický systém se typicky skládá ze čtyř základních částí [22][21]. První z nich jsou samotné senzory, které zachycují identifikační data. Typicky to mohou být kamery snímající 2–D obraz obličeje nebo skener otisků prstů. Po sběru dat je typicky vyhodnocena jejich kvalita a dochází k odfiltrování šumu. Druhou součástí je pak modul na extrakci příznaků. Jelikož v databázi typicky nejsou uchovávána samotná surová data, je nutné z nich vyextrahovat identifikační příznaky. Extrakce příznaků je proces, kdy je ze vstupních dat vygenerována jejich kompaktní reprezentace [22]. Příznaky reprezentující jednotlivé osoby jsou pak skladovány v databázi, která slouží jako repozitář biometrických informací. S databází pak pracuje porovnávací modul, jehož cílem je výpočet podobnosti mezi identitou z databáze a vstupními daty ověřované osoby. Podobnost je obvykle popsána jako skóre, kde nižší hodnoty odpovídají vyšší podobnosti.

2.1.1 Verifikace a identifikace

V oblasti biometrie se běžně lze setkat se dvěma typy úloh. První z nich, tzv. **verifikace osoby**, je proces, kdy se systém snaží ověřit, zdali je daná osoba tím, za koho se vydává [21]. Jedná se tedy o specifický typ autentizace, kdy systém porovnává biologická identifikační data osoby s jedním záznamem z databáze.

Oproti tomu u **identifikace osoby** se systém snaží zjistit, která osoba z databáze je nejpodobnější zkoumané osobě [21]. Identifikace se dále dělí na pozitivní a negativní. U pozitivní identifikace se uživatel snaží provést svoji identifikaci, aniž by specifikoval svoji identitu [22]. Naopak u negativní identifikace se předpokládá, že se zkoumaná osoba buď aktivně nesnaží o identifikaci nebo se přímo snaží svoji identitu utajit [22]. Cílem negativní identifikace je pak předejít například dvojímu využití prostředků.

2.1.2 Chyby biometrických systémů

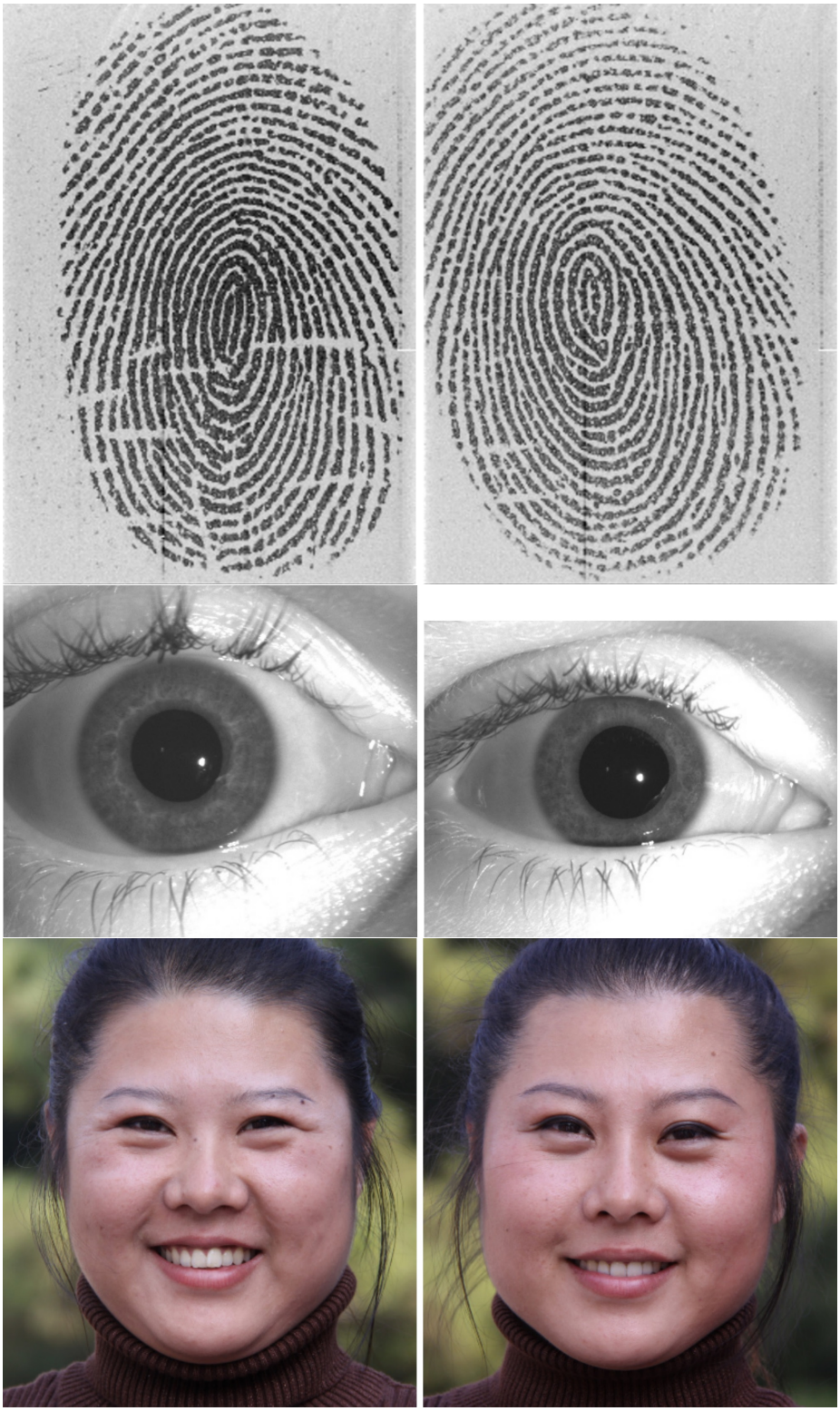
Při identifikaci osob podle jejich biometrických znaků je obvykle předpokladem, že daný znak je unikátní pro danou osobu a je dostatečně odlišný od zbytku populace. V této podkapitole jsou popsány situace, kdy tento vztah nemusí být splněn a biometrické systémy selžou v identifikaci osoby. Informace zde popsány byly čerpány především z [22].

U biometrických znaků osoby předpokládáme, že jsou neměnné v čase a že jsou skutečně unikátní pro jednu osobu. Výzkum v této oblasti však naznačuje, že biometrické znaky běžně používané k identifikaci tyto podmínky mnohdy nesplňují. Biometrické znaky osoby se mohou v průběhu času měnit, například vlivem stárnutí (především u dětí a adolescentů) nebo zraněním (poškození otisku prstu). Neměnnost dalších znaků v průběhu času, jako je sken zornice nebo obličeje, byla zkoumána minimálně.

Také jedinečnost běžně používaných identifikačních znaků je přinejmenším problematická. U identifikace podle obličeje jsou častým problémem jednovaječná dvojčata, jejichž obličeje mohou být pro tyto systémy nerozlišitelné [22]. Porovnání identifikačních znaků dvojčat je zobrazeno na obrázku 2.1. U identifikace podle obličeje hrají výraznou roli také rozdílné podmínky při pořízení snímku, typicky se může jednat o rozdílné světelné podmínky nebo odlišné natočení obličeje.

Kromě výše zmíněných podmínek se naměřené znaky člověka liší i v závislosti na psychickém stavu člověka [22], například při identifikaci podle stylu chůze lze očekávat výrazně rozdílný typy chůze pokud je člověk vystresovaný. Podobné změny budou zřejmě pozorovatelné i při rozpoznávání dle hlasu.

Výše zmíněné vlivy způsobují, že při měření biometrických znaků člověka očekáváme, že vzorky stejné osoby se budou lišit. Při procesu identifikace se pak zkoumá podobnost daných vzorků dat. Velmi podobné nebo identické vzorky naopak mohou značit snahu o podvrhnutí dat, například při znovupoužití předchozích záznamů. Tato skutečnost tvoří zásadní rozdíl od autentizace pomocí hesel nebo čipových karet, kde je očekávána bezchybná shoda.



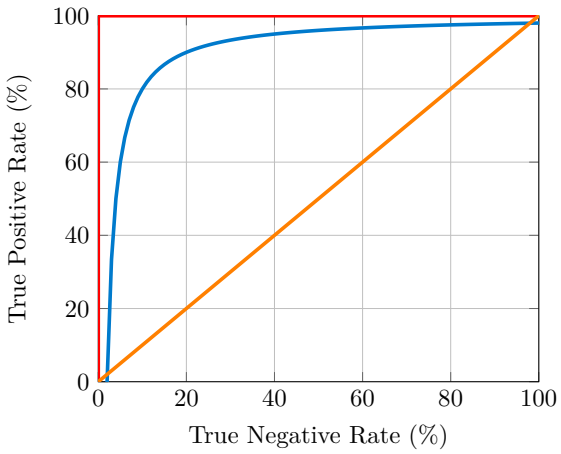
Obrázek 2.1: Porovnání otisků prstů, skenů sítnice a snímku obličejů u identických dvojčat. Rozlišení těchto vzorků je často obtížné pro člověka. Podobně náročné je jejich rozpoznání pro algoritmy strojového učení. Snímky převzaty z [22].

2.1.3 Metriky

Pro měření úspěšnosti algoritmů strojového učení je využíváno několika způsobů měření chybovosti. Základním a nejjednodušším způsobem je měření přesnosti binárních klasifikátorů (v rámci této práce například verifikačních systémů). Níže zmíněné informace byly čerpány především z [21] a z [22].

False Match Rate (FMR) označuje případy, kdy při verifikaci jsou vzorky dvou rozdílných osob systémem akceptovány jako náležící totožné identitě. Oproti tomu **False Non-Match Rate** (FNMR) značí případy, kdy jsou dva vzorky identifikačních znaků pocházejících od jedné osoby označeny za rozdílné. FMR a FNMR jsou někdy v literatuře označovány pod pojmy **False Accept Rate** (FAR) a **False Reject Rate** (FRR). Tyto metriky jsou obvykle uváděny v procentech, FNMR s pravděpodobností 5 % znamená, že systém v průměru vyhodnotí 5 ze 100 pokusů o autentizaci skutečným uživatelem jako rozdílné.

Vzájemný vztah mezi FMR a FNMR lze dále porovnat v závislosti na mezní hodnotě (*threshold*) pomocí křivky **Detection Error Tradeoff** (DET). Opakem tohoto grafu je **Receiver Operating Characteristic** (ROC). Tato křivka znázorňuje vztah mezi *true positive rate* a *true negative rate* v závislosti na hodnotě *threshold*. Ukázka křivky ROC je na obrázku 2.2. Pro zjednodušení této metriky je často namísto křivky vyjádřena přesnost systému pomocí jednoho čísla – **Area Under Curve** (AUC), tedy jako plocha pod křivkou.



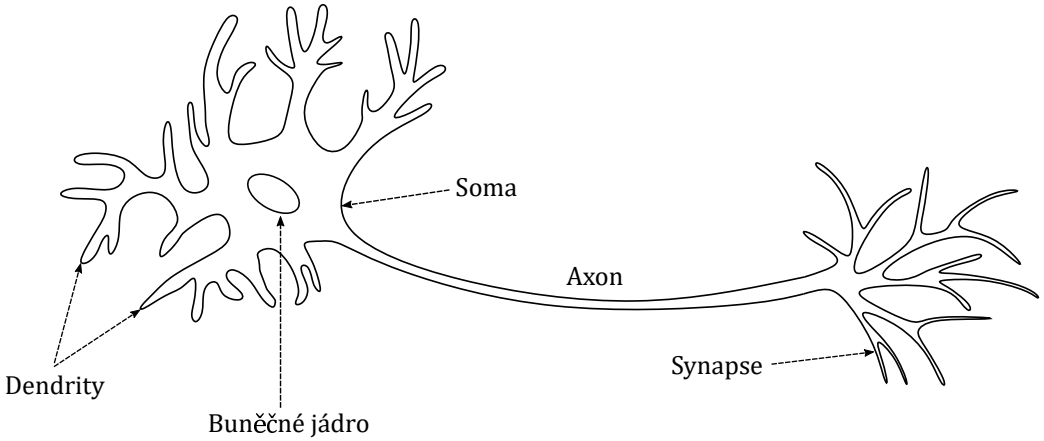
Obrázek 2.2: Ukázka možné křivky ROC (modře), popisující úspěšný systém, který vykazuje malý počet chyb a jeho křivka ROC se tak blíží levému hornímu rohu. Křivka ideálního systému je zobrazena červeně. Oranžovou barvou je pak zobrazena křivka nejhoršího možného systému (s AUC = 0,5). Takový systém není při binární klasifikaci s vyrovnanými třídami lepší, než náhodný systém (například hod mincí).

2.2 Neuronové sítě

Neuronové sítě jsou v současné době využívány pro široké spektrum úloh. Svoji roli mají i při procesu identifikace osob, kde se systémy na nich založené řadí mezi ty nejpřesnější. V této kapitole jsou popsány matematické základy a definice neuronových sítí.

Biologický neuron je základní jednotka nervové soustavy. Umělé neurony, které tvoří základ umělých neuronových sítí, jsou jím vzdáleně inspirovány. Informace o biologickém neuronu byly čerpány z [31].

Biologický neuron se skládá z buněčného těla (sóma) obsahující buněčné jádro. Ze sómy vybíhají dendrity se stromovou strukturou (vstupy neuronu). Na sómu je dále napojen výběžkovitý axon. Biologický neuron je znázorněn na obrázku 2.3. Většina neuronů má axon pouze jeden. Ten se však může dále větvit a být tak napojen na více cílových buněk. Axon je zakončený několika terminálními knoflíčky, které slouží jako výstup neuronu.

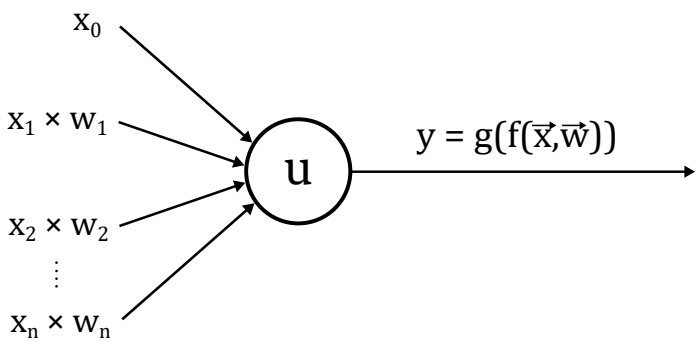


Obrázek 2.3: Biologický neuron.

Neurony lze dále klasifikovat podle jejich anatomie. Unipolární neurony nepřijímají signál od jiných neuronů, pouze samy tvoří vzruch na základě podráždění [1]. Typicky se může jednat o smyslové buňky, například tyčinky a čípky v sítnici. Bipolární neurony přijímají vzruch skrz jeden výběžek a signál propagují dále axonem [1]. Multipolární neurony pak přijímají vzruch větším počtem dendritů. Multipolární neurony jsou nejčastějším typem biologických neuronů [1].

Umělý neuron je výpočetní jednotka inspirována biologickým neuronem. Na vstupu umělého neuronu je jedna nebo více hodnot. Každá z těchto vstupních hodnot je vynásobena váhou. Vážené vstupy jsou následně sečteny a je k nim přičtena hodnota *bias*. Hodnoty váhového vektoru a hodnota bias nejsou pevně dané, jejich hodnota se mění při trénování. Před výstupem je na hodnotu v neuronu aplikována aktivační funkce. Obvykle je jako aktivační funkce použita nelineární funkce. Výpočet neuronu je poté ukončen, často je výsledek vyveden na vstupy neuronů v následující vrstvě. V případě, že se jedná o poslední vrstvu, je výstup neuronů této vrstvy chápán jako výstup celé sítě.

Celý výpočet umělého neuronu lze vyjádřit rovnicí $y = \sigma(\sum w_i x_i + b)$, kde σ značí aktivační funkci, vstupy x_0 až x_i jsou násobeny váhami w_0 až w_i a b značí bias. Umělý neuron je zakreslen na snímku 2.4.



Obrázek 2.4: Model umělého neuronu. Vektor s daty x_1 až x_n a bias x_0 vstupují do neuronu. Zde bázová funkce f vynásobí vstupní hodnoty s příslušnou váhou w_i a výsledky sečte. Následně je hodnota předána aktivační funkci g. Odtud jsou data předána další vrstvě neuronů.

2.2.1 Problémy řešené neuronovými sítěmi

V této sekci jsou podrobněji popsány typy úloh, pro jejichž řešení lze využít neuronové sítě. Úloha je v kontextu strojového učení chápána jako způsob, jakým by měl algoritmus zpracovat vzorek dat a transformovat data na výstup [12]. Tato data se skládají z příznaků (*features*), které lze chápat jako více abstraktní reprezentaci těchto dat. Výzkum neuronových sítí se stal v posledních letech velmi dynamickým odvětvím a v návaznosti na to přibývá i počet oblastí, pro které jsou využívány. Ty nejčastěji se vyskytující typy úloh jsou popsány v následujících podkapitolách.

Klasifikace

V klasifikačních problémech je účelem algoritmu specifikovat, do které z předem definovaných k kategorií spadá vstupní vzorek dat. Proces učení lze pak definovat jako snahu najít funkci $f : \mathbb{R}^n \to \{1, ..., k\}$ [12]. Existují i varianty klasifikačních úloh, kdy je výstupem vektor pravděpodobností, že vzorek dat spadá do dané třídy. V některých případech se také může jednat o situace, kdy vzorek může patřit do více než jedné kategorie nebo také varianta, kdy kategorie jsou organizovány ve stromové struktuře a podkategorie pak poskytují více specifické rozřazení.

Klasifikační úloha typická pro počítačové vidění je ověření přítomnosti objektu v obraze, kde vstupem sítě je snímek a výstupem je numerický kód, který identifikuje objekt na snímku. V posledních letech je patrný odklon od klasických algoritmů k algoritmům hlubokého učení, které dosahují výrazně lepších výsledků. Příkladem může být síť AlexNet [23], které jako první ukázala potenciál konvolučních neuronových sítí při řešení klasifikačních úloh.

Regrese

Úloha algoritmu pro regresní analýzu je, podobně jako u klasifikačních úloh, predikce výstupu na základě vstupního vektoru dat. Výstup algoritmu však na rozdíl od klasifikace není limitován předem definovanými třídami. Při trénování regresního algoritmus je cílem najít funkci $f : \mathbb{R}^n \to \mathbb{R}$ [12]. S regresními úlohami se lze setkat tam, kde lze očekávaný výsledek vyjádřit spojitou funkcí, obvykle se jedná o oblast robotiky a autonomie, analýzu počasí nebo finanční analýzu.

Přepis

Strojový přepis (*machine transcription*) je typ úlohy, kde algoritmus obdrží na vstupu nestrukturovaná data a jeho cílem je jejich transformace na diskrétní formu, obvykle do textové podoby [12]. V kontextu počítačového vidění je typickým příkladem OCR (*Optical Character Recognition*), tedy úloha, kdy je vstupem snímek textu a cílem algoritmu je jeho přepis do textové podoby. Jiné úlohy mohou vyžadovat přepis zvukového signálu na textový (*speech recognition*).

Překlad

Jako strojový překlad (*machine translation*) lze označit úlohy, kde je vstup algoritmu již strukturovaná sekvence symbolů jednoho jazyka a cílem algoritmu je překlad těchto dat do jiného jazyka [12]. Příkladem může být překlad textu z angličtiny do češtiny. Strojový překlad je obvykle příliš obtížný pro klasické algoritmy a využití strojového učení je tak mnohdy jediným způsobem, jak dosáhnout dobrých výsledků. Příkladem úspěšného řešení mohou být metody založené na rekurentních neuronových sítích, které dosahují relativně dobrých výsledků [4].

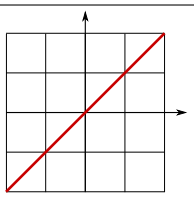
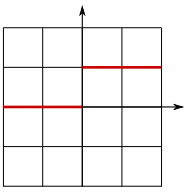
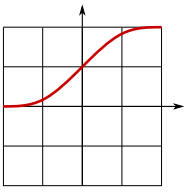
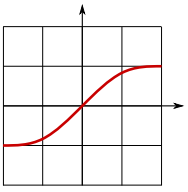
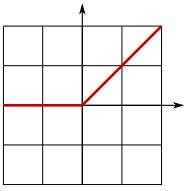
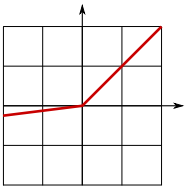
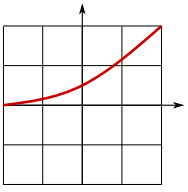
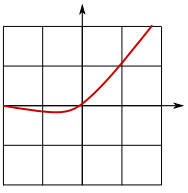
Lokalizace a detekce

Lokalizace je úloha z kategorie počítačového vidění, kde vstupem algoritmu je digitální snímek a úlohou algoritmu je nalézt pozici a velikost daného objektu. Výstup se tedy obvykle dá popsat čtyřmi čísly popisující pozici a šířku obdélníku, ve kterém se objekt nachází. Samotná lokalizace objektu není v kontextu počítačového vidění příliš používaná, obvykle se jedná pouze o první fázi algoritmu. V druhé fázi je na nalezený objekt použit klasifikační algoritmus, který určí, který objekt se v daném místě snímku nachází. Tento typ úloh, tedy kombinace lokalizace a klasifikace, se označuje jako detekce objektů (*object detection*).

2.2.2 Aktivační funkce

Aktivační funkce nebo také přechodová funkce (*transfer function*) je funkce určující výstup umělého neuronu. U vícevrstvých neuronových sítí je aktivační funkce většinou nelineární [3]. Důvodem je, že vícevrstvá neuronová síť s lineárními aktivačními funkcemi může být zjednodušena na jednovrstvou síť [3]. Pro využití potenciálu vícevrstvých sítí je tak nutné do sítě vnést nelinearitu. Z tohoto důvodu se často využívá sigmoida, která díky nelinearitě tímto problémem netrpí. Chování biologického neuronu je zároveň velmi podobné sigmoidě [55].

Často využívanou aktivační funkcí je Rectified Linear Unit (ReLU) a její varianty [55]. ReLU je definována jako $f(x) = max(0, x)$. Její výhodou oproti sigmoidě je možné zjednodušení výpočtu. ReLU je běžně modifikována, často používaná varianta *Leaky ReLU* modifikuje výpočet pro záporné hodnoty, kde záporným hodnotám přiřazuje malý pozitivní sklon. Touto modifikací lze eliminovat problém zvaný *dying ReLU problem*, kdy kvůli nulové derivaci funkce u záporných hodnot se může neuron dostat do stavu, kde nereaguje na trénování při využití gradientního sestupu [55]. Přehled nejpoužívanějších aktivačních funkcí, jejich předpisů a grafů je zachycen na obrázku 2.1.

Funkce:	**Předpis:**	**Graf:**
Identita	$f(x) = x$	
Binární prahová funkce	$f(x) = \begin{cases} 0 \text{ pro } x < 0 \\ 1 \text{ pro } x \geq 0 \end{cases}$	
Sigmoida	$f(x) = \frac{1}{1+e^{-x}}$	
Hyperbolický tangent	$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$	
ReLU	$f(x) = \begin{cases} 0 \text{ pro } x < 0 \\ x \text{ pro } x \geq 0 \end{cases}$	
Leaky ReLU	$f(x) = \begin{cases} -ax \text{ pro } x < 0 \\ x \text{ pro } x \geq 0 \end{cases}$	
SoftPlus	$f(x) = log_e(1 + e^x)$	
H–Swish	$f(x) = \frac{x}{1+e^{-x}}$	

Tabulka 2.1: Přehled aktivačních funkcí a jejich grafů.

2.2.3 Trénování neuronových sítí

Trénování neuronové sítě je proces, kdy se síť snaží nalézt co nejoptimálnější řešení problému bez toho, aby byla na daný úkol naprogramována. Toho dosáhne tak, že modifikuje hodnoty váhových vektorů tak, aby síť dosahovala při výpočtu co nejmenší chyby. Chyba sítě je vypočtena pomocí chybové funkce, tato hodnota bývá označována pod několika názvy: chybová funkce (*error*), ztrátová funkce (*loss*) nebo nákladová funkce (*cost*) [57].

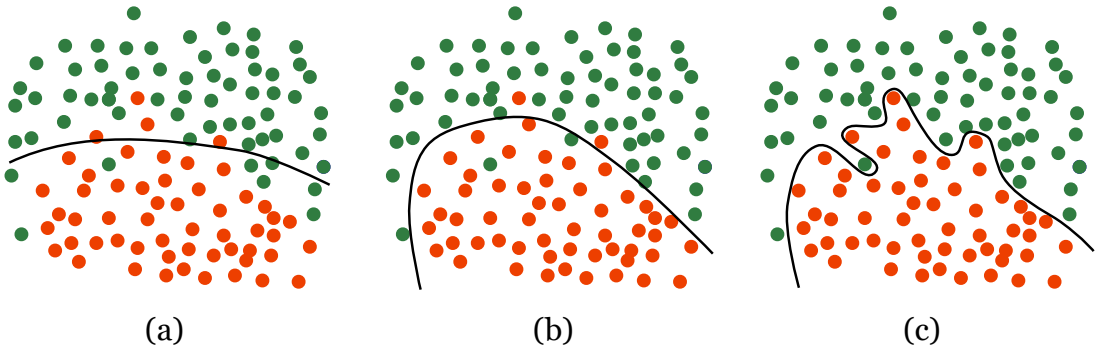
Původní algoritmus backpropagation využíval jako chybovou funkci součet kvadratických chyb dělený velikostí výstupního vektoru (*Mean Square Error – MSE*). Funkce MSE je tedy definována jako

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(Y_i - \hat{Y}_i)^2, \tag{2.1}$$

kde n je velikost výstupního vektoru, Y_i je očekávaný výstupní vektor (*label*) a $\hat{Y}_i$ je skutečný výstupní vektor sítě.

Chybové funkce počítají rozdíl mezi ideálním a skutečným výstupem sítě. Tento způsob učení je označován jako učení s učitelem (*supervised learning*) a je pro něj tedy nutné mít kromě vstupních dat také očekávané výstupní hodnoty (*ground–truth labels*). Cílem trénování neuronové sítě je pak nalezení vztahu mezi těmito výstupními hodnotami a hodnotami na vstupu sítě.

Při učení s datasetem s omezenou velikostí nastanou postupně tři fáze. Nejprve je síť tzv. podtrénovaná. Jedná se o stav, kdy ještě nebyl vyčerpán potenciál datasetu a pokračující učení vede ke zlepšení přesnosti jak u trénovacího, tak i validačního datasetu. Po následném trénování přijde bod zlomu, kdy chyba na validačním datasetu začne opět stoupat, zatímco na trénovacím datasetu bude klesat. V této fázi je vhodné trénování přerušit, jelikož jeho pokračování již přesnost sítě nezlepší. Pokud by k tomu nedošlo, síť se dostane do přetrénovaného stavu, kdy se přeučila na konkrétní data z trénovací sady za cenu horší přesnosti na nových datech. Ukázka průběhu trénování je zobrazena na obrázku 2.5.

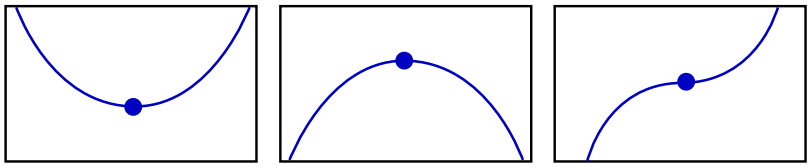


Obrázek 2.5: Vizualizace průběhu trénování. Na obrázku (a) je zobrazení predikce podtrénované sítě, na snímku (b) je predikce správně natrénované sítě a na snímku (c) je ukázka přetrénované sítě. Přetrénovaná síť je sice schopna správně klasifikovat vzorky z datasetu na kterých byla trénována, přetrénováním však přišla o schopnost správně pracovat s neznámými daty.

2.2.4 Gradientní sestup

Většina algoritmů hlubokého učení zahrnuje snahu o nalezení optimálního řešení dané úlohy. Proces optimalizace jak pak definován jako hledání minima nebo maxima určité funkce $f(x)$ za pomocí modifikace proměnné x [12]. Většina optimalizačních problémů z oblasti strojového učení bývá definována jako minimalizace funkce $f(x)$, maximalizace může být definována jako $-f(x)$ [12]. U neuronových sítí je touto funkcí chybová funkce a algoritmus pro její minimalizaci se nazývá gradientní sestup (*gradient descent*).

Při hledání minima chybové funkce je využívána derivace této funkce $f'(x)$ [12]. Derivace udává informaci o sklonu tangenty v bodě x. Pokud je v daném bodě $f'(x) = 0$ pak tento bod nazýváme extrémem funkce. Z hlediska optimalizačních úloh jsou pro nás zajímavá lokální minima funkce, tedy ty extrémy funkce, kde je $f(x)$ nižší než, ve všech okolních bodech a není tak již možné snížit hodnotu $f(x)$ posunem o nekonečně malý krok [12]. Naopak lokální maxima jsou body, kde je $f(x)$ vyšší, než všechny okolní body. Speciálním případem jsou tzv. sedlové body, tedy body, kde $f'(x) = 0$, ale sousední body jsou výše i níže, než v tomto bodě. Tyto tři případy jsou znázorněny na obrázku 2.6.



Obrázek 2.6: Vizualizace lokálních extrémů funkce jedné proměnné. Ve všech případech je v daném bodě $f(x) = 0$. Pokud je $f(x)$ v tomto bodě nižší, než v okolí bodu, jedná se o lokální minimum (a). Pokud je $f(x)$ vyšší, než body v okolí, jedná se o lokální maximum (b). Speciálním případem je sedlový bod (c), který má v okolí jak vyšší, tak nižší body a nejedná se tedy o maximum funkce [12].

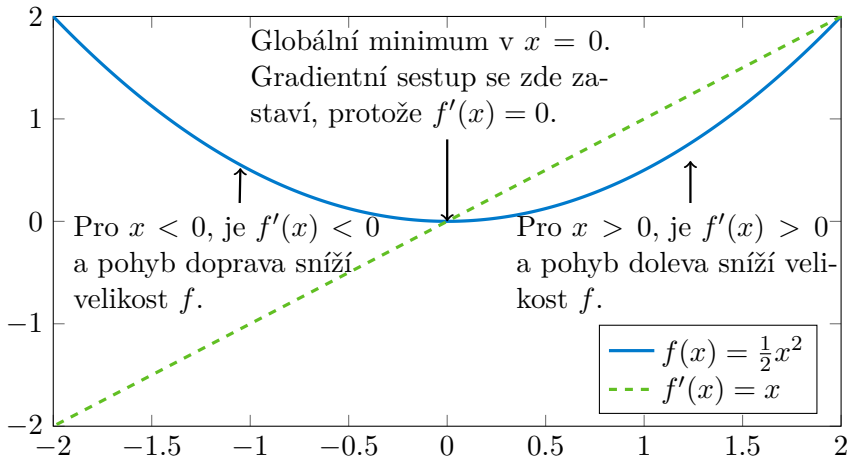
Bod, který má pro funkci f absolutně nejnižší hodnotu, se nazývá globálním minimum funkce. Každá funkce má pouze jedno globální minimum, popřípadě více globálních minim, ve kterých jsou si hodnoty $f(x)$ rovny. V kontextu strojového učení je cílem algoritmu minimalizace funkce s vícedimenzionálním vstupem. Hledání globálního minima je tak obtížné, jelikož algoritmy pro optimalizaci obvykle nezaručují nalezení globálního minima. Obvykle je jako řešení přijato lokální minimum, ve kterém je $f(x)$ dostatečně nízké, ačkoliv se nejedná o globální minimum.

U funkcí se vstupem s více dimenzemi: $f : \mathbb{R}^n \rightarrow \mathbb{R}$ se pro optimalizaci využívají parciální derivace. Parciální derivace funkce o více proměnných je její derivace vzhledem k jedné z těchto proměnných, přičemž ostatní proměnné jsou brány jako konstanty [12]. Gradient funkce f je pak vektor, který obsahuje parciální derivace vzhledem ke každé proměnné funkce f. Prvek i gradientu $\nabla_x f(x)$ je tedy parciální derivace f vzhledem k x_i [12].

V kontextu neuronových sítí je pak při učení využíván gradient chybové funkce ∇E. Chybu lze minimalizovat metodou gradientního sestupu tak, že jsou upraveny hodnoty vah neuronů podle pravidla:

$$\Delta \overrightarrow{w} = -\mu \nabla E, \tag{2.2}$$

kde $\overrightarrow{w}$ je vektor vah a $\Delta\overrightarrow{w}$ je posun hodnot tohoto vektoru, ∇E je gradient chybové funkce a μ je velikost kroku. Učení neuronové sítě tedy znamená změna prvků váhového vektoru ve směru proti gradientu chybové funkce. Vizualizace gradientního sestupu je znázorněna na obrázku 2.7.



Obrázek 2.7: Algoritmus gradientní sestup při hledání minima funkce $f(x) = \frac{1}{2}x^2$. Převzato z [45].

Algoritmus gradientního sestupu v původní podobě určuje velikost chyby vzhledem k celému datasetu. Tento postup má však za následek zvýšené časové nároky na trénování. V dnešní době datasety běžně obsahují stovky tisíc vzorků (viz dataset CelebA 3.1.2) a pro každou jednu iteraci algoritmu by bylo nutné projít celý dataset. Pro zrychlení trénování bývá využíván fakt, že gradient menšího vzorku dat bude dostatečně podobný gradientu celého datasetu a lze tak vždy zpracovat pouze menší část datasetu. V praxi to může vypadat například tak, že při datasetu obsahujícím 100 000 snímků bude v jedné dávce zpracováno pouze 32 a posun vah bude v jednom kroku algoritmu určen pouze z nich. Tato metoda učení se nazývá stochaistický gradientní sestup (*stochaistic gradient descent – SGD*)

Učení s učitelem

Učení s učitelem je proces, kdy jsou váhové vektory neuronů modifikovány tak, aby se minimalizovala chybovost výstupu sítě. Učitelem jsou v tomto kontextu atributy datasetu (*dataset labels*), které síti dodají správné řešení pro daný vstup. Nejpoužívanějším algoritmem pro učení s učitelem je algoritmus zpětného šíření chyby (*backward propagation of error – backpropagation*).

Algoritmus backpropagation byl poprvé popsán na začátku 60. let [51], nicméně širší využití tohoto algoritmu bylo limitováno využitím perceptronů se skokovými aktivačními funkcemi. Rozmach nastal až s koncem 80. let s využitím diferencovatelných funkcí [31]. Od této doby je hojně využíván při učení s učitelem.

Každá iterace algoritmu zpětného šíření chyby projde třemi fázemi. V **dopředné fázi** (*feed–forward phase*) je pro každou dvojici vstupu a výstupu $(\overrightarrow{x_d}, \overrightarrow{y_d})$ vstupní vektor $\overrightarrow{x_d}$ umístěn na vstup sítě a výpočet je postupně proveden pro každý neuron j ve vrstvě k [30]. Výpočet probíhá od vstupní vrstvy směrem k výstupním vrstvě m.

Ve **zpětné fázi** (*backward phase*) je nejprve vypočítána chyba δ_1^m pro výstupní vrstvu. Chyba je následně přenášena zpět do skrytých vrstev δ_j^k směrem od poslední skryté vrstvy ($k = m - 1$). Při přenosu chyby je vypočítána parciální derivace individuální chyby E_d vzhledem k váze w_{ij}^k [30]. Na konci zpětné fáze je vypočítán celkový gradient pro dvojici $(\overrightarrow{x_d}, \overrightarrow{y_d})$ jako průměr jednotlivých gradientů [30].

V poslední fázi jsou **upraveny váhy** jednotlivých neuronů. Váhy jsou posunuty proti směru gradientu vzhledem k velikosti učícího kroku μ [30]. Tento algoritmus je iterativně opakován, ukončen může být několika způsoby. V praxi to běžně bývá po dosažení určitého počtu opakování, dosažení dostatečně malé chyby nebo po vypršení časového limitu.

Jakmile je proces trénování dokončen, jsou váhy zmraženy (*frozen*) a síť lze využít predikci u nových vstupních dat. Je tedy pouze proveden výpočet sítě odpovídající dopředné fázi.

Ačkoliv je algoritmus zpětného šíření chyby zdaleka nejpoužívanějším a nejrozšířenějším algoritmem pro trénování neuronových sítí [53], není zdaleka jediným algoritmem vhodným pro tento účel. Trénování sítě lze dosáhnout pomocí řady jiných metod, například pomocí evolučních algoritmů, kdy váhové vektory neuronů považujeme za reprezentaci jedince a pomocí rekombinace nejúspěšnějších jedinců populace a náhodné mutace dosáhneme učení sítě. Využití alternativních algoritmů pro učení neuronových sítí je stále předmětem výzkumu.

2.2.5 Vrstvy neuronových sítí

Většina praktických problémů je příliš složitá pro vyjádření pouze jedním neuronem a je tak zapotřebí využít architektury sítí s velkým množstvím uzlů. Způsob, jakým jsou jednotlivé neurony propojeny, pak určuje, jak bude probíhat výpočet celé sítě. Správná volba architektury sítě je tak důležitým rozhodnutím pro vývojáře.

Nejjednodušším způsobem propojení neuronů je plné propojení, tedy stav, kdy výstup každého neuronu je napojen na vstup všech neuronů (včetně sebe sama). Sítě s touto architekturou se nazývají plně propojené sítě (*fully–connected networks*). Ačkoliv je tato architektura sítě velmi jednoduchá, v praxi je její využití minimální z důvodu vysokého počtu parametrů pro učení [31]. Počet parametrů roste kvadratickou rychlostí s počtem neuronů, síť s n neurony pak obsahuje n^2 vah. Tato skutečnost limituje praktické využití plně propojených sítí, protože neexistuje dostatečně rychlý způsob trénování pro větší sítě tohoto typu.

U problematiky počítačového vidění se nejčastěji setkáváme se sítěmi, které se řadí do kategorie konvolučních sítí (*convolutional neural networks – CNN*). Konvoluční neuronové sítě nacházejí využití především v situacích, kde vstupní data mají mřížovou topologii [12]. Příkladem mohou být data, která jsou vzorkované v určitých časových intervalech (1–D mřížka) nebo obrazová data, kde se jedná o 2–D mřížku jednotlivých pixelů. U specializovaných zařízení se lze setkat také s 3–D volumetrickými daty, například z výpočetní tomografie (*CT scan*).

Jako konvoluční síť lze označit ty neuronové sítě, které využívají konvolučních vrstev pro extrakci informací z dat. Typicky se jedná o data, u kterých existuje závislost sousedních vzorků dat (např. pixely fotografie). Kromě konvolučních vrstev se v nich objevují především plně propojené vrstvy a seskupující (*pooling*) vrstvy. Tyto tři typu vrstev tvoří základní stavební bloky konvolučních neuronových sítí a jsou podrobněji popsány níže.

Plně propojená vrstva

V dnešní době převládající architektura dopředných sítí rozděluje neurony do vrstev. Neurony náležící stejné vrstvě nejsou vzájemně propojeny. Pro neurony v každé vrstvě platí, že na jejich vstup jsou přivedeny výstupy z neuronů z předchozí vrstvy a podobně výstup je předán dál do následující vrstvy. Výjimkou jsou vstupy do neuronů první vrstvy, kam jsou přivedena data specifická pro danou úlohu a výstupy neuronů poslední vrstvy, které jsou interpretovány jako výstup celé sítě. Vrstvy propojené tímto způsobem jsou nazývány plně propojené (*fully connected layer*) a jsou nejjednodušším příkladem vrstev neuronových sítí. V praxi se dnes využívají architektury s kombinací většího množství typů vrstev a plně propojené vrstvy se často nacházejí v posledních vrstvách sítě, kde slouží jako klasifikátor výstupu [24][40].

Ačkoliv jsou plně propojené vrstvy výrazně úspornější z hlediska počtu parametrů než plně propojené sítě, sítě obvykle nemají více než čtyři tyto vrstvy [31]. Pro úlohy se vstupními daty s většími rozměry jsou dopředné sítě skládající se z plně propojených vrstev stále z limitovány vysokým počtem parametrů, což způsobuje komplikace při trénovaní na dnešním hardware.

Konvoluční vrstva

Konvoluce je operace dána vztahem:

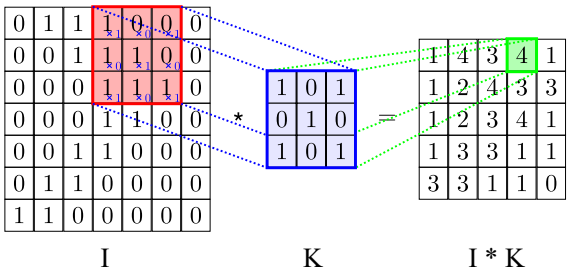
$$I(u,v) = \sum_{i=-a}^{b} \sum_{j=-c}^{d} h(i,j) I(u+i, v+j), \tag{2.3}$$

což v kontextu zpracování obrazu znamená, že nová hodnota pixelu na pozici u, v bude záviset na jeho okolí a hodnotách masky [57]. Proměnné a, b, c, d, popisují rozměry masky, které jsou dané vztahem $(a + b + 1) \times (b + c + 1)$.

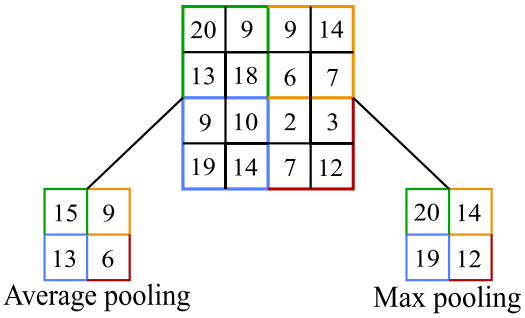
Dvourozměrnou konvoluci si lze představit jako posouvání 2–D okna (často označovaného jako *filtr*, *maska* nebo *jádro* konvoluce), které se posouvá po obraze o určitý krok (počet pixelů). Na každém kroku iterace je pak vypočtena nová hodnota pixelu tak, že data vstupního obrazu jsou vynásobena hodnotami jádra a z těch je pak vypočtena průměrná hodnota. Operace konvoluce je znázorněna na obrázku 2.8.

Jelikož je při výpočtu nových hodnot pixelů v konvolučních vrstvách bráno v potaz i okolí daného pixelu, je výpočet komplikovaný v okrajových částech obrazu. Například při použití jádra o rozměrech 7×7 nejsou všechny okrajové hodnoty definovány pro tři řady pixelů na každé straně snímku. Tento problém lze řešit několik způsoby, chybějící data lze doplnit, například průměrnou hodnotou ostatních pixelů nebo nulovou hodnotou. Druhým způsobem řešení je vynechání výpočtu u okraje matice a s tím spojené zmenšení výstupu. Při použití filtru s rozměry 7×7 by tak výstup vrstvy byl o 6 pixelů menší horizontálně i vertikálně.

Ke zmenšení výstupu dochází také v případech, kdy je velikost kroku větší než jedna. V těchto případech je konvoluční jádro posouváno po vstupu po větších částech. U konvoluce s krokem o velikosti dva pixely tak dojde ke zmenšení výstupu na poloviny vstupu. Zmenšení výstupu tímto způsobem sice může vést ke ztrátě detailních informací, nicméně výpočet sítě se tím dále zrychluje, protože další vrstvy pak také pracují s daty o menších rozměrech.



Obrázek 2.8: Vizualizace konvoluce. Jádro K je posouváno po vstupu I a jejich hodnoty pixelů na stejných pozicích jsou vzájemně vynásobeny. Výsledný pixel je pak vyhodnocen jako průměrná hodnota z těchto hodnot. Snímek převzat[2].



Obrázek 2.9: Ukázka operací *Average pooling* a *Max Pooling*. Hodnoty pixelů v okolí jsou agregovány do jedné. Při okolí o velikosti 2×2 a kroku o velikosti dva pixely má výstup vrstvy poloviční rozměry. Převzato z [45].

Tradiční vrstvy neuronových sítí při výpočtu vynásobí matici vstupních dat s maticí vah, každá jednotka vstupních dat je tedy použita pro výpočet v každém neuronu. Jelikož u konvolučních vrstev je jádro konvoluce obvykle menší než rozměry vstupních dat, je vztah mezi vstupem a výstupem označován jako řídká interakce (*sparse interaction* nebo také *sparse connectivity*) a je výrazně výpočetně i paměťově jednodušší [12]. U konvolučních vrstev totiž nejsou hledanými parametry váhy neuronů, ale hodnoty matice jádra konvoluce, kde díky menším rozměrům dochází k výrazné výpočetní úspoře. Zatímco u plně propojených vrstev se vstupem o rozměru m a výstupu s rozměrem n je při vynásobení matic výpočetní složitost $O(m \times n)$ [12]. U řídké interakce lze výstup limitovat na k a výpočetní složitost je pak $O(k \times n)$ [12]. Z praxe se ukazuje, že lze zachovat dobrou přesnost s parametrem k o několik řádů nižším. Podrobněji je tento princip znázorněn na obrázku 2.10.

U současných rozsáhlých neuronových sítí je běžné, že každá konvoluční vrstva obsahuje více než jeden filtr. Toto umožňuje síti pomocí jednotlivých filtrů detekovat rozdílné příznaky. Vhodný počet filtrů není pevně stanoven, moderní sítě obvykle používají vyšší desítky až stovky filtrů na vrstvu. Často se také počet filtrů zvyšuje směrem k výstupním vrstvám (například síť VGG obsahuje postupně 64, 128, 256 a 512 filtrů v konvolučních vrstvách).

Pooling vrstva

Seskupující vrstva (*pooling layer*) transformuje vstupní data z několika pixelů do jednoho. Způsob této transformace je definován funkcí, nejběžněji je využívána funkce *max pooling*, která z okolních pixelů (obvykle 2×2) vybere nejvyšší hodnotu [60] nebo operace *average pooling*, která z okolí spočítá průměr. Seskupující vrstva nemá žádné parametry pro trénování. Celá operace je pevně definována při specifikaci architektury sítě a je po celou dobu trénování neměnná.

Cílem seskupujících vrstev je učinit síť více robustní vůči malým invariantám na vstupu. Toto je výhodné, jelikož pro správné fungování sítě většinou není podstatná přesná poloha určitého příznaku, ale spíše jeho přítomnost v obraze a jeho relativní pozice vůči ostatním příznakům [12].

[2] 2D Convolution: https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution

Obrázek 2.10: Zobrazení detekce hran pomocí konvoluce. Hodnoty pixelů prostředního snímku byly vypočítány jako rozdíl původní hodnoty a hodnoty sousedního levého pixelu. Pro snímek napravo pak jako rozdíl původní hodnoty pixelu a sousedního horního pixelu. Za povšimnutí stojí zvýrazněné vertikální linie v prostředním obrázku a horizontální linie v pravém obrázku. Tohoto výsledku lze pomocí operace konvoluce dosáhnout efektivním způsobem. Vstupní snímek má rozměry 300×400 pixelů. Výpočtu tak lze dosáhnout za pomocí konvoluce s jádrem se dvěma prvky, za pomocí celkem 360 000 operací ($300 \times 400 \times 3$, tedy pro každý pixel vstupu dvě operace násobení a jedno sčítání [12]). Při řešení podobné operace pomocí plně propojených vrstev by došlo k násobení matic vstupu a vah, tedy $300 \times 400 \times 300 \times 400$, tedy 14,4 bilionů operací. Výpočet konvolučních vrstev je tedy o mnoho řádů jednodušší a jejich použití počátečních vrstvách sítě je téměř nutným předpokladem pro efektivnost sítí s rozsáhlejší architekturou [12].

Podobně jako u konvolučních vrstev, seskupující vrstva má za následek zmenšení obrazu, což vede ke zrychlení výpočtu v následujících vrstvách. Pro mnoho úloh je podvzorkování užitečné, jelikož klasifikační část sítě, která se obvykle nachází v pozdních vrstvách sítě, bývá sestavena z plně propojených vrstev, jejich výpočet by byl na datech s původními rozměry značně pomalý. Pomocí úprav parametrů seskupujících vrstev lze dosáhnout toho, že data jsou postupně zmenšena na rozměry vyžadované klasifikačními vrstvami sítě [12]. Ukázka operace max–pooling a average–pooling je zobrazena na obrázku 2.8.

2.2.6 Inicializace vah

Inicializace vah je proces, kdy pro nově vytvořený model sítě jsou určeny výchozí hodnoty váhových vektorů. Správná inicializace vah je jedním z klíčových faktorů pro rychlé trénování sítě a její konvergenci [42][32]. Výběr vhodného způsobu inicializace vah je závislý na aktivační funkci v dané vrstvě.

Často využívanou metodou pro inicializaci vah je nastavení hodnot podle uniformního rozdělení $N(0, v^2)$ [26]. Volba vhodného parametru v^2 byla poprvé zkoumána v roce 2010 [11]. V rámci tohoto výzkumu byla optimální hodnota pro lineární aktivační funkci stanovena na $v^2 = 1/N$, kde N udává počet neuronů v předchozí vrstvě. Tato metoda inicializace vah byla později adoptována i na jiné aktivační funkce. Časem se pro ni ustálil název **Xavierova inicializace**, podle jednoho z jejich autorů – Xavier Glorot.

Na tento výzkum navázal He a spol., [15] kteří argumentují, že Xavierova inicializace není vhodná pro často používanou aktivační funkci ReLU. Namísto toho navrhli inicializaci pomocí vztahu $v^2 = 2/N$. Tento způsob inicializace je běžně uváděn pod jménem **He inicializace**. Autoři ve svém článku demonstrovali výhodu této inicializace natrénováním sítě s 30 vrstvami, kterou nebylo možné natrénovat s Xavierovou inicializací. He inicializace navíc podává dobré výsledky s funkcí ReLU.

2.3 Metody počítačového vidění

Historie konvolučních neuronových sítí sahá do 80. let dvacátého století. Síť **Neocognitron** [7] byla prvním příkladem nasazení neuronových sítí pro počítačové vidění. První verze sítě byla schopna rozpoznávat ručně psané číslice, pozdější verze (1990) pak i k rozpoznávání písmen.

Za první moderní architekturu lze však považovat až síť **LeNet** [28]. Síť byla schopná rozpoznávat ručně psané alfanumerické znaky o velikost 32×32 pixelů. Architektura sítě byla relativně podobná dnešním sítím, nejprve konvoluční vrstvy prokládané pooling vrstvami vyextrahovaly příznaky, následně se dvě plně propojené vrstvy staraly o klasifikaci.

Ačkoliv tyto neuronové sítě vykazovaly slibné výsledky, výzkum v následující dekádě byl spíše sporadický. Obnovení zájmu nastalo až s uvedením architektury **AlexNet** [23]. Tato síť se skládá z celkem 8 vrstev s trénovatelnými parametry. Prvních pět jsou konvoluční vrstvy a zbylé tři jsou plně propojené vrstvy. Neurony využívají jako aktivační funkci ReLU. Poslední vrstva provádí klasifikaci pomocí funkce softmax (do 1000 tříd specifikovaných datasetem ImageNet). Mezi konvolučními vrstvami se dále nachází max–pooling vrstvy a normalizační vrstvy.

Síť byla trénována dnes běžnou technikou zpětného šíření chyby. Trénování této poměrně rozsáhlé sítě (60M vah, 612M spojení neuronů) bylo možné díky pokroku ve vývoji hardware, síť byla trénována paralelně na dvou GPU. Na datasetu ImageNet dosáhla tato síť chybovosti 16,4 %, čímž výrazně překonal ostatní metody. Druhá nejúspěšnější metoda SIFT + FVs [36] dosáhla ve stejném roce chybovosti 25,7 %.

V následujících letech došlo k postupných vylepšením této architektury. Model **ZFNet** [58] přidal další konvoluční vrstvy a snížil velikost konvolučních filtrů. ZFNet dosáhla chybovosti 14,7 % na stejném datasetu. Podobnou lineární architekturu využívá také model **VGG** [41]. Autoři zde využívají větší hloubky sítě (celkem 19 vrstev) pro zpřesnění výsledků. Došlo také k zefektivnění sítě použitím menších filtrů. Autoři argumentují, že použitím dvou po sobě následujících konvolučních vrstev s filtry o velikosti 3×3 lze dosáhnout podobného výsledku, jaké by přineslo použití jedné vrstvy s 7×7 filtry. Zároveň ale dojde k výraznému snížení počtu parametrů. Síť VGG dosáhla přesnosti 6,8 % (ImageNet12).

V roce 2014 byla také představena architektura **GoogleNet** (známá pod pojmem Inception) [47]. První vrstvy tohoto modelu připomínají starší modely se sérií konvolučních a pooling vrstev. Od šesté vrstvy se však architektura modelu výrazně liší, namísto lineárně propojených vrstev se skládá ze série modulů *inception*. Každý z těchto modulů obsahuje několik paralelně uspořádaných konvolučních vrstev, které se liší velikostmi filtrů. Jejich výstup je poté konkatenován a výsledek je předán dalším modulu. Moduly inception jsou lineárně poskládány za sebou. V posledních vrstvách se nachází ještě plně propojená vrstva starající se o klasifikaci výstupu. Před samotným vstupem dat do prvního modulu inception je také provedena série konvolucí.

Vývoj těchto sítí poukazuje na fakt, že přesnost modelu je závislá na hloubce sítě a počtu konvolučních vrstev. S hlubšími modely však dochází k potížím s trénováním. Autoři architektury **ResNet** [14] řeší tento problém tak, že přidávají propojení vrstev, které vynechává vždy dvě konvoluční vrstvy a posílá signál do pozdějších vrstev. Tento signál je pak sečten s výstupem přeskočených konvolučních vrstev. Toto zaručuje, že se s vysokým počtem vrstev vstupní informace neztrácí a je možné ji dále propagovat. Autoři ve své práci navrhli větší množství sítí s počtem vrstev od 31 až po 152 (ResNet–34, ResNet–50, ResNet–101, ResNet–152).

Idea residuálních spojení byla také využita pro vylepšení některých existujících architektur. Při kombinaci s architekturou inception vzniklo několik sítí – tzv. **Inception-Resnet** sítě. Výzkum [46] poukazuje na fakt, že přidáním residuálních spojení do existující architektury došlo k mírnému zlepšení přesnosti.

Na podobném principu je postavena i síť **DenseNet** [18] z roku 2018. Tento model se skládá ze série hustě propojených bloků (*dense block*). Každý z těchto bloků obsahuje vrstvy, kde je výstup každé vrstvy přiveden na vstup všech následujících vrstev. Na rozdíl od tradičních sítí, které mají L spojení mezi L vrstvami, DenseNet obsahuje $\frac{L(L+1)}{2}$ spojení. Na rozdíl od sítí z rodiny ResNet nejsou hodnoty vstupů do jednotlivých vrstev sečteny, ale jsou konkatenovány.

Paralelně s výzkumem DenseNet vzbudily velký zájem výzkumníků i architektury tvz. **kapsulových neuronových sítí** (*Capsule networks*). Sítě využívající kapsulí byly poprvné představeny v roce 2017 týmem vedeným Geoffrey Hintonem [34]. Ti ve své práci argumentovali, že běžné používané konvoluční sítě kladou malý důraz na vzájemnou pozici jednotlivých příznaků v obraze a je tak jednoduché síť zmást, pokud snímek obsahuje všechny očekávané části objektu, ale jejich pozice byly pozměněny. Hinton a spol. poukazují na fakt, že při výpočtu běžné konvoluční neuronové sítě je kladen důraz na invariantnost vůči různým transformacím objektu, čehož je dosaženo především pomocí konvolučních a pooling vrstev, což nicméně vede k výše zmíněnému problému.

Jako řešení byla navržena architektura využívající kapsule. Kapsule, na rozdíl od běžných neuronů jejichž výstupem je skalár, provedou výpočet nad jejich vstupem a výsledek následně zabalí do malého výstupního vektoru. Další změna je u aktivační funkce. Zatímco u běžných konvolučních sítí je nelinearita zajištěna např. funkcí ReLU, u kapsulí je toto zajištěno funkcí *squash* aplikovanou na výstupní vektor:

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|}, \tag{2.4}$$

kde s_j je výstup z předchozího kroku a v_j je výstup aktivace (vektor o velikosti 1).

Výše uvedené sítě sice dosahují *state–of–the–art* přesnosti, nicméně při jejich návrhu je na ně pohlíženo téměř výhradně pouze z tohoto hlediska. Část výzkumu se naopak snaží najít optimální rovnováhu mezi velikostí modelu a přesností. Snahou je nalézt modely, které jsou schopné běžet s výrazně menšími nároky na hardware a které zároveň minimálně kompromitují přesnost. Nejznámějším zástupcem tohoto přístupu jsou sítě z rodiny **MobileNet** [17]. Tento typ architektury z roku 2017 se zaměřuje na jednoduché snížení počtu parametrů sítě pomocí dvou hyperparametrů: parametr šířky α a parametr rozlišení ρ. Modifikací těchto dvou hyperparametrů lze efektivně zmenšit počet parametrů sítě podle účelu. Sítě MobileNet lze použít na široké spektrum úloh, typicky pro klasifikaci, lokalizaci a segmentaci. Samotná architektura dosáhla dalších úspor pomocí tzv. hloubkově oddělitelné konvoluce *(depthwise separable convolution)*. Ta oproti běžné konvoluci přináší výrazně jednodušší výpočet za cenu malé ztráty přesnosti.

Architektura byla dále vylepšena v několika následujících iteracích. V architektuře MobileNetV2 [37] byly do modelu přidány tzv. invertované residuální vrstvy a lineární bottleneck bloky. Poslední verze, tzv. MobileNetV3 [16] z roku 2019, je architektura získaná pomocí algoritmu na vyhledávání optimální struktury sítě. Autoři poskytli dvě varianty, MobileNetV3Large a MobileNetV3Small. Jejich použití se liší především podle dostupných zdrojů, menší model je určen pro mobilní zařízení. Struktura sítí MobileNetV3 je zobrazena v tabulce 2.2.

Vstup	Operátor	SE	NL	s
$224^2 \times 3$	conv2d	-	HS	2
$112^2 \times 16$	bneck, 3×3	-	RE	1
$112^2 \times 16$	bneck, 3×3	-	RE	2
$56^2 \times 24$	bneck, 3×3	-	RE	1
$56^2 \times 24$	bneck, 5×5	✓	RE	2
$28^2 \times 40$	bneck, 5×5	✓	RE	1
$28^2 \times 40$	bneck, 5×5	✓	RE	1
$28^2 \times 40$	bneck, 3×3	-	HS	2
$14^2 \times 80$	bneck, 3×3	-	HS	1
$14^2 \times 80$	bneck, 3×3	-	HS	1
$14^2 \times 80$	bneck, 3×3	-	HS	1
$14^2 \times 80$	bneck, 3×3	✓	HS	1
$14^2 \times 112$	bneck, 3×3	✓	HS	1
$14^2 \times 112$	bneck, 5×5	✓	HS	2
$7^2 \times 160$	bneck, 5×5	✓	HS	1
$7^2 \times 160$	bneck, 5×5	✓	HS	1
$7^2 \times 160$	conv2d, 1×1	-	HS	1
$7^2 \times 960$	pool, 7×7	-	HS	1
960	conv2d 1×1	-	HS	1
1280	conv2d 1×1	-	-	1

Vstup	Operátor	SE	NL	s
$224^2 \times 3$	conv2d	-	HS	2
$112^2 \times 16$	bneck, 3×3	✓	RE	2
$56^2 \times 16$	bneck, 3×3	-	RE	2
$28^2 \times 24$	bneck, 3×3	-	RE	1
$28^2 \times 24$	bneck, 5×5	✓	HS	2
$14^2 \times 40$	bneck, 5×5	✓	HS	1
$14^2 \times 40$	bneck, 5×5	✓	HS	1
$14^2 \times 40$	bneck, 5×5	✓	HS	1
$14^2 \times 40$	bneck, 5×5	✓	HS	1
$14^2 \times 40$	bneck, 5×5	✓	HS	2
$7^2 \times 96$	bneck, 5×5	✓	HS	1
$7^2 \times 96$	bneck, 5×5	✓	HS	1
$7^2 \times 96$	conv2d, 1×1	✓	HS	1
$7^2 \times 576$	pool, 7×7	✓	HS	1
576	conv2d, 1×1	✓	HS	1
1024	conv2d, 1×1	✓	-	1

Tabulka 2.2: Zobrazení architektury MobileNetV3Large (vlevo) a MobileNetV3Small (vpravo) [16]. Levý sloupec značí rozměry vstupu do dané vrstvy. Operátor specifikuje, o jaký typ vrstvy se jedná. Sloupec SE udává, jestli daný blok obsahuje *Squueze–And–Exite*, což je úprava lineárních bottleneck bloků představená sítí typu MnasNet [49]. NL udává nelinearitu (aktivační funkci) vrstvy, HS značí funkci H–Swish a RL značí ReLU. Parametr *s* udává krok (*stride*) konvoluce.

2.3.1 Lokalizační metody

Lokalizace objektu je úloha, kde je cílem algoritmu nalézt pozici a rozměry hledaného objektu. Pro účely této práce se jedná o obličej, jeho přesná lokalizace je tak prvním nutným krokem celého výsledného systému. Nejjednodušší detekční systémy pracují na principu **posuvného okna**. Tyto algoritmy posouvají výřez s pevně danými rozměry po vstupním snímku a na výřezech provádějí klasifikaci (např. pomocí konvoluční neuronové sítě). Ačkoliv je tato metoda velmi jednoduchá na implementaci, není v praxi téměř používána kvůli problémům s ní spojeným.

Především je výše zmíněná metoda do značné míry pomalá. Výpočet je nutné opakovat mnohokrát a pro větší snímek je pak rychlost neúnosně pomalá. Dalším problémem je volba velikosti kroku. Příliš velký krok by mohl vynechat podstatné části snímku, příliš malý krok však vede k neúnosné náročnosti na výpočet. Se zmenšením kroku o polovinu je nutné provést čtyřnásobně více klasifikací.

Dalším problémem je velikost samotného okna. Při nekontrolovaných podmínkách jsou zachycené objekty různě vzdálené od kamery a tedy se jeví různě velké. Většinou je tak nutné použít více oken s různými rozměry, což opět vede k pomalému výpočtu.

Naopak výrazně rychlé zpracování nabízí algoritmus **Viola and Jones** [52] z roku 2001. Autoři navrhli systém pro detekci objektů (nejvíce byl však používán pro detekci obličeje), kterým bylo možné analyzovat i video v téměř reálném čase. Tento algoritmus využívá tzv. haarových příznaků, které mají tvar obdélníku nebo čtverce. Každý příznak se dělí na dvě části (obvykle značené jako bílé a černé části) a hodnota každého příznaku se spočítá jako rozdíl součtu intenzity pixelů pod těmito částmi. Jelikož je možných příznaků příliš velké množství, bylo by neefektivní pro detekci využívat všech. Viola a Jones tak využili modifikovaného algoritmu AdaBoost pro výběr nejvhodnějších příznaků a pomocí nich pak probíhá klasifikace.

Algoritmus vyniká především svojí rychlostí. Pro jeho běh není potřeba paralelizace na GPU, je možné jej využívat i na koncových zařízeních s menším výkonem. Tento algoritmus je také implementován v běžně dostupných knihovnách jako je OpenCV. V posledních letech se však výzkum zaměřuje více na využití neuronových sítí pro detekční úlohy. Ty obvykle dosahují výrazně lepší přesnosti a jsou robustnější pro detekci rozličných objektů.

Neuronových sítí využívají například algoritmy z rodiny **RCNN** (*Region–based Convolutional Neural Network*) [10]. Tento systém nejprve na vstupním snímku vyhledá kandidátní oblasti, které mohou obsahovat hledané objekty. Algoritmus postupuje tak, že se snaží spojit oblasti které patří stejnému objektu. Je využito toho, že pixely náležící stejnému objektu si jsou barevně blíže, než pixely rozdílných objektů. Takto je získáno asi 2000 výsečí ze kterých jsou pomocí CNN (architektura inspirovaná sítí AlexNet) vyextrahovány příznaky a nad nimi je následně provedena klasifikace. Autoři pro klasifikaci použili algoritmus *support vector machine* (SVM).

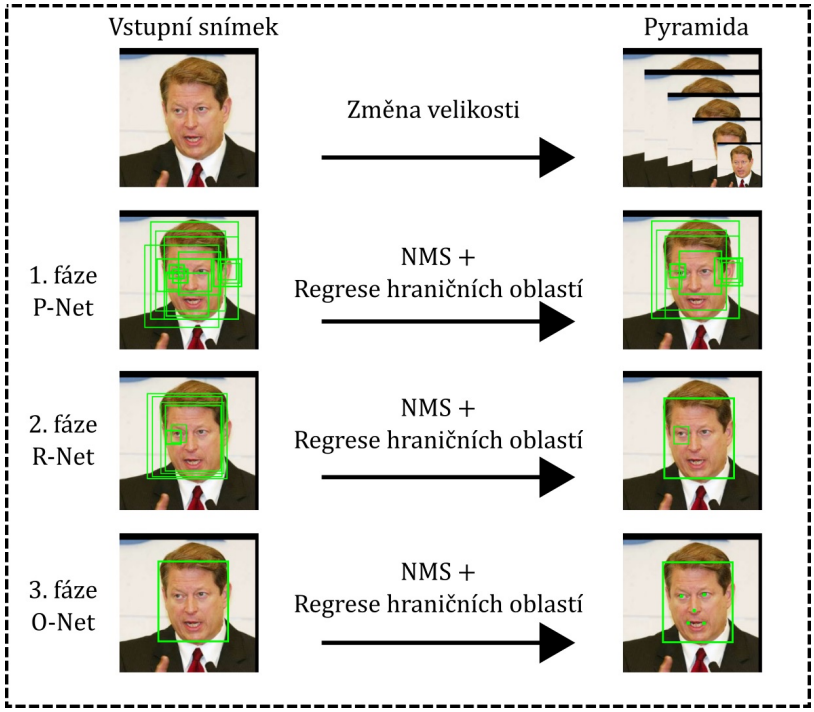
Celý systém byl následně několikrát aktualizován. Varianta Fast RCNN [9] využívá posunu ve výzkumu konvolučních sítí a namísto sítě AlexNet využívá síť VGG. Další změna nastala u klasifikace, kde namísto SVM byla použita funkce softmax, podobně jako je tomu u neuronových sítí starající se o klasifikaci obrazu. Tyto změny celý systém zrychlily asi desítinásobně. V roce 2016 pak byl představen model Faster RCNN [33], který modifikuje vyhledávání vhodných výsečí. Autoři představili *Region Proposal Network* (RPN), tedy plně konvoluční síť, která predikuje vhodné kandidátní oblasti. Tato změna dále zrychlila celý systém a zvýšila jeho přesnost.

Výše popsané modely jsou schopny lokalizovat libovolný počet objektů náležící do libovolného počtu tříd. Pro účely této práce je lze však natrénovat pro lokalizaci jedné třídy – lidského obličeje. Kromě tohoto přístupu existují i systémy, které se zaměřují pouze na lokalizaci obličeje. Mezi ně se řadí **Multi–task Cascaded Convolutional Network** (MTCNN) [59].

MTCNN je systém konvolučních sítí skládající se ze tří hlavních částí. První část systému se skládá z tzv. *Proposal Network* (P–Net), která se stará o vyhledávání vhodných kandidátních oblastí. K tomuto je využita plně konvoluční síť. Počet překrývajících se kandidátních oblasti je následně zredukován pomocí Non–Maxima Surpression (NMS).

NMS je algoritmus, které ze seznamu hraničních oblastí (B), korespondujících hodnot určujících jistotu (S) a hraniční úrovně překrytí (N) vytvoří nový, filtrovaný seznam (D) [35]. Algoritmus nejprve vybere hraniční oblast z B s nejvyšší jistotou a vloží ji do množiny D. Dále je vypočítán překryv této oblasti s ostatními oblastmi z množiny B. Pokud je překryv vyšší než N, pak je daná oblast vyřazena. Tento proces se opakuje, dokud není množina B prázdná. Tímto způsobem jsou vyřazeny oblasti, které jsou podobné jiné oblasti ohodnocené vyšší jistotou.

Druhá část systému, tzv. *Refine Network* (R–Net) dále zavrhuje nevhodné kandidátní oblasti. Třetí část – *Output Network* (O–Net) přidává k hraniční oblasti také souřadnice



Obrázek 2.11: Proces lokalizace obličeje pomocí algoritmu MTCNN se skládá ze tří hlavních částí. Nejprve jsou vygenerovány kandidátní oblasti pomocí Proposal Network (P-Net). Ty jsou následně zpřesněny pomocí Refinement Network (R-Net). Ve třetí fázi vygeneruje Output Network (O-Net) finální obdélník a označí pět klíčových bodů na obličeji.

pěti významných bodů na obličeji. Konkrétně se jedná o pozice očí, pozice koutků úst a pozici nosu. Celý proces algoritmu MTCNN je zobrazen na snímku 2.11.

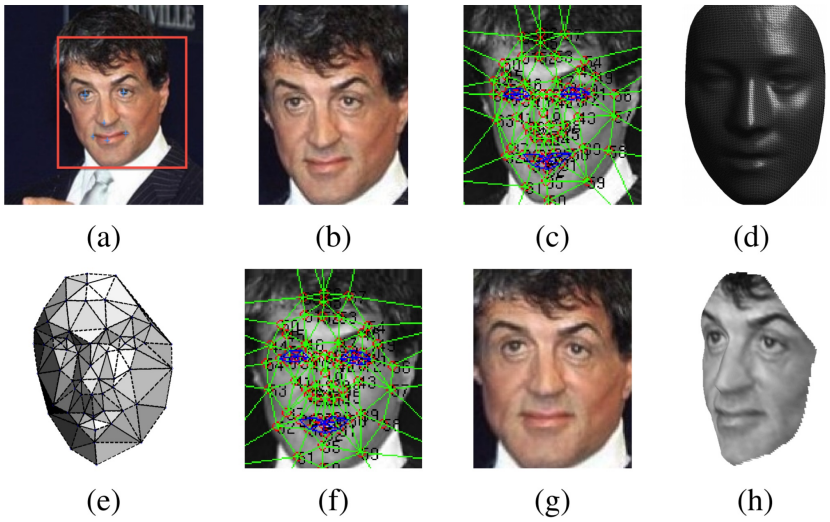
2.3.2 DeepFace

Systém DeepFace [48] z roku 2014 byl jedním z prvních pokusů o využití hlubokých konvolučních neuronových sítí pro identifikaci osob podle obličeje. Navržený algoritmus v prvním kroku provede detekci šesti bodů na obličeji (střed obou očí, špička nosu, střed úst a oba okraje úst), podle kterých se následně provede zvětšení a zarovnání 2–D obrazu.

Pro přesné natočení obličeje je dále vytvořen 3–D model tváře. Na obličeji je detekováno 67 bodů, podle které je obličej následně umístěn na obecný 3–D model tváře. Tento model je pak natočen čelem ke kameře. Odtud model opět převeden do 2–D reprezentace. Proces zarovnání obličeje je zobrazen na snímku 2.12.

Zarovnaný a natočený snímek je následně předán neuronové síti. Autoři zvolili pro extrakci příznaků vlastní architekturu. Snímek o rozměrech 152×152 pixelů je předán první konvoluční vrstvě s 32 filtry o velikosti $11 \times 11 \times 3$, následovaném max–pooling vrstvou s krokem o velikosti 2 [48]. Výsledek je pak předán druhé konvoluční vrstvě s 16 filtry o rozměrech $9 \times 9 \times 16$ [48].

Následují tři lokálně propojené vrstvy a dvě plně propojené vrstvy. Výstup poslední vrstvy je následně předán funkci K–softmax (K je zde počet tříd), která vytvoří rozložení pravděpodobnosti náležitosti snímku do dané třídy (identity). Autoři síť trénovali s chybo-



Obrázek 2.12: Vizualizace procesu zarovnání obličeje algoritmem deepface. Obličej je nejprve lokalizován (a) a je na něm detekováno šest klíčových bodů. Obličej je následně oříznut (b). Poté je na obličeji detekování 67 bodů (c). (d) zobrazuje obecný 3–D model, podle kterého je realizováno natočení. Na snímku (e) je vyobrazení natočení 3–D modelu ke kameře, tmavé trojúhelníky jsou otočeny směrem od kamery. Za pomoci tohoto modelu je pak obličej natočen směrem ke kameře (f), výsledek této operace je opět převeden na 2–D reprezentaci (g). Model (h) zobrazuje nově generovaný 3–D model obličeje (tyto modely nebyly pro tento algoritmus využity). Snímek byl převzat z [48].

vou funkcí *cross–entrophy* pomocí běžně používaného stochaistického gradientního sestupu. Trénování proběhlo na datasetu Social Face Classification.

Přesnost celého systému byla následně otestována na datasetech Labeled Faces in the Wild (LFW)[20] a YouTube Faces (YTF) [54]. Na datasetu LFW byl dosažen nejlepší výsledek 97,35 %, což je výsledek blížící se k přesnosti naměřené u lidí [25]. Autoři také měřili vliv předzpracování na přesnost. Při vynechání úprav natočení obličeje klesla přesnost na 94,3 %, bez zarovnání a natočení pak na 87,9 %. Na datasetu YTF byla naměřena přesnost modelu 91,4 % [48].

Ačkoliv se model v době vydání řadil na přední příčky žebříčku, v dnešní době byl již překonán sofistikovanějšími modely. Zejména architektura konvoluční sítě je z dnešního pohledu triviální, s použitím novější páteřní sítě lze na datasetu LFW získat přesnost přesahující 99 %.

2.3.3 FaceNet

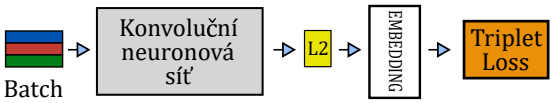
FaceNet [38] je systém vyvinutý společností Google na převod snímku na identifikační vektor příznaků (*face embedding*) reprezentující daný obličej. Systém je založen na hluboké neuronové síti, která je trénována tak, aby normalizovaná vzdálenost mezi identifikačními vektory odpovídala podobnosti obličejů. Snímky stejné osoby by tedy měly být systémem vyhodnoceny jako blízké vektory, vektory rozdílných lidí by naopak měly být více vzdálené. Jakmile síť vytvoří vektor reprezentující osobu, lze pomocí něj provádět běžné biometrické úkony. Verifikaci pomocí obličeje lze provést jako výpočet rozdílu mezi dvěma vektory,

jedním z databáze osob a druhým pořízeným z kamery, vyhledání podobných osob lze provést shlukovacími metodami jako je například algoritmus k–means.

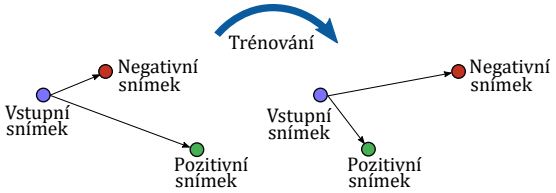
Předchozí systémy pro identifikaci osob využívaly rozsáhlé klasifikační vrstvy [43], což mělo za následek, že výstup sítě byl obvykle poměrně velký (až tisíce rozměrů). Oproti tomu FaceNet produkuje vektor příznaků o relativně malých rozměrech, což má za následek rychlejší následné zpracování. Autoři pro reprezentaci použili 128–D vektor čísel s pohyblivou plovoucí čárkou, nicméně dle jejich testování lze vektor zredukovat na 128 bytů bez ztráty přesnosti [38]. Další zmenšení je možné, je však vykoupené malou ztrátou přesnosti a je tak vhodné pro mobilní zařízení.

Autoři jako páteřní síť vyzkoušeli několik architektur. Především se zaměřili na síť Zeiler&Fergus (známá jako ZFNet) [58] a na varianty sítě Inception [47]. Nejvyšší přesnost byla naměřena u sítí Zeiler&Fergus (87,9 %), Inception 224×224 (89,4 %) a Inception 160×160 (88,3 %). Je však zřejmé, že pro páteřní síť by bylo možné využít i jiné architektury. Model systému FaceNet je zobrazen na snímku 2.13

Úloha sítě se dá popsat jako vytvoření reprezentace obličeje pomocí vektoru příznaků $f(x) \in \mathbb{R}^d$ ze snímku x tak, aby vzdálenost mezi všemi snímky obličejů zaznamenaných v různých světelných podmínkách a s různým natočením byla malá pro stejnou osobu a vysoká pro rozdílné osoby [38]. Autoři k tomuto účelu navrhli postup trénování sítě využívající chybovou funkci *triplet loss*, která podporuje tvorbu vzdálenosti (*margin*) mezi dvojicemi snímků různých osob.



Obrázek 2.13: Architektura sítě FaceNET. Snímky jsou nejprve zpracovány hlubokou konvoluční neuronovou sítí, následně jsou L_2 normalizovány. Výstupem normalizace je vektor příznaků (*face embedding*). Při trénování je pak chyba vyhodnocena pomocí funkce *triplet loss*.



Obrázek 2.14: Trénování s využitím funkce *triplet loss* vede ke zmenšení vzdálenosti mezi vstupním snímkem a snímkem s positivní identitou, snímky s negativní identitou jsou naopak trénováním maximalizovány.

Správnou tvorbu příznakových vektorů lze popsat vztahem:

$$\begin{aligned} &\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2, \\ &\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \tau, \end{aligned} \tag{2.5}$$

vektor příznaků $f(x_i^a)$ vstupního snímku x_i^a (*anchor*) zobrazující danou osobu je blíže vektorům příznaků $f(x_i^p)$ pozitivního snímku x_i^p zobrazující stejnou osobu, než vektorům příznaků $f(x_i^n)$ negativních snímků x_i^n zobrazující jiné osoby [38]. Parametr α zde značí vzdálenost (*margin*), kterou se metoda snaží umístit vektory různých osob dál od sebe. τ je množina všech trojic snímků o velikosti N, ze kterých se síť trénuje (vstupní snímek, snímek s pozitivní identitou a snímek s negativní identitou). Odtud lze chybu, která je trénování minimalizována, definovat jako:

$$L = \sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n) + \alpha\|_2^2 \right]_+ . \tag{2.6}$$

Tento proces trénování je vizualizován na snímku 2.14.

Autoři algoritmu uvádějí, že pro zvýšení přesnosti je při trénování důležitá volba vhodných trojic snímků. Vygenerování všech možných trojic by vedlo k tomu, že velký počet z nich by velmi lehce splnil podmínku z rovnice 2.5 a volbou složitých trojic, které přispívají ke zlepšení modelu, lze dosáhnout lepší výsledné přesnosti [38]. Toto znamená, že pro vstupní snímek x_i^a je vhodné vybrat snímek s pozitivní identitou x_i^p jako $argmax_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$ a snímek s negativní identitou jako $argmin_{x_i^p} \|f(x_i^a) - f(x_i^n)\|_2^2$. Tímto způsobem lze získat náročné pozitivní a negativní snímky (*hard positive, hard negative*). Autoři uvádějí, že není vhodné provádět tento výpočet pro celý dataset, jelikož by tato operace byla příliš výpočetně složitá a možné špatně označené snímky datasetu by se zde vždy projevily. Z tohoto důvodu autoři navrhli dva způsoby řešení.

První je tzv. offline generování, kdy je trénování sítě zastaveno po n krocích a z posledního checkpointu je vypočítán *argmax* a *argmin* na části datasetu. Oproti tomu při online generování jsou složité trojice vybírány při trénování z mini–batch. Autoři systémy FaceNet experimentálně zjistili, že pro tento systém bylo vhodné generování online, ve většině experimentů tedy volili složité trojice z mini–batch o velikosti 1800 vzorků.

Při evaluaci modelu na datasetu LFW dosáhl systém přesnosti 98,87 % při použití protokolu pro neomezené využití externích dat. S tímto výsledkem se algoritmus zařadil mezi nejúspěšnější *state–of–the–art* metody.

2.3.4 ArcFace

Autoři algoritmu ArcFace [6] identifikovali chybovou funkci při trénování sítě jako rozhodující faktor ovlivňující její konečnou přesnost. Z tohoto důvodu navrhli funkci *Additive Angular Margin Loss*, jejímž účelem je oddělit vzorky dat z odlišných tříd a snížit rozdíly vzorků stejné třídy. Funkce vychází z běžně používané funkce *softmax loss*, kterou autoři definují jako:

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}, \tag{2.7}$$

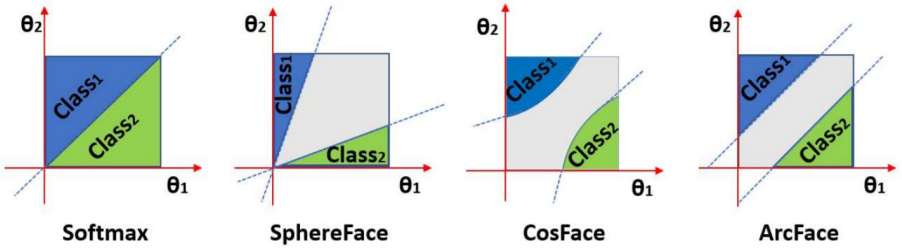
kde $x_i \in \mathbb{R}^d$ značí příznak vzorku i, který náleží do třídy s indexem y_i. Pro vektor příznaků o velikosti d označuje $W_j \in \mathbb{R}^d$ sloupec s indexem j váhy $W \in \mathbb{R}^{d \times n}$ a $b_j \in \mathbb{R}^n$ označuje hodnotu bias. Proměnná N značí velikost trénovací dávky (*batch*) a n značí počet tříd v datasetu [6].

Výpočet chyby byl odtud autory upraven na:

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N} \frac{e^{s(cos(\theta_{y_i}+m))}}{e^{s(cos(\theta_{y_i}+m))} + \sum_{j=1, j\neq y_i}^{n} e^{s\ cos\ \theta_j}}. \tag{2.8}$$

Došlo tedy k úpravě hodnoty bias na $b_j = 0$, dále k upravě $W_j^T x_i = \|W_j\|\ \|x_i\|\ cos\ \theta_j$, kde θ_j značí úhel mezi váhou W_j a příznakem x_i. Dále pomocí L_2 normalizace došlo k nastavení váhy $\|W_j\| = 1$ a velikost příznaku $\|x_i\|$ byla nastavena na s. Parametr m značí úhlový rozdíl (*angular margin*) mezi x_i a W_{y_i} [6]. Vizualizace oddělení vzorků pomocí této funkce a ostatních chybových funkcí je na snímku 2.15.

Jako páteřní síť na extrakci vektoru příznaků zvolili autoři síť ResNet [14] ve variantách ResNet50 a ResNet100. Pro trénování bylo sloučeno několik datasetů do jednoho, konkrétně byly použity datasety CASIA[56], VGGFace2[2], MS1MV2 a DeepGlint-Face. Po trénování



Obrázek 2.15: Chybová funkce využívaná algoritmem ArcFace je výpočetně podobná ostatním chybovým funkcím, liší se však geometrickým tvarem parametru *margin*. Snímek porovnává tento tvar u několika funkcí využívaných k generování identifikačních vektorů. Světle modrou čarou je zobrazena rozhodovací hranice, šedě zobrazená oblast pak značí *margin*. Autoři algoritmu ArcFace argumentují, že přesnost algoritmu vychází právě z lineárního geometrického rozdělení tříd. Snímek byl převzat z [6].

byla úspěšnost modelu ověřena na datasetech Labeled Faces in the Wild[20] a Youtube Faces DB, na kterých dosáhl přesnosti 99,83 % a 98,02 %.

2.3.5 Probabilistic Face Embeddings

Většina *state–of–the–art* metod vytváří embedding jako vektor popisující obličej podle jejich rysů (*features*). Algoritmy při tvorbě těchto vektorů neberou v potaz, zda–li si je model jistý přítomností určitého rysu. Dle autorů algoritmu využívajícího Probabilistic Face Embedding [39] z roku 2019 je toto problematické, protože vlivem degradované kvality snímku může být přítomnost některých rysů nejistá. V praxi je velká část vstupních obrazových dat poškozených a zjistit přítomnost některých rysů nemusí být pro model možné, což následně vede ke vzdálení vektorů.

Jako řešení tohoto problému byl navrhnut systém využívající tzv. probabilistické vektory obličeje (*probabilistic face embeddings*), které rysy namísto bodu v prostoru definují jako rozdělení pravděpodobnosti. Parametry tohoto rozdělení pak určují, do jaké míry si je síť jistá přítomností daného rysu. Výstupem modelu je pak distribuce $p(z|x)$, která reprezentuje potenciální vzhled obličeje. Konkrétně se jedná o normální distribuci:

$$p(z|x_i) = N(z; \mu_i, \sigma_i^2 I), \tag{2.9}$$

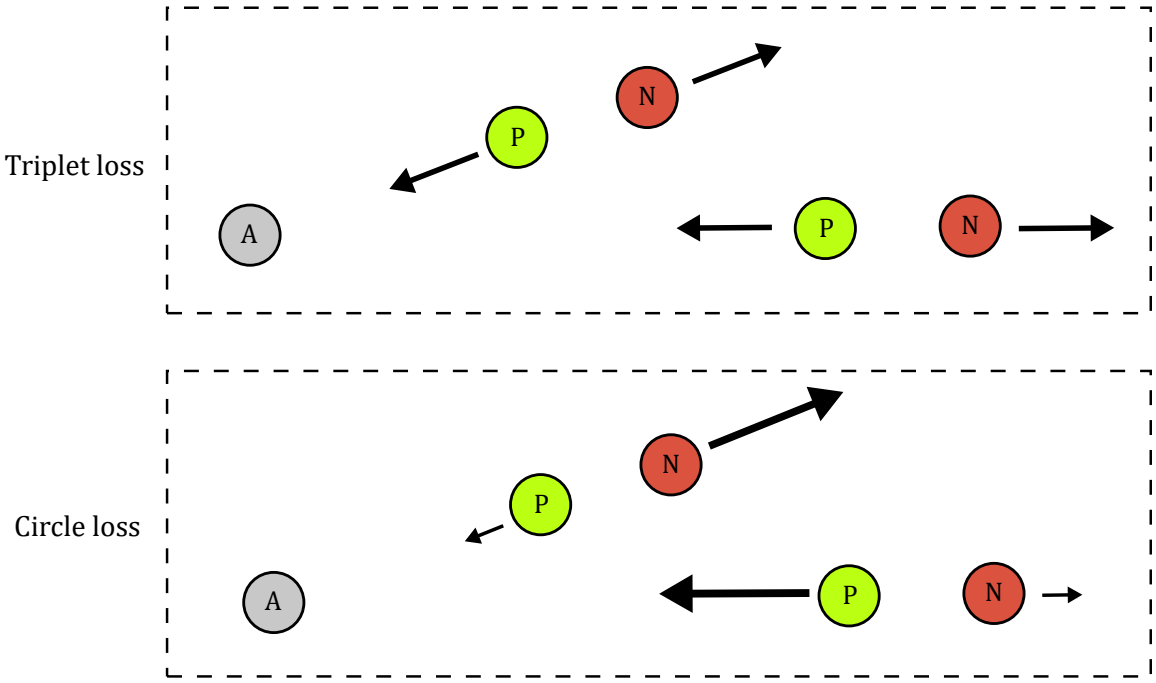
kde μ_i a σ_i jsou vícedimenzionální vektory generované sítí ze vstupního obrázku x_i [39]. ${}_m u$ zde značí přítomnost nejpravděpodobnějších rysů a σ značí míru, do které si je model jistý přítomností těchto rysů. Autoři obě tyto distribuce vytváří pomocí stejné sítě.

Tuto metodu popisu obličejů lze použít v kombinaci s existujícími architekturami sítí. Autoři testovali úspěšnost *state–of–the–art* modelů v původní verzi (generující deterministický embedding) a ve verzi generující pravděpodobnostní rozložení. Experimenty ukázaly, že reprezentace rysů pomocí pravděpodobnosti vedlo ke zlepšení přesnosti modelů.

2.3.6 Circle Loss

Systém využívající chybové funkce Circle Loss byl představen v roce 2020 [44]. Autoři argumentují, že dosud využívané optimalizační funkce jsou málo flexibilní, jelikož při porušení podmínky a generování chyby je typicky počítáno s konstantní chybou. Tato konstantní

hodnota je určena pro každou chybně vyhodnocenou trojici, při trénování však není zohledněno, jak moc je podmínka porušena. Pro trénování sítě by bylo výhodné, kdyby u dvojic vektorů *positive–negative*, které jsou blízko vektoru *anchor*, byla optimalizována vzdálenost mezi *anchor* a *negative*. Naopak u dvojic, které jsou vzdálené vektoru *anchor*, by byla výhodná optimalizace mezi vektory *anchor* a *positive*. Tento proces je demonstrován na snímku 2.16.



Obrázek 2.16: Ukázka procesu optimalizace u triplet loss a circle loss. Vektor *anchor* je zde označen jako A, *positive* jako P a vektor *negative* jako N. Triplet loss při trénování posune pozitivní a negativní vektory stejnou mírou u obou dvojic. Naopak funkce triplet loss u bližší dvojice bude více optimalizovat pozici negativního vektoru a u vzdálené dvojice pozici pozitivního vektoru.

Tento způsob optimalizace je výhodný především u případů, kdy je vzdálenost snímků *anchor* a *positive* již malá a je tak vhodné se zaměřit spíše na oddálení negativní identity. Podobně v případech, kdy je vzorek *negative* již dostatečně odlišný, je výhodné zaměřit se na přiblížení pozitivní identity.

Samotná funkce center loss je pak definována jako:

$$\mathcal{L}_{cirle} = \log[1 + \sum_{j=1}^{L} \exp(\gamma \alpha_n^j (s_n^j - \Delta_n)) \sum_{i=1}^{K} \exp(-\gamma \alpha_p^i (s_p^i - \Delta_p))], \tag{2.10}$$

kde L značí počet negativních snímků, K značí počet pozitivních snímků [44]. Dále s_p^i je vzdálenost pozitivních vektorů a s_n^j je vzdálenost negativních vektorů. γ je hyperparametr škálování, Δ_n a Δ_p značí *margin* pro negativní a pozitivní snímky. a_j^n a a_p^i jsou váhové faktory pro negativní a pozitivní snímky.

Autoři s touto chybovou funkcí natrénovali několik modelů, zaměřili se na úlohy identifikace a verifikace osob podle obličeje. Nejúspěšnější model s páteřní sítí ResNet100 dosáhl *state–of–the–art* přesnosti v identifikační úloze. Oproti ostatním modelům se zlepšil o dvě

desetiny procenta na datasetu MFC1. V úloze verifikace se síť s páteřní konvoluční sítí ResNet34 zlepšila o řádově o setiny procenta na datasetech LFW, YTF a CFP-FP.

2.4 Frameworky pro strojové učení

Jelikož programování neuronových sítí od základů by bylo velmi obtížné, drtivá většina modelů je implementována v rámci nějakého frameworku, který tuto práci usnadňuje. Framework typicky obsahuje předdefinované vrstvy neuronových sítí, chybové funkce, aktivační funkce a algoritmy pro trénování. Programátor tak nemusí například implementovat zpětné šíření chyby pro svůj vlastní model, ale většinou stačí nadefinovat architekturu sítě a specifikovat proces trénování. Framework také obvykle obsahuje vysoce optimalizované implementace výpočtů a díky paralelizaci lze dosáhnout řádově rychlejšího trénování. Z tohoto důvodu je volba vhodného frameworku klíčovým krokem při vývoji modelů strojového učení. Nejpoužívanější frameworky jsou podrobněji popsány v následujících podkapitolách.

2.4.1 Tensorflow

Tensorflow[3] je volně dostupná knihovna obsahující nástroje pro strojové učení. Tento framework je vyvíjen výzkumnou skupinou Google Brain a je dostupný pod svobodnou licencí Apache 2.0. Z toho plynoucí výhodou je, alespoň v současné době, rychlý vývoj a velmi kvalitně zpracovaná dokumentace.

Ačkoliv Tensorflow obsahuje nástroje pro rozličné úlohy z oblasti oblasti strojového učení, je nejvíce využíván pro definici architektury a trénování modelů hlubokých neuronových sítí. Spolu s frameworkem Torch je jedním z nejvíce využívaných nástrojů pro oblast počítačového vidění.

Původní verze Tensorflow (tzv. Tensorflow 1.X) byla implementována jako nízkoúrovňový framework, kde programátor měl jednoduchý přístup ke všem aspektům vývoje modelu. Tento nízkoúrovňový přístup však nebyl jednoduchý na osvojení pro začínajícího vývojáře, z tohoto důvodu byl ve verzi 1 čistý Tensorflow využíván zřídka. Častější byla jeho kombinace s frameworkem Keras, který zjednodušoval práci se sítí.

Tento stav byl změněn s příchodem Tensorflow 2, který integroval většinu funkcionality z Frameworku Keras přímo do Tensorflow (modul tf.keras). Při programování si tak vývojář může zvolit, s jakou úrovní abstrakce chce pracovat. Tensorflow 2 také přinesl nový způsob vykonání operací. Pomocí tzv. *eager execution* lze vyhodnocovat operace za běhu programu. Není tak již nutné sestavit výpočetní graf, který pak běží na pozadí. Toto je výhodné především při debuggování, jelikož operace přímo navrací hodnoty a hledání chyb v programu je tak jednodušší.

2.4.2 Keras

Keras[4] je open–source knihovna, která poskytuje vysokoúrovňové rozhraní pro Tensorflow. Do verze Keras 2.3 jej bylo možné používat také s frameworky Theano, Microsoft Cognitive Toolkit a PlaidML, ale od verze 2.4 je již jediným podporovaný backend Tensorflow. Keras je vyvíjen pod svobodnou licencí MIT.

Keras obsahuje nástroje zaměřené na vývoj a trénování neuronových sítí. Obsahuje běžně používané vrstvy, aktivační funkce a nástroje pro předzpracování obrazu. Kromě

[3] https://www.tensorflow.org/
[4] https://keras.io/

práce s neuronovými sítěmi však neobsahuje nástroje z jiných oblastí strojového učení. Framework se snaží poskytnout vývojáři vysokou míru abstrakce při práci s modely, je tak možné sestavit a natrénovat vlastní neuronovou síť s velmi minimálním zdrojovým kódem.

2.4.3 PyTorch

Torch byla open–source knihovna pro strojové učení, založený na skriptovacím jazyce Lua. Backend frameworku byl vyvíjen v jazyce C. Od roku 2018 není tento framework dále vyvíjen. Namísto toho je vyvíjen z něj odvozený framework PyTorch[5], který obsahuje rozhraní pro jazyk Python a do menší míry i pro C++.

PyTorch obsahuje podobně jako Tensorflow široké spektrum nástrojů pro strojové učení. Kromě nástrojů pro práci s neuronovými sítěmi lze pomocí PyTorch jednoduše akcelerovat výpočty s maticemi na GPU. Často využívaný je tento framework i v oblasti počítačového vidění.

PyTorch je vyvíjen pod záštitou Facebooku v laboratoři pro výzkum umělé inteligence (Facebook's AI Research lab – FAIR) pod svobodnou licencí BSD. PyTorch je často volen jako vhodný framework pro vývoj modelů ve firemním prostředí. Je využíván například v projektu Tesla Autopilot.

2.4.4 Caffe

Framework CAFFE[6] (Convolutional Architecture for Fast Feature Embedding) je framework zaměřený na práci s hlubokými neuronovými sítěmi. Framework byl vyvinut na univerzitě Berkley v Kalifornii a je vydáván pod svobodnou licencí BSD. Framework je sepsán v C++, ale obsahuje také rozhraní pro Python. V roce 2017 oznámil Facebook vývoj Caffe2, nicméně později byl tento modul zařazen pod PyTorch a v současné době není dále vyvíjen.

2.4.5 Theano

Theano[7] je knihovna pro Python zaměřená na práci s maticemi. Výpočty je možné akcelerovat na GPU s podporou technologie CUDA. Výrazy v Theano jsou syntakticky podobné modulu numpy pro python. Framework byl vyvíjen Institutem pro strojové učení v Montrealu (Monreal Institute for Learning Algorithms – MILA), nicméně v roce 2017 byl oznámen konec vývoje, z důvodu zvýšeného počtu alternativních frameworků.

2.4.6 Darknet

Framework Darknet[8] je open–source projekt sepsaný v jazyce C s podporou CUDA. Tento framework je proslulý především díky detektorům objektů typu YOLO, které v něm byly vytvořeny. Ačkoliv Darknet obsahuje nástroje pro tvorbu vlastních modelů, není však (kromě modelů YOLO) vývojáři příliš používán. O něco častější využití má verze přeložená do Tensorflow – tzv. Darkflow[9].

[5] https://pytorch.org/
[6] http://caffe.berkeleyvision.org/
[7] https://github.com/Theano/Theano
[8] https://pjreddie.com/darknet/
[9] https://github.com/thtrieu/darkflow

Kapitola 3

Návrh řešení a implementace

Pro implementaci byla vytvořena vlastní architektura, která čerpá poznatky z algoritmů popsaných v kapitole 2.3. Výsledky z těchto prací často nejsou bez provedení modifikací replikovatelné, protože modely byly trénovány na neveřejných datasetech, využívaly speciální hardware pro trénování a zarovnávaly obličej pomocí neveřejných algoritmů. Síť navržená v této práci využívá volně dostupná data a proces trénování lze provést na běžné GPU s podporou technologie CUDA. Dosažené výsledky jsou tak replikovatelné, popřípadě mohou být získané znalosti přeneseny na řešení podobných úloh.

3.1 Datasety

Základní predispozicí pro úspěšné trénování neuronových sítí je rozsáhlý dataset obsahující snímky daného subjektu a manuálně nebo strojově získané anotace. Jelikož je problematika reidentifikace osob často řešeným problémem, je výběr datasetu poměrně rozsáhlý. Níže popsané datasety jsou volně přístupné pro výzkumné účely, většina je distribuována pod otevřenou licencí. Jejich porovnání je shrnuto v tabulce 3.1.

Kromě nich existují uzavřené, veřejnosti nepřístupné datasety, které jsou často výrazně rozsáhlejší. Například Facebook používá pro trénování vlastní dataset obsahující 500 milionů snímků zachycujících přes 10 milionů osob [2], Google pro trénování modelu FaceNet [38] využil vlastní dataset obsahující 200 milionů fotografií zachycujících 8 milionů osob.

3.1.1 Labeled Faces in the Wild

Labeled Faces in the Wild (LFW) [20] je veřejně dostupný dataset zaměřený na verifikaci osob pomocí snímků obličeje. Sada obsahuje více než 13 000 snímků. Každý snímek obsahuje jednu osobu se známou identitou. Celkem se v datasetu nachází 5749 osob, z toho 1680 osob se vyskytuje na více než jednom snímku. Dataset je poskytnut ve verzi s neupravenými obličeji nebo ve verzi se zarovnanými obličeji.

Na rozdíl od starších datasetů, LFW se nesnaží o normalizaci a unifikaci snímků. Ty tak obsahují snímky osob s přirozenou variací natočení, pózy, osvětlení, zaostření a výrazu tváře. Všechny snímky k dispozici v rozlišení 250×250 pixelů, většina snímků je v barevném provedení, malá část je černobílá.

Jelikož vlastní rozdělení na trénovací, validační a testovací část může mít vliv na skóre testovaného algoritmu, autoři datasetu poskytují předem definované rozdělení. Toto rozdělení by mělo být dodrženo z důvodu přesného porovnání mezi jednotlivými identifikačními modely. Jelikož se tento dataset svou velikostí řadí spíše mezi menší, není často využíván

pro trénování modelů. Slouží však jako jeden z nejvíce využívaných datasetů pro srovnání přesnosti různých modelů. Na webových stránkách datasetu je také přístupný žebříček nejúspěšnějších modelů.

3.1.2 CelebFaces Attributes Dataset

CelebFaces Attributes Dataset (CelebA)[29] je obsáhlý dataset s více než 200 000 snímky obličejů celebrit. V datasetu se nachází 10 177 rozdílných identit, každá z fotografií je anotována 40 atributy. Atributy blíže popisují zachycenou osobu, jedná se například o informace o barvě vlasů, zdali daná osoba má nasazené brýle, jestli má vousy a podobné. Z hlediska této práce tyto atributy nejsou podstatné, důležitá je identita osoby, která je v tomto datasetu popsána unikátním číslem – ID. Jména osob nejsou v datasetu zveřejněna.

Samotné snímky jsou ve formátu `.jpg` v rozlišení 178×218 v barevném provedení. Autoři poskytují přesné rozdělení snímků na trénovací, validační a testovací část.

3.1.3 Youtube Faces Database

Youtube Faces Database (YDB) [54] je dataset zaměřující se na identifikaci osob ve videu. Skládá se z celkem 3425 video souborů, na kterých se vyskytuje 1595 různých osob. Všechna videa byla stažena z Youtube, pořízené záznamy tedy obsahují vysoce rozdílné ukázky natočení, rozlišení a světelných podmínek. Extrakcí snímků z videa také může dojít k dalšímu snížení kvality kvůli kompresi a rozmazání snímku pohybem. Osoby ve videích byly identifikovány pomocí kombinace automatických a manuálních technik, průměrně se každá osoba vyskytuje v 2,15 videích.

3.1.4 Yale Face Database B

Dataset Yale Face Database B (YFDB) [8] obsahuje celkem 5760 snímků zachycujících 10 různých osob v širokém spektru světelných podmínek. Původní dataset byl vytvořen tak, že kolem focené osoby bylo umístěno 64 počítačem řízených světel, které osobu postupně osvítily. Pro každé světlo byl zachycen jeden snímek. Každá z osob byla vyfocena v devíti různých pozicích. Dále byl pro každou pozici zachycen také snímek bez osvětlení. Snímky jsou zarovnané na obličej a jsou zachycené pouze v odstínech šedi v rozlišení 168×192.

Dataset byl později rozšířen, The Extended Yale Face Database B obsahuje dalších 28 osob zachycených ve stejných podmínkách jako v původním datasetu. Celkem je tedy v rozšířeném datasetu 16 128 snímků.

3.1.5 VGGFace2

Dataset VGGFace2 [2] se svými rozměry řadí mezi nejrozsáhlejší volně přístupné datasety. Celkem 9131 osob je zachyceno na více než 3,31 milionech snímcích. Fotografie byly sesbírány pomocí Google Image Search a zachycují tak široké spektrum osob v různorodých podmínkách (různé pózy, světelné podmínky, úhel natočení ke kameře a kvalita fotografie). Průměrně je každá osoba na 362 snímcích, nejméně zachycená osoba je na 87 snímcích, nejvíce zachycená osoba pak na 843.

3.1.6 SCface

Surveillance Cameras Face Database (SCface) [13] je dataset zaměřující se na identifikaci osob podle obličeje. Snímky byly pořízené v nekontrolovaném prostředí, pochází z pěti bez-

pečnostních kamer, které byly umístěny na Elektrotechnické fakultě Záhřebské univerzity. Tyto kamery se liší ve kvalitě záznamu. Tomu odpovídá natočení obličeje vzhledem ke kameře. Osoby se dívají směrem pod kameru, která bývá umístěna u stropu místnosti. Dataset obsahuje celkem 4160 snímků na kterých je zachyceno 130 různých osob. Kamery snímaly ve viditelném nebo infračerveném světelném spektru.

Dataset se svým rozsahem řadí spíše k menším datasetům. Autoři jej doporučují používat na testování identifikačních modelů využívajících sledovacími systémy. V těchto případech je snímek z kamery porovnán se snímkem z databáze.

3.1.7 Casia–Webface

Dataset Casia–Webface [56] vznikl ze snahy snížit rozdíly mezi veřejně dostupnými a privátními datasety v oblasti identifikace osob. Autoři získali snímky z filmové databáze IMDb, kde automatickým sběrem fotografií celebrit pořídili přes 500 000 snímků zachycujících přes 10 000 osob. Posbírané snímky byly anotovány poloautomatickým systém. Rozměry u všech fotografií byly nastaveny na 250×250 pixelů. Zachycené osoby jsou v rozmanitých situacích a v různých pózách. Některé z fotografií jsou barevné, některé v odstínech šedi.

3.1.8 FEI Face Database

Fei Face Database je dataset obsahující obličeje zachycený v laboratoři umělé inteligence na brazilské univerzitě Centro Universitário da FEI [50]. Celkem se v něm vyskytuje 200 unikátních osob. Každá z nich je zachycena na 14 fotografiích. Všechny snímky jsou v barevném provedení, focené s bílým pozadí a s uniformním osvětlením. Každá osoba je zachycena s různým natočením obličeje.

Kromě této varianty zaměřené na testování robustnosti algoritmů vůči natočení je dataset i ve verzi s frontálními snímky, který byl manuálně zarovnán a oříznut. V této variantě je každá osoba zachycena na dvou snímcích, jednou s neutrálním výrazem a podruhé s úsměvem.

Dataset	Počet snímků	Počet osob	Rozlišení	Barvy	Podmínky
LFW	13 000	5749	250×250	Různé	Nekontrolované
CelebA	200 000	10 177	178×218	Barevně	Nekontrolované
YDB	3425 videí	1595	200×200	Šedé	Kontrolované
EYFDB	5760	28	168×192	Šedé	Kontrolované
VGGFace2	3,31 milionů	9131	136×155	Barevně	Nekontrolované
SCface	4160	130	Různé	Různé	Nekontrolované
Casia	500 000	10 000	250×250	Různé	Nekontrolované
FEI Face	2800	200	640×480	Barevné	Kontrolované

Tabulka 3.1: Srovnání datasetů zaměřených na identifikaci osob podle obličeje.

3.2 Architektura

Pro extrakci příznaků reprezentujících obličej bylo vybráno několik konvolučních neuronových sítí. Konkrétně testované byly modely z rodiny ResNet, model InceptionV3, model DenseNet a model EfficientNet. K těmto modelům byl také natrénován model MobileNet,

což je architektura optimalizovaná pro nízké výpočetní nároky. Struktura těchto sítí je podrobně popsána v sekci 2.3. Jelikož byly tyto sítě původně navrženy na klasifikační úlohy, obsahovaly ke konci sítě plně propojené vrstvy, které byly odstraněny.

Nad páteřní sítí byla umístěna nová vrstva provádějící operaci global–average–pooling, za kterou pak následuje normalizační vrstva (*batch–normalization layer*). Po těchto vrstvách byla přidána nová plně propojená vrstva. Počet jejich neuronů byl pevně stanoven na 1024. Za ní pak následuje výstupní vrstva, kde počet neuronů je roven velikosti výsledného embedding vektoru. Pro účely této práce byla zvolena hodnota 128, podobně jako u architektury FaceNet. Za tuto vrstvu byla umístěna normalizační vrstva, která vyextrahované příznaky l_2 normalizuje pomocí vztahu:

$$y = \frac{x}{\|x\|_2} \tag{3.1}$$

kde x značí vstupní vektor a $\|x\|_2$ je jeho velikost. Tato normalizace zaručuje, že vektory reprezentující identitu osob mají stejnou velikost a lze je tak jednoduše vzájemně porovnávat. Architektura je shrnuta v tabulce 3.2.

Jelikož první plně propojená vrstva obsahuje relativně velké množství parametrů, byla zkoumána i varianta, kde byla za tuto vrstvu umístěna dropout vrstva. Tato varianta sítě však nevykazovala zlepšení oproti výše zmíněnému modelu a z tohoto důvodu nebyla tato vrstva do finální architektury zahrnuta.

Pro optimalizaci parametrů byly využity chybové funkce zaručující nízkou variabilitu výstupních vektorů mezi vzorky stejné třídy a vyšší variabilitu u vzorků rozdílných tříd. Více o jednotlivých funkcích je uvedeno v podkapitole 4.3.

Vrstva	**Rozměry výstupu**	**Počet parametrů**
Vstup	224×224	-
BackBone CNN (MobileNetV3Large)	(Batch Size, 7, 7, 1280)	4 226 432
Global Average Pooling 2D	(Batch Size, 1280)	0
Dense (ReLU)	(Batch Size, 1024)	1 311 744
Batch Normalization	(Batch Size, 1024)	4096
Dense (Linear)	(Batch Size, 128)	131 200
Lambda (L2 Normalization)	(Batch Size, 128)	0
Celkem parametrů: 5 673 472		
Trénovatelných parametrů: 5 647 024		
Netrénovatelných parametrů: 26 448		

Tabulka 3.2: Architektura modelu a počet parametrů na vrstvu. Extrakce příznaků je realizována pomocí páteřní konvoluční sítě (zde MobileNet) s odstraněnými plně vrchními vrstvami. Za nimi je přidána vrstva provádějící operaci average pooling. Následuje nová plně propojená vrstva o velikosti 1024 neuronů s aktivační funkcí ReLU. Poté je provedena batch–normalizace a následně je výstup tvořen plně propojenou vrstvou bez aktivace (s počtem neuronů dle velikosti embedding). L_2 normalizace je realizována jako tensorflow lambda vrstva.

Pro implementaci sítí byl zvolen framework Tensorflow ve verzi 2.4.1. Tento framework obsahuje většinu funkcionality potřebné pro definici a trénování výše zmíněné architektury. Kromě toho obsahuje také nástroje pro předzpracování datasetu a jeho rozdělení na trénovací dávky, které usnadňují vývoj.

3.3 Trénování

Velké množství modelů pro identifikaci osob využívá pro trénování dataset VGGFace2 3.1.5, který díky svému rozsahu (3,3 milionů snímků) umožňuje robustní trénování identifikátorů. Tento dataset byl také první volbou pro tuto práci. Nicméně v době tvorby této práce byl tento dataset autory odstraněn a není tak již volně dostupný. Z tohoto důvodu bylo nutné pro trénování zvolit menší dataset. Jako vhodnou náhradou se ukázal být Casia–WebFace 3.1.7 obsahující více než 500 000 snímků. Tento dataset byl dále rozdělen na trénovací a validační část v poměru 8:2. Pro trénování bylo tedy použito asi 400 000 snímků.

Pro trénování modelů byl vytvořen Jupyter Notebook starající se o tvorbu a předzpracování datasetu, definici architektury modelů a jejich následné trénování. Tento notebook byl zprovozně v cloudovém prostředí Google Colaboratory a pro akceleraci trénování byla využita GPU nVidia Tesla P100 s 16GB paměti. V prostředí Colaborary jsou k dispozici i jednotky pro zpracování tensorů (TPU), ale jejich využití neurychlilo trénovací proces oproti GPU.

Z trénovací množiny byly vyloučeny identity s příliš nízkým počtem snímků. U těchto osob by nastal problém, kdy by nebylo možné vytvořit dostatečné množství dvojic s touto identitou. Ostatní identity byly v každé epoše náhodně promíchány do trénovacích dávek tak, že v každé dávce bylo vybráno M identit a pro každou identitu bylo vybráno N snímků, které tvořily trénovací dávku. Velikost dávky pak tedy odpovídá vztahu $N \times M$. Velikost dávky se lišila podle použité páteřní sítě, vždy však byla nastavena na nejvyšší možnou vzhledem k paměti GPU.

Při předzpracování datasetu byl každý snímek převeden na správnou vstupní velikost. Přesné dimenze se liší dle páteřní sítě, například pro ResNet se jedná o rozměry $224 \times 224 \times 3$, pro model Inception pak $299 \times 299 \times 3$. Změna rozměrů byla realizována pomocí bilineární interpolace. Hodnoty pixelů byly také po načtení převedeny na typ float o velikost 32 bitů. Následně byly snímky augmentovány, s náhodnou pravděpodobností byly provedeny změny kontrastu, jasu, saturace, snížení kvality a zrcadlového otočení.

Součástí předzpracování je také oříznutí snímku tak, aby obličej byl ve středu a snímek obsahoval minimum pozadí. K tomuto je využit algoritmus MTCNN, který lokalizuje obličej ve snímku. K souřadnicím výsledné hraniční oblasti je pak přičteno 32 pixelů ze všech stran a snímek je oříznut. Tímto způsobem byly upraveny snímky pro trénování i snímky pro testování. Oříznutí snímků mělo výrazný vliv na přesnost celého systému. Vizualizace oříznutých snímků je na obrázku 3.1 Naměřené výsledky jsou podrobně rozepsány v kapitole 4.

Jako optimizer byl zvolen algoritmus Adam, velikost trénovacího kroku (*learning rate*) byl nastaven na 0,0002. Při trénování byla na konci každé epochy vyhodnocena úspěšnost modelu na validační části datasetu a v případě zlepšení přesnosti byl uložen checkpoint modelu. Maximální počet epoch byl stanoven na 100, jelikož trénování v pozdějších epochách již nevedlo ke zlepšení přesnosti. Zobrazení vývoje chyby při trénování je na obrázku 3.2.

3.4 Uživatelské rozhraní

Prvním krokem při tvorbě uživatelského rozhraní bylo stanovení požadované funkcionality. Aplikace by měla umožnit uživateli pracovat s databázemi identit, jednotlivé databáze by mělo být možné načítat z disku a zpětně ukládat. Měla by také existovat možnost založit zcela novou prázdnou databázi. Po načtení databáze by měl uživatel mít možnost databázi zobrazit, aplikace by také měla umožnit přidání nového záznamu do databáze z obrazových

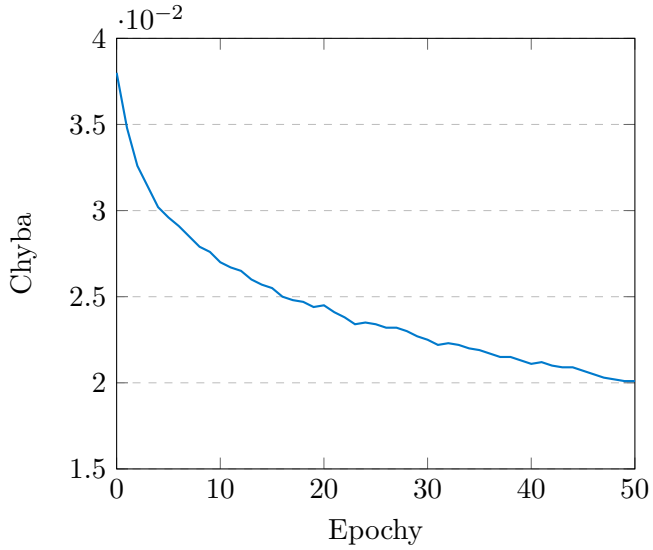
Obrázek 3.1: Zobrazení oříznutí snímků. V prvním a třetím řádku jsou zobrazeny původní neupravené snímky z datasetu Casia Webface – identity 49 a 99. Na druhém a čtvrtém řádku jsou stejné snímky s aplikovanou úpravou oříznutí.

dat. Podobně by mělo mělo být umožněno smazání identity z databáze. Druhou funkcí aplikace by měla být analýza obrazu. Uživatel by měl mít možnost nahrát do aplikace vlastní obrazová data a pomocí neuronové sítě je analyzovat. Embedding vytvořený sítí by pak měl být porovnán s údaji v databázi a uživateli by se měla zobrazit výsledná nalezená identita. Uživateli by také měly být zobrazeny informace o podobnosti nasnímaného obličeje se záznamem v databázi (vzdálenost mezi embeddings).

Pro implementaci aplikace byl zvolen framework PyQt5, ve kterém byla vytvořena jednoduchá demonstrační aplikace. PyQt je meziplatformní toolkit pro tvorbu uživatelských rozhraní. Framework QT byl původně napsán v C++ a je vydáván jak pod svobodnými a open–source licencemi GPL a LGPL, tak pod komerční licencí. Pro jednodušší definici a rozmístění ovládacích prvků aplikace byl využit nástroj QT Creator. Pomocí něj je vytvořen soubor ve formátu *.xml* který specifikuje parametry jednotlivých widgetů. Samotná aplikace následně pouze načte tento soubor a vykreslí GUI. Tímto způsobem bylo oddělena funkční část aplikace od uživatelského rozhraní.

Samotné UI je rozděleno na dvě části, první z nich slouží pro správu databáze, druhá část pro analýzu obrazu. V sekci pro správu databáze je možné načítat a ukládat databáze z disku. Byla také implementována možnost vytvoření nové databáze. Databáze je realizována jako samostatný objekt, který si uchovává informace o identitách. Každá z identit může být také pojmenována. Ukládání a načítání databáze z disku je realizováno pomocí modulu *pickle*. Tento modul umožňuje převádět objekty z jazyka Python do binárních souborů a jejich zpětné načtení. Objekt databáze je tak pomocí modulu jednoduše převeden do bajtového streamu a uložen v momentě, kdy uživatel spustí danou akci. Při načtení je pak zpětně databáze načtená ze souboru do paměti.

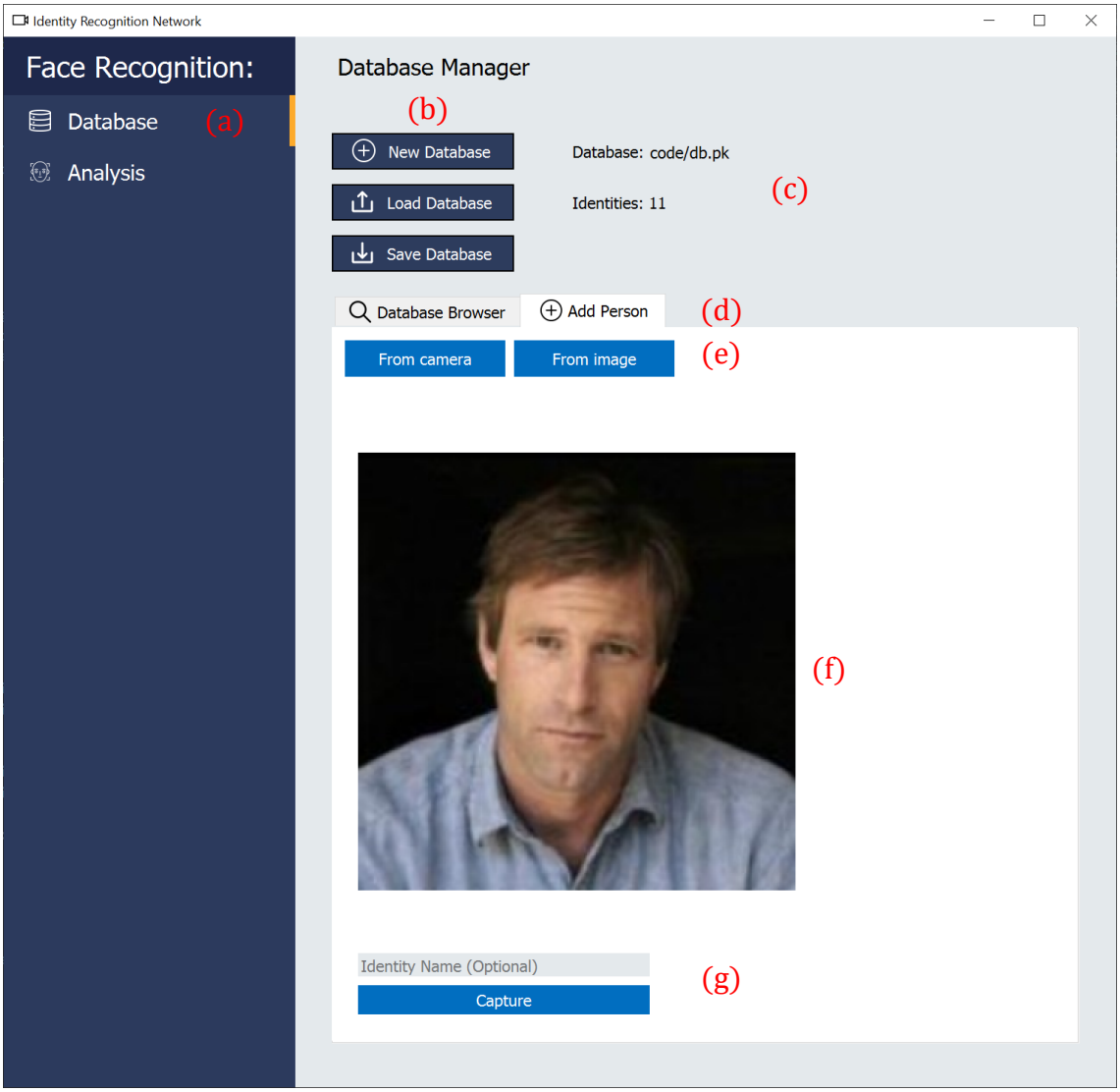
V sekci pro správu databází lze zobrazit záznamy v načtené databázi. Uživateli se také zobrazí informace o celkovém počtu záznamů. Uživatel může do databáze přidávat vlastní



Obrázek 3.2: Vývoj průměrné chyby na trénovacích datech během prvních 50 epoch.

identity (*enrollment*). Toto je možné buď z načteného obrázku nebo z videokamery. Při vkládání identity z kamery je uživateli zobrazena oblast, kam by se měl postavit pro ideální nasnímání. Po nahrání obrazových dat je obličej lokalizován pomocí modulu MTCNN. Tento postup je identický jako u trénování sítě. Výřez je následně zvětšen na rozměry vstupní vrstvy neuronové sítě a z těchto dat je následně vytvořen embedding, který je vložen do databáze. Vložený embedding je možné označit jménem nasnímané osoby.

V sekci pro analýzu jsou vstupní data po nasnímání a vytvoření embedding porovnána se záznamy z databáze a je vybrán nejpodobnější vektor. Uživateli je pak zobrazena nejpodobnější identita z databáze, je také zobrazena informace o vzdálenosti těchto vektorů. Uživatelské rozhraní je zobrazeno na snímku 3.3.



Obrázek 3.3: Vytvořená demonstrační aplikace. Vlevo je menu (a) pro výběr mezi správou databáze a analýzou. Pro ukládání, načítání a tvorbu nové databáze slouží ovládací prvky (b), vedle nich (c) jsou zobrazeny informace o aktuálně načtené databázi. Dále v sekci (d) lze přepínat mezi procházením databáze a přidáváním identit do databáze. V záložce *Database Browser* lze zobrazit identity z aktuálně načtené databáze. Při načítání nové identity jsou vstupní obrazová data načtena z kamery nebo z obrázků (e). Načtený obrázek je zobrazen uprostřed hlavního okna (f). Přidání do databáze je dokončeno pojmenováním osoby a stisknutím tlačítka *Capture* (g).

Kapitola 4

Experimenty a výsledky

Pro testování byl zvolen dataset Labeled Faces in the Wild, který je vědeckou komunitou používán pro porovnávání úspěšnosti jednotlivých modelů. V následujících podkapitolách jsou popsány jednotlivé experimenty, pomocí nichž byl zkoumán vliv úprav algoritmu na přesnost modelu.

4.1 Zarovnání obličeje

Poznatky z některých článků [38] napovídají, že vhodné zarovnání obličeje může mít velký vliv na úspěšnost celého modelu. Některé pokročilé algoritmy [48] pak kromě oříznutí okrajů také vytváří 3D model obličeje, který poté natáčí vhodným způsobem ke kameře. Pro účely této práce byl zkoumán vliv jednoduchého oříznutí oblasti okolo obličeje. Lokalizace obličejů byla provedena pomocí MTCNN. U snímků, kde lokalizace selhala, byl ponechán původní snímek.

Při experimentech byly srovnány dva modely, první z nich byl trénován a testován na původních snímcích, druhý pak na snímcích upravených výše popsanou metodou. Bylo naměřeno, že úpravou snímků stoupla výrazným způsobem přesnost algoritmu z 85 % na 93 %. Z tohoto důvodu byl tento proces zarovnání využit jak pro trénování, tak v samotné aplikaci jako předzpracování snímků.

4.2 Páteřní síť

Pro extrakci příznaků lze použít většinu konvolučních neuronových sítí. V rámci této práce byly porovnány modely s páteřními sítěmi Inception, varianty sítí ResNet, MobileNet, DenseNet a síť typu EfficientNet. U větších sítí bylo nutné snížit velikost trénovací dávky, jelikož se zvýšeným počtem vrstev došlo k vyčerpání paměti na GPU.

Naměřené výsledky v závislosti na páteřní sítí jsou zobrazeny v tabulce 4.1. Je zde patrný trend vyšší přesnosti u jednodušších sítí (například u modelů ResNet). Toto může být způsobeno tím, že menší síť dovoluje využití větší trénovací dávky, ze které je pro síť možné vytvořit více validních trojic identit. Celkově byl naměřený vliv páteřní sítě na přesnost malý. Algoritmus dokázal i při využití malých konvolučních sítí pracovat přesně. Z tohoto důvodu se jako vhodnější varianta jeví menší sítě, které lze trénovat na větší dávce batch a zároveň jsou výrazně rychlejší při následné inferenci.

Páteřní síť	Počet parametrů	Velikost batch	Přesnost na LFW
InceptionV3	24M	128	$0,926 \pm 0,01$
Resnet50V2	25M	128	$0,918 \pm 0,01$
Resnet101V2	44M	80	$0,903 \pm 0,01$
Resnet152V2	60M	48	$0,883 \pm 0,02$
MobileNetV3Small	2,7M	240	$0,940 \pm 0,01$
MobileNetV3Large	5M	192	$0,937 \pm 0,01$
DenseNet121	8M	88	$0,928 \pm 0,01$
DenseNet169	14M	80	$0,920 \pm 0,03$
InceptionResnetV2	56M	64	$0,924 \pm 0,01$
EfficientNetB3	12M	32	$0,902 \pm 0,01$

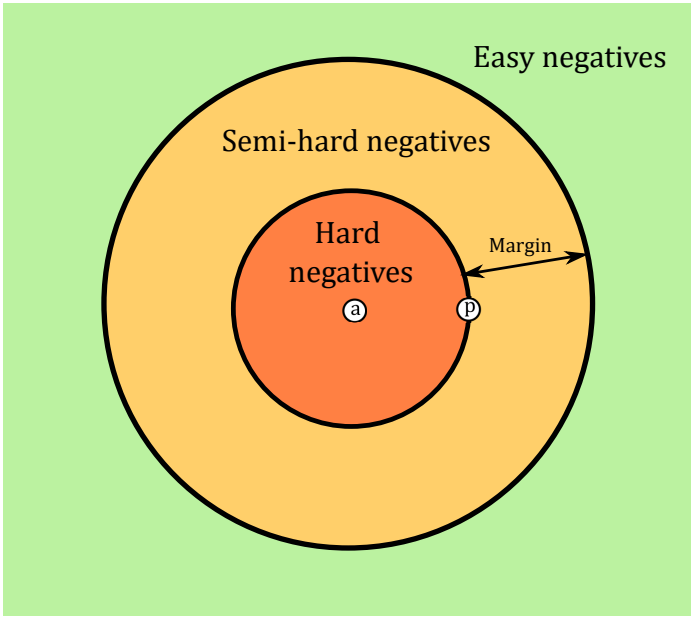
Tabulka 4.1: Porovnání přesnosti sítí v závislosti na použité páteřní síti. Pro větší modely bylo nutné snížení velikosti batch, což nejspíše vedlo ke snížené přesnosti u větších modelů, které by jinak teoreticky měly být přesnější. Toto je patrné při porovnání modelů ResNet, kdy rozsáhlejší modely vykázaly horší přesnost. Kromě velikosti batch byly sítě trénovány s identickými hyperparametry.

4.3 Chybové funkce

Správný výběr chybové funkce může pomoci síti s tvorbou vhodných příznakových vektorů. V rámci této práce byl zkoumán vliv chybové funkce na přesnost modelu. Byly porovnány čtyři varianty chybové funkce.

První ze zkoumaných byla funkce Triplet loss. Při trénování bylo v každém cyklu vytvořeno pole obsahující vyextrahované vektory. Velikost tohoto pole odpovídá velikosti batch. Poté byly mezi těmito vektory nalezeny vhodné trojice, tedy ty trojice, kde dvě identity patří stejné osobě a třetí identita patří jiné osobě. U těchto trojic pak byla vypočítána vzájemná euklidovská vzdálenost vektorů a byla vypočtena chyba dle vztahu popsaném v kapitole 2.3.2. U této varianty nebyly zvlášť voleny náročné trojice, chybovost se pouze vypočítá ze všech možných trojic v trénovací dávce, které generují pozitivní chybu.

Další variantou byla tzv. Semi–Hard Triplet loss. Tato funkce oproti naivní implementaci modifikuje výběr trojic tak, aby byl větší důraz kladen na obtížné trojice, tedy na ty, které se nejvíce podílejí na tvorbě chyby. Semi–Hard Triplet loss tedy zavádí navíc tzv. *semi–hard negative mining*, tedy výběr vhodného těžkého negativního snímku. Tyto identity lze jednoduše nalézt dle jejich vzdálenosti v euklidovském prostoru od snímku *anchor*. Funkce pro negativní identity volí ty, které jsou dostatečně daleko, aby přispívaly k chybě, ale zároveň nejsou blíže, než pozitivní identita (částečně obtížné snímky jsou zobrazeny na obrázku 4.1). Tato funkce nevybírá těžké pozitivní identity, protože toto může vést k situacím, kdy se špatně označené snímky z datasetu budou vyskytovat v pozitivní identitě a síť při učení z těchto vzorků bude divergovat. Zobrazení těžkých, částečně těžkých a triviálních negativních dvojic je zobrazen na obrázku 4.1.



Obrázek 4.1: Zobrazení obtížných, částečně obtížných a triviálních trojic pro funkci Triplet loss. Na snímku pozice (a) značí *anchor* a (p) značí pozitivní identitu. Pokud je negativní identita blíže než pozitivní, jedná se o obtížnou trojici. Pokud je negativní vzorek vzdálen více než pozitivní vzorek, ale méně než je hodnota *margin*, jedná se o částečně obtížný případ. Tyto trojice využívá semi–hard triplet loss. Jako triviální trojice lze označit ty, kde vzájemná pozice vzorků negeneruje žádnou chybu.

Třetí zkoumanou funkcí byla varianta, kdy namísto vzdálenosti v euklidovském prostoru byl pro porovnání vzdálenosti využíván úhel mezi jednotlivými embeddings. Každý embedding si lze představit, jako vektor udávající pozici v N rozměrném prostoru (kde N odpovídá počtu dimenzí vektoru, v této práci typicky 128). Pro výpočet úhlu mezi vektory $\overrightarrow{u}$ a $\overrightarrow{v}$ lze pak použít vztah:

$$\cos\alpha = \frac{\overrightarrow{u}\cdot\overrightarrow{v}}{|\overrightarrow{u}|\cdot|\overrightarrow{v}|} = \frac{u_1v_1 + u_2v_2 + \cdots + u_Nv_N}{\sqrt{u_1^2 + u_2^2 + \cdots + u_N^2}\cdot\sqrt{v_1^2 + v_2^2 + \cdots + u_N^2}} \tag{4.1}$$

Čtvrtou variantou byla kombinace předchozích dvou, tedy kombinace úhlového rozdílu vektorů s trénováním z částečně náročných trojic. Při porovnání všech chybových funkcí se ukázala jako nejpřesnější funkce s výběrem náročných negativů a s vzdálenostní metrikou zohledňující úhel mezi vektory. Výsledky jsou shrnuty v tabulce 4.2.

Chybová funkce:	**Přesnost**
Všechny validní trojice + euklidovská vzdálenost	93,4 %
Všechny validní trojice + úhlová vzdálenost	92,9 %
Semi–hard trojice + euklidovská vzdálenost	94,0 %
Semi–hard trojice + úhlová vzdálenost	95,2 %

Tabulka 4.2: Vliv chybové funkce na přesnost algoritmu. Jako neúspěšnější se projevila varianta s výběrem semi–hard trojic a s výpočtem vzdálenosti jako úhly mezi vektory.

4.4 Augmentace

Pro zvýšení variability datasetu byl proveden i experiment s augmentací snímků. Na každé z fotografií byly náhodně provedeny úpravy kontrastu, jasu, saturace a snížení kvality snímku. Každý snímek byl také s pravděpodobností 50 % zrcadlově otočen. Vizualizace augmentací je zobrazena na snímku 4.2. V rámci experimentu byly natrénovány dvě sítě s identickými parametry, jedna využívající augmentací datasetu a druhá s původními snímky.

Experiment ukázal, že augmentace snímků nevedla ke zlepšení systému. Síť trénovaná na augmentovaných fotografiích dosáhla přesnosti 93,31 %, bez augmentací pak 93,71 %. Dataset je zřejmě dostatečně rozsáhlý na to, aby vyžadoval dodatečnou augmentaci vstupních dat a sítě jsou schopné se naučit pracovat s drobnými variacemi osvětlení ze surových dat. Pro další vývoj tedy nebyly augmentované snímky použity.

(a) (b) (c) (d) (e) (f)

Obrázek 4.2: Vizualizace augmentací použitých pro experimenty. Na snímku (a) je zobrazen původní snímek bez úprav. Ve snímku (b) je ukázka úpravy saturace. Snímek (c) zobrazuje úpravu jasu, snímek (d) obsahuje změnu kontrastu a snímek (e) degradaci kvality (v důsledku vyšší komprimace formátu jpeg). Na snímku (f) je pak zobrazeno zrcadlové otočení. Všechny tyto úpravy byly v experimentu použity s náhodně zvolenou intenzitou, každý snímek tak mohl být ovlivněn větším množstvím augmentací.

4.5 Velikost embedding

Další z experimentů se zaměřoval na vliv velikosti výstupního vektoru (embedding) na přesnost systému. Předchozí výzkum naznačuje [38], že velikost nad 128 obvykle nevede ke zvýšení přesnosti. Pro otestování této teorie byly natrénovány sítě s různými velikostmi embedding, kromě původních 128 také 256 a 512.

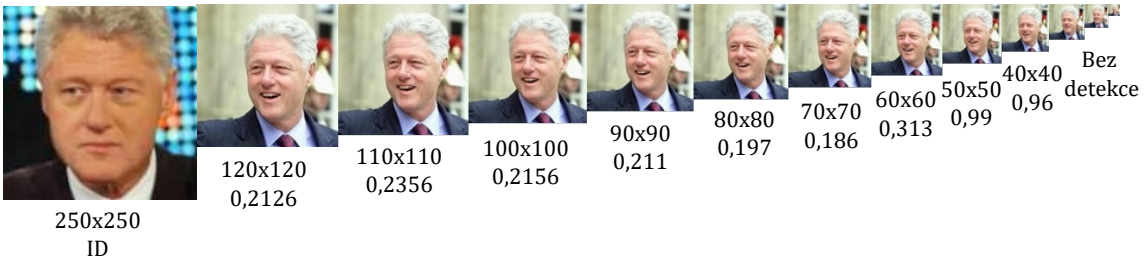
Naměřené výsledky potvrdily předpoklady, vyšší velikost embedding nepřispěla k přesnosti sítě. Síť s embedding o velikosti 128 dosáhla 94,1 %, s 256 94,2 % a s 512 93,8 %. Z tohoto důvodu byl pro další vývoj ponechán výstupní rozměr 128, který je zároveň dostatečně malý, aby nezpůsoboval problémy u databází s větším počtem identit.

4.6 Vliv vzdálenosti na přesnosti algoritmu

Častým problémem identifikačních systémů je vyšší chybovost při větší vzdálenosti mezi kamerou a snímanou osobu. Se vzdáleností přirozeně klesá rozlišení relevantních vstupních dat, které musí být následně pro neuronovou síť zvětšeny. Při nízkém rozlišení však nemusí snímek obsahovat dostatečné množství dat pro úspěšnou identifikaci. Chyba může nastat jak při identifikaci osoby, tak při předcházející lokalizaci osoby pro zarovnání obličeje. Jelikož jsou identifikační systémy často používány jako nadstavba existujících kamerových systémů,

nelze tyto situace zcela eliminovat. V tomto experimentu bylo zkoumáno, jak rychle roste chyba identifikačního systému (vzdálenost identifikačních vektorů) v závislosti na rozlišení vstupního snímku.

Pro experiment byl přidán jeden snímek v plném rozlišení jako identita osoby. Poté byla měřena vzdálenost mezi touto identitou a výstupním vektorem generovaným nad jiným snímkem stejné osoby. Rozlišení této druhé fotografie bylo postupně snižováno. Měření ukázalo, že systém byl schopný tuto osobu správně identifikovat do rozlišení 60×60 pixelů. Při této velikosti snímku zabírá obličej asi 25×25 pixelů. U nižšího rozlišení nebyl systém provést správnou identifikaci. Od rozlišení 30×30 pixelů pak také selže detektor obličeje (MTCNN). Naměřené výsledky jsou zobrazeny ve snímku 4.3.

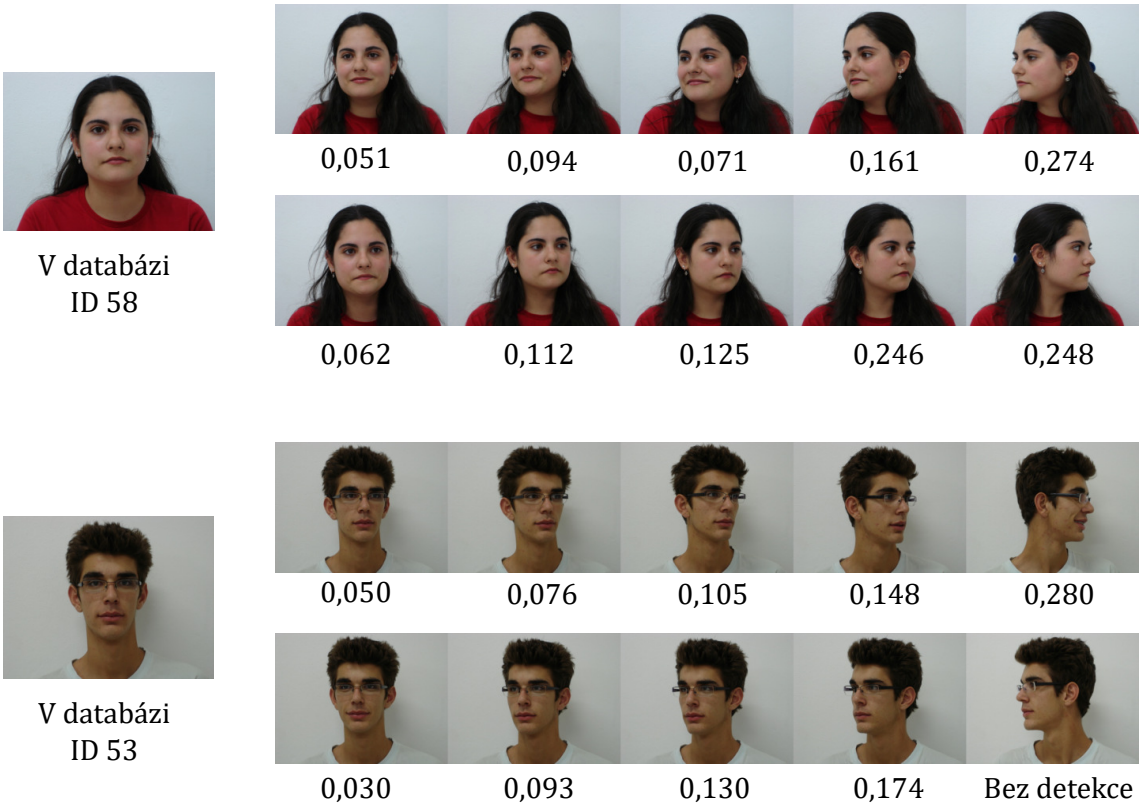


Obrázek 4.3: Závislost rozlišení na schopnosti systému provádět identifikace osoby. Na levém snímku je fotografii použitá pro vytvoření identifikačního vektoru. S ním byly porovnány vektory vytvořené ze snímků se snižujícím se rozlišením. První řádek pod fotografií udává rozlišení snímku, druhý údaj je vzdálenost mezi tímto vektorem a identitou. Zhoršení rozpoznávacích schopností je patrný od velikosti 60×60 pixelů. Pro rozlišení 30×30 a nižší pak dojde také k selhání detektoru (MTCNN).

4.7 Vliv natočení na přesnost algoritmu

Jedním z požadavků na systém je schopnost provádět identifikace dostatečně robustně i na snímcích, kdy osoba není natočená čelem do kamery. Pro zhodnocení splnění tohoto požadavku byl proveden experiment na snímcích z datasetu FEI Face Database 3.1.8. Tento dataset obsahuje osoby vyfocené v kontrolovaných podmínkách s variabilním natočením ke kameře a je tak vhodný pro tento účel.

Do databáze byl přidán snímek s osobou dívající se čelně do kamery (varianta s neutrálním výrazem) a byla měřena vzdálenost vektorů rysů mezi touto fotografií a ostatními snímky dané osoby. Očekávaný výsledek byl zhoršení přesnosti (větší vzdálenost mezi vektory) s vyšším natočením. Experiment toto potvrdil, systém vykazoval vyšší rozdíl vektorů v závislosti na natočení. Identifikace však byla možná i u snímků s vysokým úhlem natočení. Z databáze osob byla u všech testovaných snímků vybrána správná identita. Vizualizace výsledků je na snímku 4.4.



Obrázek 4.4: Vizualizace schopnosti sítě identifikovat osoby s různým úhlem natočení ke kameře. Vlevo jsou fotografie dvou osob, ze kterých byly vytvořeny vektory příznaků vložené do databáze. V pravé části je pak pro každou osobu 10 snímků s různým natočením ke kameře, číslo pod snímkem udává vzdálenost mezi identifikačními vektory. Systém byl schopný v databázi vyhledat správnou osobu i s vysokým úhlem natočení. Vzdálenost vektorů se však s vyšším úhlem zvyšuje. Toto je nejvíce patrné u natočení o 90°, kde se vzdálenost blíží hodnotě 0,3. U jedné fotografie pak selhalo zarovnání detektorem MTCNN.

4.8 Naměřené výsledky

Na základě znalostí získaných z experimentů byly zvoleny vhodné hyperparametry a způsob trénování pro konečnou neuronovou síť. Při trénování byl zahrnut krok s oříznutím obličeje popsaný v sekci 4.1. Tato úprava měla nejvyšší naměřený vliv na přesnost systému a její využití je tak více než vhodné. Jako páteřní síť byla zvolena architektura MobileNet, která dosáhla v experimentu nejlepšího skóre. Vliv rozdílných páteřních sítí na přesnost byl spíše malý a využití menšího modelu přináší výhody v podobě rychlejší inference. Pro trénování byla využita chybová funkce triplet loss porovnávající úhel mezi výstupními vektory využívající semi–hard trojice vzorků z datasetu. Pro validaci byl pak rozdíl měřen mezi všemi validními trojicemi. Augmentace snímků nebyla využita, jelikož neprokázala zlepšení přesnosti. Velikost výstupu byla omezena na 128 čísel s plovoucí desetinou čárkou, vyšší velikost embedding nepřinesla žádné zlepšení.

Přesnost výsledného modelu byla ověřena na datasetu LFW. Ten je pro testování vhodný ze dvou důvodů. Prvním je jeho velmi rozsáhlé využití ve výzkumných článcích. Dataset je běžně chápán jako benchmark pro úlohu identifikace a je tedy snadné porovnat tento

model s ostatními. Druhým důvodem je variabilita snímků v datasetu. Ty jsou zachyceny v různorodých podmínkách, obličeje jsou také vyfoceny s různým natočením. Jedním z úkolů této práce je ověření robustnosti modelu a schopnosti pracovat s různým natočením obličeje ke kameře a tento dataset je tak vhodný pro testování modelu v tomto směru.

Autoři datasetu poskytují několik možných protokolů testování. Modely, které při vývoji dodrží pravidla stanovená těmito protokoly, se pak mohou umístit do žebříčku podle úspěšnosti. Jednotlivé protokoly se liší podle dovoleného způsobu práce s daty z datasetu a s externími daty, jejich podrobný popis lze dohledat v [19] a [27]. Jedná se o protokoly:

- Unsupervised
- Image–restricted with no outside data.
- Unrestricted with no outside data.
- Image–restricted with label–free outside data.
- Unrestricted with label–free outside data.
- Unrestricted with labeled outside data.

Při protokolu **unsupervised**, tedy při učení bez učitele nemá algoritmus přístup k identitám osob (*labels*). Stejně tak nemůže mít algoritmus informace o rozložení identit v datasetu.

Rozdílný přístup k práci s trénovací částí datasetu je definován pojmy **image–restricted** a **unrestricted**. U image–restricted protokolů není možné při trénování tvořit jiné dvojice identit, než ty, které definovali autoři datasetu. U unrestricted protokolů toto omezení neplatí a při trénování tak lze tvořit libovolné množství dvojic snímků, u kterých pak algoritmus posuzuje jejich shodu.

Druhé pravidlo se týká práce s externími daty. Protokoly s pravidlem **no outside data** nesmí využívat pro trénování žádné externí snímky, trénovat lze pouze na datech poskytnutých v rámci datasetu LFW. Pravidlo platí také pro algoritmy, které využívají zarovnání obličeje jako předpřípravu samotné identifikace. Takový algoritmus také nesmí být trénován na externích datech. Protokoly s pravidlem **label–free outside data** mohou využít dodatečných snímků obličeje, výřezů fotografií nebo další data pocházející mimo dataset LFW. Tato data však nesmí být anotována. Podobně jsou zakázány snímky z videí, protože zde by bylo pro algoritmus snadné odvodit identitu pomocí trackování osoby. Protokoly s pravidlem **labeled outside data** nekladou žádné limity na využití externích dat.

Z těchto pravidel lze odvodit protokol pro testování modelu vytvořeného v této práci. Ten pro trénování nevyužíval nově generovaných dvojic z datasetu LFW, spadá tak do kategorie *image–restricted*. Dále pak pro testování byla využita externí anotovaná data, konkrétně anotovaný dataset Casia–Webface. Z tohoto důvodu spadá do kategorie *labeled outside data*. Jelikož však protokol *image–restricted with no outside data* není oficiálně podporován autory žebříčku LFW, musí být tento typ trénování zařazen do nejširší kategorie *unrestricted with labeled outside data*. Tato kategorie je nejpoužívanější u výzkumných článků, přímé srovnání modelů je však komplikováno výrazně rozdílnou kvalitou dat, které mají výzkumníci k dispozici. Dataset Casia–Webface, který byl využit pro tuto práci, se řadí spíše mezi menší datasety využité pro tuto kategorii. Například algoritmus Facenet [38] jako externí data využil dataset o velikosti 100 až 200 milionů snímků. Casia–Webface oproti tomu obsahuje pouze 500 000 snímků.

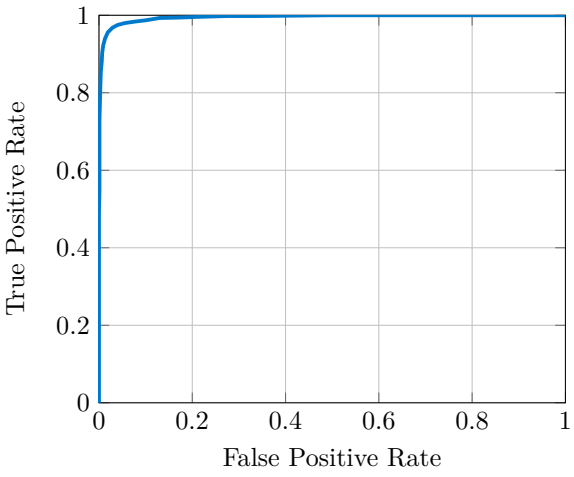
Samotný postup testování je blíže popsán na webových stránkách datasetu LFW[1]. Dvojice snímků, nad kterými algoritmus rozhoduje o identitě, jsou předem pevně dané. Autoři datasetu také poskytují předpřipravené skripty na evaluaci přesnosti a výpočtu křivky ROC a výpočtu AUC. Pro účely této práce byly výsledky měřeny již existujícím skriptem v jazyce python.

Měření výsledného modelu vykázalo přesnost 97,08 %. Pro dataset LFW je přesnost počítána jako podíl součtu skutečně negativních a skutečně pozitivních predikcí k celkovému počtu predikcí:

$$\frac{TP + TN}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{4.2}$$

ROC křivka pro tento model je zobrazena na snímku 4.5.

Tento výsledek je srovnatelný s ostatními algoritmy, nejlepší z modelů využívající stejný protokol se pohybují mezí 97 % a 99,5 %. Rozdíl v přesnosti od nejlepších modelů lze přisuzovat několika faktorům. Prvním z nich je relativně malý dataset, na kterém byl model trénován. Současné *state–of–the–art* modely jsou většinou trénovány na řádově větších datasetech, ze kterých lze vytvořit o několik řádů více trojic pro trénování. Je velmi pravděpodobné, že při využití dat s podobným rozsahem by přesnost algoritmu stoupala. Druhým faktorem může být hardwarová limitace. Experimenty provedené v této práci naznačují, že větší dávka batch vede k lepším výsledkům, jelikož je možné z větší dávky vytvořit více obtížných trojic. V této práci však byla největší možná trénovací dávka nastavena na 240 (u menší verze MobileNet) a v nejmenším případě na 32 (u sítě EfficientNet). Z výzkumu však vyplývá, že pro trénování některých modelů [38] byla trénovací dávka stanovena i na několik tisíc snímků, pro efektivnější online tvorbu obtížných tripletů. K tomu bylo využito trénování na clusterech CPU, které byly schopny pracovat s větší dávkou batch. V rámci této práce však podobné hardwarové zdroje nebyly k dispozici a velikost batch byla limitována velikostí paměti GPU na řádově nižších hodnotách.



Obrázek 4.5: Modrou barvou je zobrazena ROC křivka pro nejlepší natrénovaný model s páteřní sítí mobilenet. AUC je pro tento model rovno 99,5 %, EER 3,1 %.

[1] http://vis-www.cs.umass.edu/lfw

4.8.1 Porovnání s lidskou přesností

Vyjádření přesnosti algoritmu pomocí procentuální hodnoty není příliš reprezentativní, lepší metrikou je srovnat chybovost modelu s chybovostí lidského řešitele. V práci z roku 2009 [25] byla zkoumána přesnost lidí na datasetu LFW při identifikaci stejných a rozdílných dvojic fotografií.

Výzkumníci nechali různé osoby (v průměru deset na každý pár fotografií) hodnotit dvojice snímků. Pro snímky v nezměněném formátu byla naměřena přesnost 99,2 %, tedy velmi vysoké skóre, překonané pouze několika málo systémy. Při dalším zkoumání bylo však zjištěno, že lidé pro identifikaci do velké míry spoléhají na pozadí u snímků. Informace z pozadí pak lidem pomáhají s identifikací. Při druhém testu byly proto obličeje osob oříznuty tak, aby nezahrnovaly pozadí snímků. Tato varianta je také nejpodobnější způsobu, jakým s fotografiemi pracuje model z této práce. Na oříznutých snímcích pak byla lidská přesnost 97,5 %, tedy mírně lepší výsledek, než model z této práce.

Skutečnost, že pozadí a kontext snímku pomáhá lidem s identifikací byl ověřen třetím testem. V něm byla maska invertovaná vůči druhému testu, tedy lidem bylo zobrazeno pouze pozadí bez obličeje. Překvapivě i s těmito fotografiemi byli lidé schopni dosáhnout přesnosti 94,2 %, což potvrzuje, že lidé při identifikaci pracují s širším spektrem informací. Za zmínku stojí fakt, že zatímco lidé se s širším záběrem okolí v identifikaci zlepšují, pro algoritmy strojového učení toto neplatí. Na datasetu LFW je patrný trend vyšší přesnosti u oříznutých fotografií.

Kapitola 5

Závěr

Cílem této práce bylo nastudovat problematiku rozpoznávání osob na základě snímků obličeje, prostudovat současné moderní trendy a zjistit, jakými nedostatky trpí. Na základě těchto znalostí pak sestrojit systém schopný provádět rozpoznávání osob podle obličeje na fotografiích nasnímaných z různých úhlů natočení. Navržený systém pak implementovat s jednoduchým uživatelským rozhraním, celý systém otestovat a provést experimenty zaměřené na zhodnocení úspěšnosti identifikace.

V práci je popsán úvod do biometrie, podrobně je vysvětleno fungování neuronových sítí a proces jejich trénování. Dále je v práci shrnut současný stav výzkumu v oblasti identifikace osob podle obličeje a jsou zde popsány *state–of–the–art* metody, které se pro tento účel využívají. Samostatně jsou také popsány datasety, které je možné pro trénování těchto algoritmů využít.

Na základě znalosti problematiky byl vytvořen algoritmus založený na neuronových sítích. Pro extrakci příznaků ze snímků je využita konvoluční síť z rodiny MobileNet. Za ní jsou následně umístěny nové vrstvy starající se o vytvoření identifikačního vektoru osoby. Síť byla trénována pomocí funkce triplet loss, díky které jsou výsledné vektory jedné osoby blíže, než vektory jiných osob. Po jejich porovnání je tak možné přímo zjistit podobnost dvou obličejů. Síť byla pro tento účel trénována na datasetu Casia–Webface.

Pro zlepšení modelu byla provedena řada experimentů, které měly za účel najít vhodnou architekturu sítě, odhalit vhodné hyperparametry a vyladit proces trénování. Na základě těchto experimentů byl celý systém vyladěn a následně otestován na datasetu Labeled Faces in the Wild, kde byl naměřen výsledek 97,08 % s využitím protokolu neomezujícím práci s externími daty. Schopnost rozpoznávat osoby z různým natočením ke kameře pak byla ověřena na datasetu FEI Face Database.

Pro demonstraci systému byla vytvořena desktopová aplikace, která umožňuje snadnou vizualizaci dosažených výsledků. Uživateli umožňuje načítat a ukládat databáze osob na disk. Lze také přidávat do databáze nové osoby na základě nahraných snímků nebo z webkamery. Dále je možné vyhledat nejpodobnější osobu z databáze a vyčíslit jejich podobnost.

Jako přínos této práce lze považovat fakt, že výsledků bylo dosaženo za využití otevřených dat a algoritmů. Mnoho systémů na identifikaci osob využívá pro trénování modelů proprietární data, která nejsou přístupná veřejnosti. Běžné je také použití proprietárních algoritmů pro zarovnání a oříznutí obličeje nebo také trénování na speciálním hardware. Algoritmus vytvořený v rámci této práce však využívá volně dostupné metody a data a zároveň dosahuje srovnatelných výsledků, jako současné *state–of–the–art* modely.

Literatura

[1] Beitz, A. J. a Fletcher, T. F. *Lab 1: Nervous Tissue Histology, Neurons – Multipolar Neurons*. 2020. [online], [cit. 2021-05-09]. Dostupné z: http://vanat.cvm.umn.edu/neurLab1/neuron.html.

[2] Cao, Q., Shen, L., Xie, W., Parkhi, O. M. a Zisserman, A. VGGFace2: A dataset for recognising faces across pose and age. *CoRR*. 2017, abs/1710.08092, [cit. 2020-12-14]. Dostupné z: http://arxiv.org/abs/1710.08092.

[3] Chollet, F. et al. *Deep learning with Python*. 1. vyd. Manning New York, 2018. ISBN 9781617294433.

[4] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K. a Bengio, Y. Attention-Based Models for Speech Recognition. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M. a Garnett, R., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, sv. 28, s. 577–585. Dostupné z: https://proceedings.neurips.cc/paper/2015/file/1068c6e4c8051cfd4e9ea8072e3189e2-Paper.pdf.

[5] Dalal, N. a Triggs, B. Histograms of oriented gradients for human detection. In: IEEE. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. 2005, sv. 1, s. 886–893.

[6] Deng, J., Guo, J. a Zafeiriou, S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *CoRR*. 2018, abs/1801.07698, [cit. 2021-02-15]. Dostupné z: http://arxiv.org/abs/1801.07698.

[7] Fukushima, K. a Miyake, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: *Competition and cooperation in neural nets*. Springer, 1982, s. 267–285. ISSN 0262-8856.

[8] Georghiades, A., Belhumeur, P. a Kriegman, D. From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*. 2001, sv. 23, č. 6, s. 643–660.

[9] Girshick, R. B. Fast R-CNN. In:. 2015, abs/1504.08083 [cit. 2020-11-29]. Dostupné z: http://arxiv.org/abs/1504.08083.

[10] Girshick, R. B., Jeff Donahue, T. D. a Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*. 2013, abs/1311.2524, [cit. 2021-02-20]. [online]. Dostupné z: http://arxiv.org/abs/1311.2524.

[11] Glorot, X. a Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In: JMLR Workshop and Conference Proceedings. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, s. 249–256.

[12] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. ISBN 0262035618, 9780262035613. http://www.deeplearningbook.org.

[13] Grgic, M., Delac, K. a Grgic, S. SCface–surveillance cameras face database. *Multimedia tools and applications*. Springer. 2011, sv. 51, č. 3, s. 863–879.

[14] He, K., Zhang, X., Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. *CoRR*. 2015, abs/1512.03385, [cit. 2021-02-03]. Dostupné z: http://arxiv.org/abs/1512.03385.

[15] He, K., Zhang, X., Ren, S. a Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE international conference on computer vision*. 2015, s. 1026–1034.

[16] Howard, A., Sandler, M., Chu, G., Chen, L., Chen, B. et al. Searching for MobileNetV3. *CoRR*. 2019, abs/1905.02244, [cit. 2021-03-19]. Dostupné z: http://arxiv.org/abs/1905.02244.

[17] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*. 2017, abs/1704.04861, [cit. 2021-03-19]. Dostupné z: http://arxiv.org/abs/1704.04861.

[18] Huang, G., Liu, Z., Maaten, L. van der a Weinberger, K. Q. *Densely Connected Convolutional Networks*. 2018 [cit. 2021-01-17].

[19] Huang, G. B., Mattar, M., Berg, T. a Learned Miller, E. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In: *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*. 2008.

[20] Huang, G. B., Ramesh, M., Berg, T. a Learned Miller, E. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. 07-49. University of Massachusetts, Amherst, October 2007.

[21] Jain, A. K., Flynn, P. a Ross, A. A. *Handbook of biometrics*. 1. vyd. Springer Science & Business Media, 2007. ISBN 978-0-387-71040-2.

[22] Jain, A. K., Ross, A. A. a Nandakumar, K. *Introduction to biometrics*. 1. vyd. Springer Science & Business Media, 2011. ISBN 978-0-387-77325-4.

[23] Krizhevsky, A., Sutskever, I. a Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F., Burges, C. J. C., Bottou, L. a Weinberger, K. Q., ed. *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, s. 1097–1105. [online], [cit. 2020-11-29]. Dostupné z: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[24] Krizhevsky, A., Sutskever, I. a Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F., Burges, C. J. C., Bottou, L. a Weinberger, K. Q., ed. *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, s. 1097–1105 [cit. 2020-12-10]. [online]. Dostupné z: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[25] Kumar, N., Berg, A. C., Belhumeur, P. N. a Nayar, S. K. Attribute and simile classifiers for face verification. In: IEEE. *2009 IEEE 12th international conference on computer vision*. 2009, s. 365–372.

[26] Kumar, S. K. On weight initialization in deep neural networks. *CoRR*. 2017, abs/1704.08863, [cit. 2020-12-18]. Dostupné z: http://arxiv.org/abs/1704.08863.

[27] Learned Miller, G. Labeled faces in the wild: Updates and new reporting procedures. *University of Massachusetts, Amherst, Tech. Rep. UM-CS-2014-003*. 2014.

[28] LeCun, Y., Bottou, L., Bengio, Y. a Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. Ieee. 1998, sv. 86, č. 11, s. 2278–2324.

[29] Liu, Z., Luo, P., Wang, X. a Tang, X. Deep Learning Face Attributes in the Wild. In: *Proceedings of International Conference on Computer Vision (ICCV)*. December 2015.

[30] McGonagle, J., Shaikouski, G., William, C., Hsu, A., Khim, J. et al. *Backpropagation, Brilliant.org*. 2020 [cit. 2020-10-08]. [online]. Dostupné z: https://brilliant.org/wiki/backpropagation/.

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

[32] Mishkin, D. a Matas, J. *All you need is a good init*. 2016.

[33] Ren, S., He, K., Girshick, R. B. a Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*. 2015, abs/1506.01497, [cit. 2021-02-20]. [online]. Dostupné z: http://arxiv.org/abs/1506.01497.

[34] Sabour, S., Frosst, N. a Hinton, G. E. *Dynamic Routing Between Capsules*. 2017 [cit. 2021-01-28]. Dostupné z: http://arxiv.org/abs/1710.09829.

[35] Sambasivarao, K. *Non-maximum Suppression (NMS) – A technique to filter the predictions of object detectors*. 2019 [cit. 2021-05-09]. [online]. Dostupné z: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c.

[36] Sánchez, J. a Perronnin, F. High-dimensional signature compression for large-scale image classification. In: IEEE. *CVPR 2011*. 2011, s. 1665–1672.

[37] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A. a Chen, L. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. *CoRR*. 2018, abs/1801.04381, [cit. 2021-03-19]. Dostupné z: http://arxiv.org/abs/1801.04381.

[38] Schroff, F., Kalenichenko, D. a Philbin, J. FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. Jun 2015, [cit. 2020-12-27]. DOI: 10.1109/cvpr.2015.7298682. Dostupné z: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[39] Shi, Y., Jain, A. K. a Kalka, N. D. Probabilistic Face Embeddings. *CoRR*. 2019, abs/1904.09658, [cit. 2021-03-05]. Dostupné z: http://arxiv.org/abs/1904.09658.

[40] Simonyan, K. a Zisserman, A. Very deep convolutional networks for large-scale image recognition. *ArXiv preprint arXiv:1409.1556*. 2014.

[41] Simonyan, K. a Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015.

[42] Skorski, M., Temperoni, A. a Theobald, M. *Revisiting Initialization of Neural Networks*. 2020.

[43] Sun, Y., Wang, X. a Tang, X. *Deeply learned face representations are sparse, selective, and robust*. 2014.

[44] Sun, Y., Cheng, C., Zhang, Y., Zhang, C., Zheng, L. et al. Circle loss: A unified perspective of pair similarity optimization. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, s. 6398–6407.

[45] Svoboda, J. *Detektor hlavy v obraze*. Brno 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Goldmann.

[46] Szegedy, C., Ioffe, S. a Vanhoucke, V. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *CoRR*. 2016, abs/1602.07261, [cit. 2021-03-26]. Dostupné z: http://arxiv.org/abs/1602.07261.

[47] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. et al. *Going Deeper with Convolutions*. 2014.

[48] Taigman, Y., Yang, M., Ranzato, M. a Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, s. 1701–1708.

[49] Tan, M., Chen, B., Pang, R., Vasudevan, V. a Le, Q. V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *CoRR*. 2018, abs/1807.11626, [cit. 2021-03-19]. Dostupné z: http://arxiv.org/abs/1807.11626.

[50] Thomaz, C. E. a Giraldi, G. A. A new ranking method for principal components analysis and its application to face image analysis. *Image and Vision Computing*. 2010, sv. 28, č. 6, s. 902–913. DOI: https://doi.org/10.1016/j.imavis.2009.11.005. ISSN 0262-8856. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0262885609002613.

[51] Van Der Malsburg, C. Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. In: Palm, G. a Aertsen, A., ed. *Brain Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, s. 245–248. ISBN 978-3-642-70911-1.

[52] Viola, P. a Jones, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001.* 2001, sv. 1, s. I–I.

[53] Volná, E. *Neuronové sítě 1.* 2008 [cit. 2020-11-25]. [online]. Dostupné z: https://web.osu.cz/~Volna/Neuronove_site_skripta.pdf.

[54] Wolf, L., Hassner, T. a Maoz, I. Face recognition in unconstrained videos with matched background similarity. In: IEEE. *CVPR 2011.* 2011, s. 529–534.

[55] Xu, B., Wang, N., Chen, T. a Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *CoRR.* 2015, abs/1505.00853, [cit. 2020-10-09]. Dostupné z: http://arxiv.org/abs/1505.00853.

[56] Yi, D., Lei, Z., Liao, S. a Li, S. Z. Learning Face Representation from Scratch. *CoRR.* 2014, abs/1411.7923, [cit. 2020-12-14]. Dostupné z: http://arxiv.org/abs/1411.7923.

[57] Zbořil, F. V. *Soft Computing 4 – Neocognitron a konvoluční neuronové sítě.* 2020. [online], [cit. 2020-11-28].

[58] Zeiler, M. D. a Fergus, R. *Visualizing and Understanding Convolutional Networks.* 2013. Dostupné z: http://arxiv.org/abs/1311.2901.

[59] Zhang, K., Zhang, Z., Li, Z. a Qiao, Y. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. *CoRR.* 2016, abs/1604.02878. Dostupné z: http://arxiv.org/abs/1604.02878.

[60] Zhou a Chellappa. Computation of optical flow using a neural network. In: *IEEE 1988 International Conference on Neural Networks.* 1988, s. 71–78 vol.2. DOI: 10.1109/ICNN.1988.23914.

Příloha A

Obsah přiloženého paměťového nosiče

`README.txt` – Obsahuje podrobnější popis poskytnutých souborů a návod na jejich použití.
`/report` – Obsahuje text práce.
    `/src` – Zdrojové texty v LaTeXu.
`/bin` – Obsahuje přeložený binární soubor aplikace s knihovnami.
`/src` – Adresář se zdrojovými kódy programu.
    `/checkpoints` – Adresář s uloženým modelem sítě.
    `/doc` – Obsahuje technickou dokumentaci aplikace.
    `/external` – Obsahuje převzaté části kódu.
    `/db.pk` – Databáze s několika nahranými identitami.
    `/img` – Obsahuje několik snímků z datasetů LFW a FEI database.