VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HLUBOKÉ NEURONOVÉ SÍTĚ PRO ROZPOZNÁNÍ TVÁŘÍ VE VIDEU

DEEP LEARNING FOR FACIAL RECOGNITION IN VIDEO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **VLADIMÍR JEŘÁBEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL HRADIŠ, PhD.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21083/2017/xjerab21

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Jeřábek Vladimír**
Obor: Informační technologie
Téma: **Hluboké neuronové sítě pro rozpoznání tváří ve videu**
Deep Learning for Facial Recognition in Video
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy teorie neuronových sítí, konvolučních neuronových sítí a zpětné propagace chyb.
2. Vytvořte si přehled o současných metodách pro rozpoznávání tváří pomocí konvolučních hlubokých neuronových sítí ve video sekvencích.
3. Vyberte konkrétní metody a aplikujte je na úlohu rozpoznání tváří ve video sekvencích.
4. Obstarejte si databázi vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Taigman et al.: DeepFace: Closing the Gap to Human-Level Performance in Face Verification. CVPR 2014.
- Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman. "Deep face recognition." Proceedings of the British Machine Vision 1.3 (2015): 6.
- Yang, Jiaolong, et al. "Neural aggregation network for video face recognition." arXiv preprint arXiv:1603.05474 (2016).
- Ebrahimi Kahou, Samira, et al. "Recurrent neural networks for emotion recognition in video." Proceedings of the 2015 ACM on International Conference on Multimodal Interaction. ACM, 2015.
- Yang, Jiaolong, et al. "Neural aggregation network for video face recognition." arXiv preprint arXiv:1603.05474 (2016).

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hradiš Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
L.S. 612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký

Abstrakt

Tato práce se zabývá rozpoznáváním tváří ve videu pomocí neuronových sítí. Na začátku je popsán výběr a ověření konvolučních neuronových sítí pro generování příznakových vektorů z obrázků různých identit. V další části se tato práce věnuje agregování příznakových vektorů ze snímků videa. Agregování probíhá pomocí agregačních neuronových sítí. Na konci této práce jsou diskutovány výsledky, kterých dané agregační metody dosáhli.

Abstract

This work deals with face recognition in video using neural networks. IIn the beginning, there is described the process of selection and verification of convolution neural network to generate feature vectors from images of different identities. In the next part, this work deals with the aggregation of feature vectors from video frames. Aggregation takes place through aggregation neural networks. At the end of this work, the results obtained by the aggregation methods are discussed.

Klíčová slova

Rozpoznávání tváří, hluboké neuronové sítě, agregace, UMDFaces, Dlib, konvoluční neuronové sítě, extrakce příznakového vektoru

Keywords

Facial recognition, deep neural networks, aggregation, UMDFaces, Dlib, convolutional neural networks, feature vector extraction

Citace

JEŘÁBEK, Vladimír. *Hluboké neuronové sítě pro rozpoznání tváří ve videu.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, PhD.

Hluboké neuronové sítě pro rozpoznání tváří ve videu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše, Ph.D.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vladimír Jeřábek
16. května 2018

Poděkování

Rád bych chtěl velmi poděkovat mému vedoucímu práce Ing. Michalovi HRadišovi, Ph.D., za jeho ochotu, pomoc, cenné rady a investovaný čas, který mi poskytl při odborném vedení této bakalářské práce.

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

Obsah

Kapitola 1

Úvod

V současné době dochází ke značnému pokroku v oblasti strojového učení, konkrétně pak i v oblasti počítačového vidění. Tento pokrok je způsoben díky konvolučním neuronovým sítím a taky díky pokroku ve výpočetní technice. Jednou z úloh, které se řeší v oblasti počítačového vidění [6], je i rozpoznávání tváří, kterou se tato práce zabývá.

Rozpoznávání obličejů je jedna z nejdůležitějších schopností člověka, kterou používáme v každodenním životě. Rozpoznávání obličejů má několik výhod oproti jiným biometrickým znakům, jako například otisk prstu nebo rozeznání duhovky. Jednou z nejvýznamnějších výhod tváře je, že může být zachycen z dálky a skrytým způsobem. Existuje několik důvodů proč se rozrůstá zájem o automatizaci v této oblasti. Jedním z důvodů je například identifikace osoby na hranicích a letištích. Dalším důvodem je například ověření identity, pro získání fyzického či logického přístupu do mobilních zařízení.

Existuje však velké množství metod, které řeší problém automatické identifikace tváře z obrázku, avšak většina z nich je založena na přístupu, kdy se generuje tzv. příznakový vektor, který reprezentuje identitu jedince na obrázku. Tyto příznakové vektory pak mohou být porovnávány a nebo uloženy do galerie k pozdějšímu porovnávání. Příznakové vektory jsou oproti obrázkům kompaktních velikostí. Tato práce se zabývá použitím podobného přístupu, tedy reprezentovat identitu jedním příznakovým vektorem, avšak ve videu.

Řešení tohoto problému je rozděleno do dvou částí. V první části se vybírá konvoluční neuronová síť, která slouží pro extrakci příznakových vektorů z obrázku. Úkolem této sítě je generovat příznakové vektory z obrázků tak, aby u tváří patřící stejné identitě byla vzdálenost příznakových vektorů co nejmenší, a naopak u vektorů reprezentující rozdílné identity, aby tato vzdálenost byla co největší.

V další části jsou informace z příznakových vektorů agregovány v jeden vektor, který charakterizuje identitu v daném videu. Tyto metody ohodnocovávají jednotlivé prvky v příznakových vektorech a na základě toho poté přiřazují těmto prvkům odpovídající váhy, které se promítnou ve výsledném vektoru.

Následující kapitola obsahuje stručný úvod do neuronových sítí, jaké existují vrstvy a také jak se tyto sítě trénují. Dále v kapitole 3 jsou popsány existující metody, které se zabývají rozpoznáváním tváří a nebo agregací příznakových vektorů. Dostupné datasety a použité metody pro rozpoznávání tváří ve videu se nachází v kapitole 4. Tato kapitola také pojednává o způsobu trénování jednotlivých agregačních neuronových sítí. V další kapitole 5 jsou popsány jednotlivé experimenty, které byly provedeny za použití neuronových sítí. Nachází se zde také porovnání námi použitých metod vůči *state-of-the-art* metodám. V závěru této práce jsou poté shrnuty výsledky, kterých bylo dosaženo, a je zde pojednán i případný budoucí vývoj.

Kapitola 2

Umělé neuronové sítě

Model umělých neuronových sítí byl inspirován [10] na základě biologické podobnosti s centrálním nervovým systémem. Hlavním posláním centrálního nervového systému je řídit organismus. Toho je docíleno pomocí zpracovávání vstupních signálů, čímž jsou zpravidla vzruchy smyslových receptorů, a následném vyslání pokynu směrem k efektorům, například svalům. Tato síť je však schopná se učit reakcím na nové vzruchy a to i díky vznikáním, či zanikáním spojů mezi jednotlivými neurony.

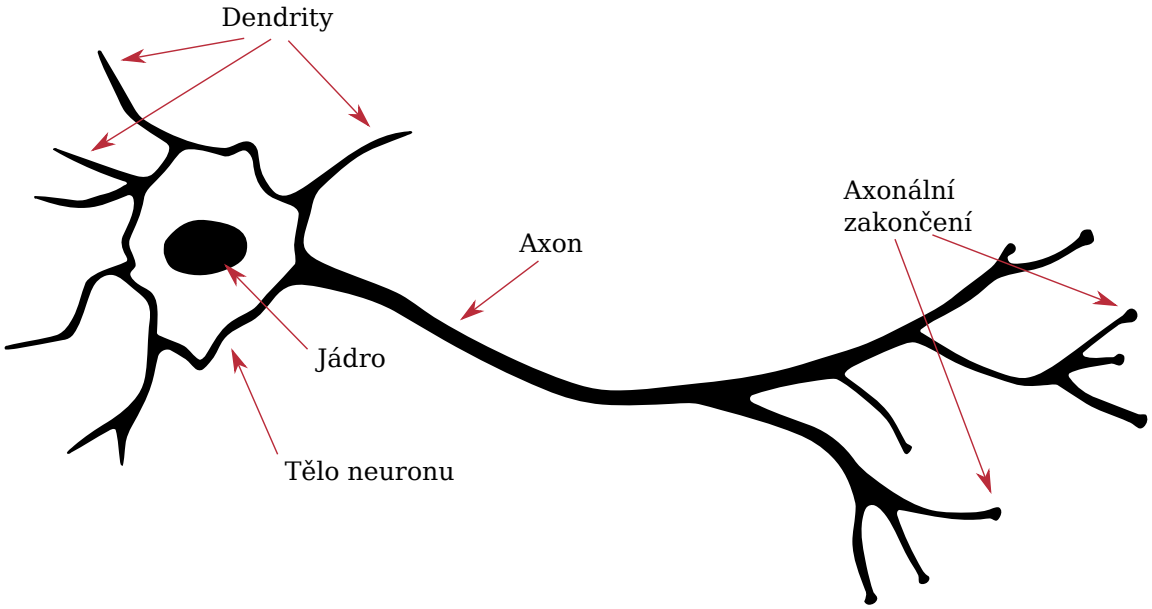
Na velmi zjednodušené analogii fungují již zmíněné neuronové sítě, které se skládají taktéž z velkého množství umělých neuronů a spojů mezi nimi. Díky tak velkému množství umělých neuronů, které jsou složeny především z trénovatelných parametrů, tak tyto sítě nabízí schopnost učit se. Zásluhou tohoto faktu je, že jsou tyto počítačové modely používány při různých úlohách, jako například počítačové vidění [6], rozpoznávání řeči, strojový překlad [1], či různé druhy klasifikace a další.

2.1 Model neuronu

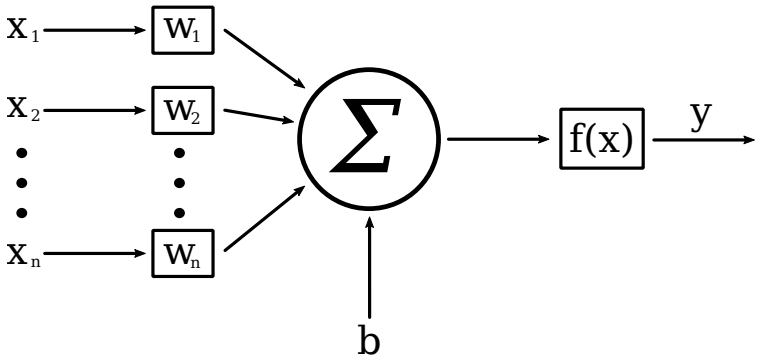
Základní stavební jednotkou mozku je biologický neuron [10] (viz obrázek 2.1 a). Jednotlivé tyto neurony jsou mezi sebou propojeny vazbami, které se označují jako *synapse*. Každý neuron je vybaven několika výběžky, dostředivé výběžky se nazývají *dendrity* a výběžek, který předává signál dalším neuronům je tzv. *axon*. Tyto neurony lze tedy chápat jako výpočetní jednotky, které na základě vstupu realizují výstup.

Na základě tohoto pozorování vznikl jeden z nejpoužívanějších modelů dnešní doby a to model (viz obrázek 2.1) publikovaný vědci *W. McCulloch* a *W. Pitts* v roce 1943 [19]. V roce 1957 na tuto práci navázal Frank Rosenblatt [26], který zobecnil model neuronu z roku 1943 na tzv. *perceptron* a taktéž navrhl nový učící algoritmus. Další informace ohledně vývoje umělých neuronových sítích lze nalézt v knize s názvem „*Neural networks : a comprehensive foundation*“ [10].

Do každého umělého neuronu vede určitý počet vstupů, který lze chápat jako vstupní vektor $x = (x_1, x_2, x_3, ..., x_n)$. Jednotlivé vstupy neuronu jsou obecně z množiny reálných čísel. Každý vstup je nejprv vynásoben odpovídající váhou. Jednotlivé váhy vstupů lze taktéž chápat jako vektor $w = (w_1, w_2, w_3, ..., w_n)$. Dále jsou tyto vstupní hodnoty sečteny a v případě existence skalární veličiny v neuronu tzv. *biasu* je k tomuto součtu vstupů přičten ještě *bias*. Dále je v neuronu obsažena *přechodová funkce* [21] (často označovaná jako *aktivační funkce*), jejíž výstupem je výstup celého neuronu.



a) Biologický neuron



b) Matematický model neuronu

Obrázek 2.1: Porovnání biologického neuronu a abstraktního modelu neuronu

Funkci neuronu lze jednoduše matematicky vyjádřit následovně:

$$y = f(\sum_{i=1}^{n} w_i x_i + b), \tag{2.1}$$

kde x_i představuje příslušnou vstupní hodnotu do neuronu, w_i je odpovídající váha pro daný vstup, b je *bias* neuronu a funkce f je *přechodovou funkcí*. Výstupem neuronu je hodnota y. V rámci fáze učení jsou měněny koeficienty jako váhy w_i a hodnota *biasu* b.

Přechodové funkce. Přechodové funkce jsou funkcemi neuronu, které realizují zobrazení $\mathbb{R} \to \mathbb{R}$ a určují výstup neuronu. Tyto funkce mohou být jak lineární tak nelineární, nicméně nelinearita aktivační funkce je velmi důležitá. V článku od *P. Panfilov* [23], je ukázáno, že pokud by neuronová síť obsahovala pouze posloupnost lineárních prvků (neuronů s lineárními aktivačními funkcemi), pak by tato sekvence produkovala lineární transformaci

a celá neuronová síť by byla ekvivalentní jednomu neuronu (nebo jedné vrstvě lineárních neuronů - v případě několika výstupů).

V poslední době se těší velké popularitě aktivační funkce s názvem: ***Rectified Linear Unit*** (zkráceně *ReLU*) [22]. Tato funkce je definována vzorcem:

$$f(x) = max(0, x), \tag{2.2}$$

kde x je výstup z neuronu před aktivační funkcí. V práci *G. E. Dahl et al.* [4] bylo ukázáno, že použitím této funkce se zlepšuje rychlost trénování u neuronových sítí s konvolučními vrstvami.

Více informacích o aktivačních funkcích lze nalézt v knize s názvem „*Neural networks : a comprehensive foundation*“ [10].

2.2 Základní vrstvy neuronových sítí

Jednotlivé umělé neurony jsou seskupovány do větších celků, kterým se říkají vrstvy. Každá neuronová síť může být složena s více různých vrstev. Tato topologie neuronových sítí je označována jako *více-vrstvá* [10]. Vstupem a výstupem každé neuronové vrstvy bývá obecně *tenzor* $n-$tého řádu. Výstupy z jedné vrstvy mohou být vstupem pro vrstvy další.

Níže je popsáno několik základních vrstev neuronových sítí, které jsou dále použity v této práci.

Plně propojená vrstva. Nejjednodušší vrstvou používanou v neuronových sítích je plně propojená vrstva. Tato vrstva se skládá z určitého množství umělých neuronů, kde každý neuron je navázán na všechny vstupy této vrstvy. Výstupem této vrstvy je vektor složený s výstupů jednotlivých neuronů v plně propojené vrstvě. Celkový počet trénovatelných parametrů N lze matematicky vyjádřit jako

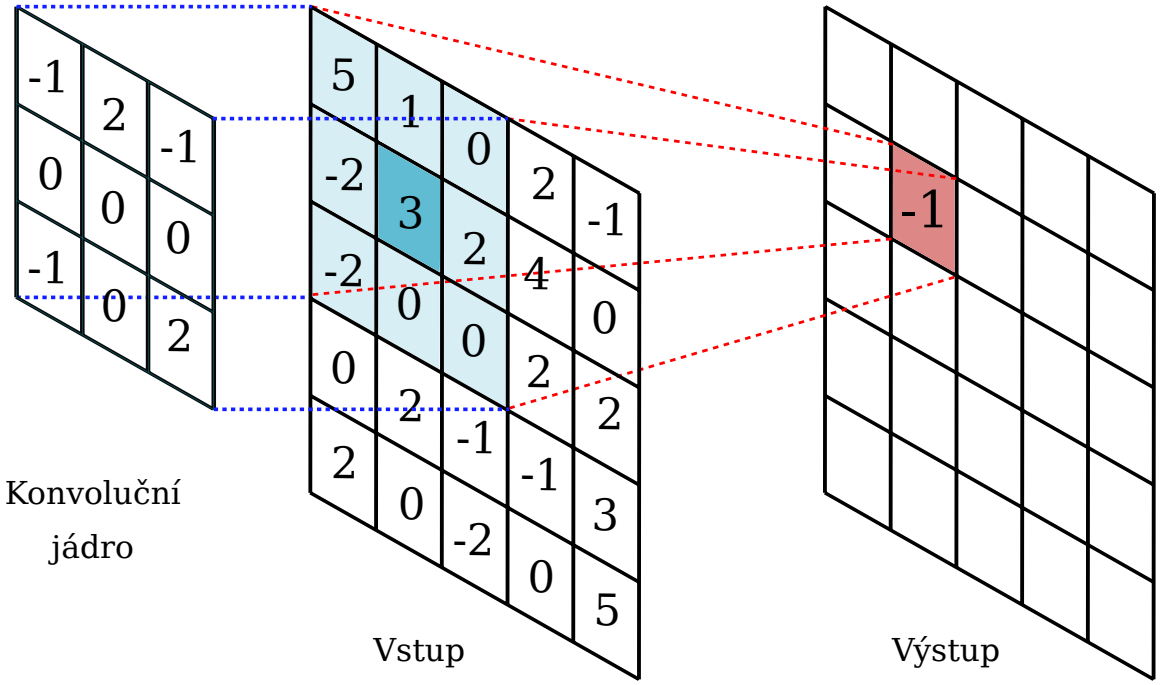
$$N = k(n + 1), \tag{2.3}$$

kde k je počet neuronů v plně propojené vrstvě, n je dimenze vstupního vektoru.

Konvoluční vrstva. Motivací pro vznik konvoluční vrstvy byl fakt, že při zpracovávání obrázkových dat může docházet k posunutí, částečné deformaci a taktéž k rotaci daných objektů na obrázku, což způsobovalo velký problém pro plně propojené vrstvy při učení. Totiž jednotlivé neurony se učily, kde nalézt přesné pozice významných bodů v obrázku, avšak při posunutí objektu tyto významné body tam nenacházely. Dalším problémem byl počet učících parametrů. Například při vstupním obrázku 32x32 pixelů o 3 kanálech je dimenze vstupního vektoru 3072, a pokud by bylo alespoň 128 neuronů v plně propojené vrstvě, pak celkový počet učících parametrů v první plně propojené vrstvě (dle vzorce 2.3) je 393 344. Více informací o motivaci pro vznik konvoluční vrstvy je popsáno v knize *Deep Learning* [8].

Tyto problémy byly vyřešeny představením a použitím principu konvoluce v neuronových sítích v roce 1979 [7]. Myšlenkou této vrstvy je získávání lokálních charakteristik obrázku a za použití techniky sdílení vah mezi neurony je dosaženo robustnosti právě vůči transformacím a různým deformacím vstupních objektů na obrázku.

Základní operací této vrstvy je konvoluce (viz obrázek 2.2). Obecně konvoluce [8] je operací nad dvěma funkcemi a lze ji zapsat rovnicí



Obrázek 2.2: Ukázka operace konvoluce nad 2D maticí s konvolučním jádrem 3x3.

$$s(t) = \int x(a)w(t-a)da, \tag{2.4}$$

kde $x(a)$ a $w(t-a)$ jsou vstupními spojitými funkcemi a $s(t)$ je výstupem konvoluce nad vstupními funkcemi.

V našem případě, pracujeme-li s daty na počítači, bude čas diskretizován a tedy definice konvoluce bude vypadat takto:

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{2.5}$$

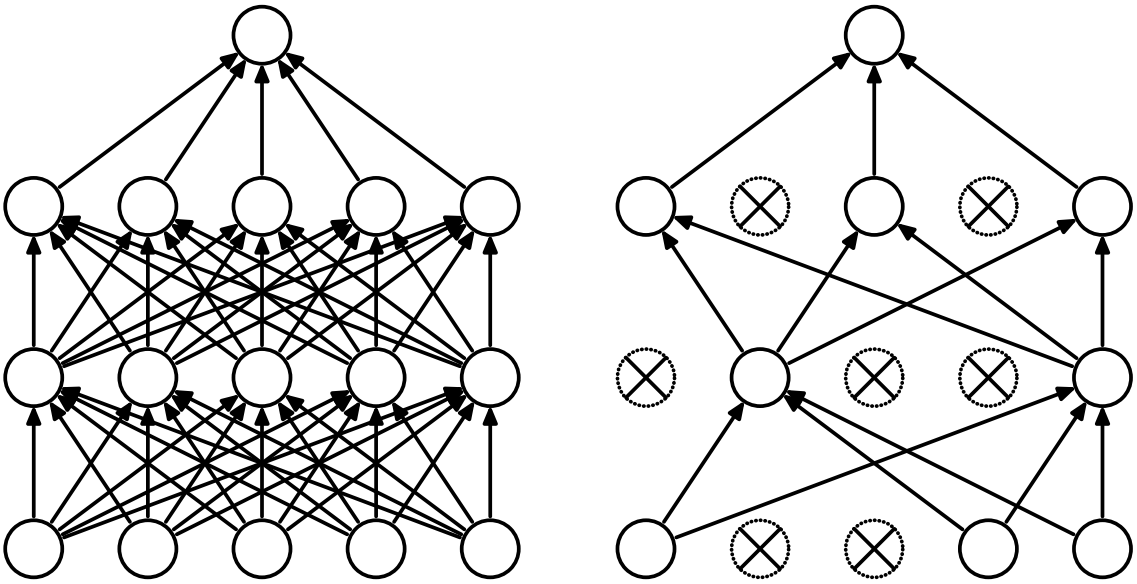
Podobně jako u rovnice 2.4, je i tato rovnice složena z dvou vstupních funkcí $x(a)$ a $w(t-a)$. Vstupem takové to vrstvy je obecně tensor k-tého řádu a výstupem je taktéž tensor, avšak obecně jiného například n-tého řádu.

V počítačovém vidění se nejčastěji pracuje s obrázky, které obsahují 3 barevné kanály, tedy vstupními daty je nejčastěji tensor 3-ho řádu. Konvoluce nad těmito daty vypadá následovně:

$$s_t[i,j] = \sum_c \sum_k \sum_l (x_c[i-k, j-l] w_{c,t}[k,l]) + b_t, \tag{2.6}$$

kde t je index výstupního kanálu, i a j jsou index daného bodu v kanálu, c je index vstupního kanálu, k a l jsou rozměry konvolučního jádra (častokrát označovaného jako *kernel*), $w_{c,t}$ je konvoluční jádro pro daný vstupní kanál c a výstupní kanál t a b_t je *bias* pro daný výstupní kanál t.

U této vrstvy se nastavuje velikost konvolučního jádra a počet výstupních kanálů. Dále lze nastavit tzv. *padding*, což je způsob jakým se rozšiřuje původní obrázek tak, aby výstup z této vrstvy byl o stejných rozměrech jako vstup .



Obrázek 2.3: Příklad aplikování *dropout* vrstvy. **Vlevo:** Standardní neuronová síť s plně propojenými vrstvami. **Vpravo:** Příklad neuronové sítě zleva, na kterou byl aplikován *dropout* (neurony s křížem byli zahozeny). Převzato z [30].

Dropout vrstva. Motivací pro vznik této vrstvy (popsané v práci *N. Srivastava et al.* [30]) byl problém přetrénování (tzv. *overfitting*), kdy dochází ke zvýšení chybovosti na validačních datech, avšak u dat trénovacích dochází k výraznému snížení hodnot chybové funkce. Toto přetrénování je jeden z hlavních problémů velkých neuronových sítí s hodně parametry. Velký počet parametrů umožňuje takovým sítím se naučit velmi komplikované vztahy mezi jejich vstupními a výstupními daty, což na jednu stranu je chtěné, nicméně na stranu druhou si tato síť vytvoří vazby, které nemají co společného s danými vstupními daty. Jde o vazby, které jsou specifické pro daný vstup a výstup, ne však obecně.

Tento neduh je řešen pomocí tzv. *dropout* (viz obrázek 2.3), kdy u náhodně vybraných neuronů (společně s jejich vazbami) dochází k zahození jejich výstupů. K tomuto procesu zahazování dochází pouze při trénování sítě.

Tato vrstva má pouze jeden parametr, který lze nastavit a to je míra zahazování.

Softmax vrstva. Tato vrstva je jedna z aktivačních vrstev, jejíž výstupem je vektor reálných hodnot, která jsou v rozmezí od $(0, 1)$. Součet jednotlivých hodnot vektoru je vždy roven 1, a proto se tato aktivační funkce nejčastěji používá pro kategorickou pravděpodobnost, tedy pro klasifikaci 1 z K. Výpočet k-té hodnoty softmax vrstvy je definován následovně:

$$y_k = \frac{e^{x_k}}{\sum_{j=1}^{K} e^{x_j}}, \tag{2.7}$$

kde x_k je k-tá hodnota ze vstupního vektoru x, K je dimenze vstupního vektoru a y_k je k-tá hodnota výstupního vektoru y.

Pooling vrstva. Tato vrstva slouží k tzv. *podvzorkování*, neboli zmenšení prostorového rozlišení. V práci *Y. Boureau et al.* [3] se zabývají různými přístupy k tomuto zmenšení prostorového rozlišení. *Pooling* metoda je založena na zastupování (chcete-li nahrazování)

aktivačních hodnot z lokálních oblastí jednou hodnotou, která je vypočítána z původních hodnot a to například funkcemi jako: průměr, nebo maximum. Tato vrstva bývá používána především mezi vrstvami konvolučními. U této vrstvy se taktéž udává velikost lokální oblasti, která má být pomocí dané metody sjednocena a nahrazena tak jednou zástupnou hodnotou.

2.3 Trénování neuronových sítí

Vlastnost, která má primární význam pro umělé neuronové sítě, je schopnost sítě se *učit*, tedy zlepšovat jeho výkonnost na trénovacích datech. Při tomto interaktivním procesu dochází k (pře)nastavování vah neuronů, za účelem minimalizovat chybovou funkci [10]. K tomu se například užívají metody zpětného šíření chyby nebo *gradientní sestup*. Tyto metody jsou blíže popsány v následujících řádcích.

Chybová funkce. Známá taktéž jako *loss funkce*, je funkcí která určuje rozdíl mezi výstupem sítě a očekávaným výstupem, po výpočtu této chyby je hodnota propagována skrze neuronovou síť a jsou měněny váhy jednotlivých neuronů.

V tomto článku bude využívaná funkce *Contrastive loss*, která byla zavedena v práci *R. Hadsell et al.* [9]. Tato funkce patří mezi tzv. *vzdálenostní* chybové funkce, což znamená, že výsledná chyba je určena na základě vzdáleností dvou prvků. V případě této práce předkládány vzdálenosti dvou příznakových vektorů. Tento příznakový vektor je kompaktní reprezentace tváře na obrázku. Úkolem této chybové funkce je vzdálenost mezi příznakovými vektory stejných osob zmenšovat a naopak vzdálenost příznakových vektorů odlišných osob zvětšovat až za hranici tzv. *margin* (viz obrázek 2.4). Tuto funkce je definována jako

$$L(X_1, X_2) = (1 - Y)\frac{1}{2}(D_w)^2 + (Y)\frac{1}{2}\{max(0, m - D_w)^2\}, \tag{2.8}$$

kde L je výsledná chyba pro dané příznakové vektory X_1 a X_2, proměnná Y je binární hodnotou přiřazená k danému páru. $Y = 0$, jestliže X_1 a X_2 jsou vektory stejné identity a naopak $Y = 1$, jestliže X_1 a X_2 jsou to vektory různých identit. Hodnota D_w určuje vzdálenost vektorů X_1 a X_2 od sebe na základě předpovědi neuronové sítě. Hodnota m je tvz. *margin*, který nám určuje ideální vzdálenost stejných a různých identit od sebe navzájem, tato hodnota se musí nastavovat.

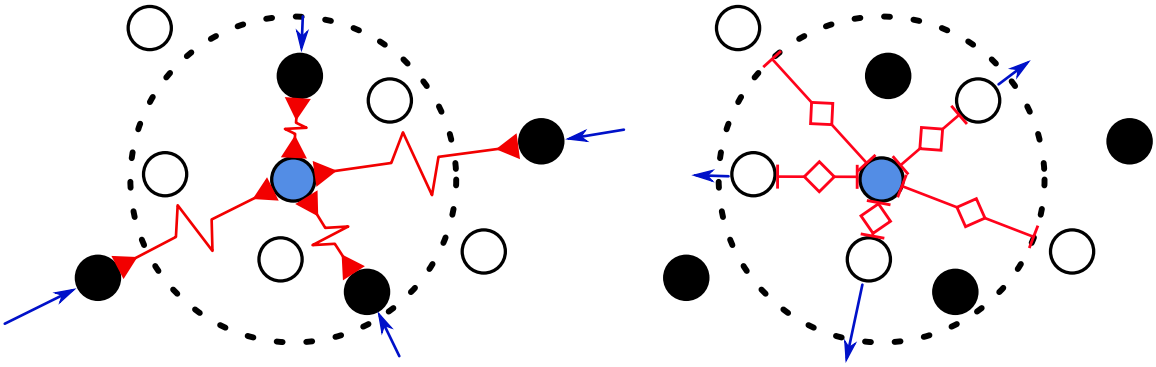
Zpětná propagace chyby. Též známá jako *backpropagation* [17], je metoda, která slouží k počítání parciálních derivací chybové funkce vzhledem k vahám neuronům. Tyto vypočítané gradienty jsou vstupem pro optimalizační metody, jakou je například metoda zvaná *gradient descent* (více v sekci 2.3) .

U této metody dochází k dopřednému průchodu sítí, za účel získání výstupu sítě, dále výpočet chyby (vůči cílovému výsledku, viz sekce chybová funkce 2.3) a posléze výpočet gradientů (parciálních derivací) této chyby u jednotlivých neuronů. Při této části se využívá tzv. *chain rule* (řetězové pravidlo), které je definováno:

$$\frac{dF}{dx} = \frac{df}{dg}\frac{dg}{dx}, \tag{2.9}$$

platí-li:

$$F(x) = f(g(x)). \tag{2.10}$$



Obrázek 2.4: Pružinový model chybové funkce *contrastive loss*, podle [9]. Plné kruhy představují body, které jsou podobné s bodem uprostřed (modrý kruh). Prázdné kruhy představují body, které jsou nepodobné s centrálním bodem. Pružiny jsou znázorněny jako červené *zigzag* čáry. Síla, která působí na jednotlivé kruhy je znázorněna modrou šipkou, jejíž velikost by měla přibližně odpovídat délce dané čáry. **Obrázek vlevo** znázorňuje body, které jsou spojeny přitahující pružinou. **Obrázek vpravo** znázorňuje vytlačování nepodobných bodů, které jsou v kruhu o poloměru m (*margin*) od centrálního bodu.

Tyto průchody sítí (jak dopředné, tak i zpětné), se iterativně opakují, dokud chybovost sítě není přijatelná.

Optimalizace. Vstupem pro optimalizační algoritmus jsou gradienty chybové funkce u jednotlivých vah neuronů, který je výstupem metody *backpropagation* popsané v sekci 2.3. Nicméně úkolem optimalizační metody je správné nastavování vah, aby se minimalizovala chybová funkce nad daným vstupem.

Jedním z nejznámějších optimalizačních metod je *gradient descent* [28] neboli gradientní sestup. U této metody dochází k aktualizaci parametrů jednotlivých neuronů podle rovnice:

$$w_{i+1} = w_i + \Delta w_i, \tag{2.11}$$

kde w_i označuje současnou hodnotu váhy, w_{i+1} je nová hodnota váhy. Hodnota Δw_i je rozdíl o který je současná hodnota parametru změněna. Tuto změnu lze vypočítá podle rovnice:

$$\Delta w_i = -\eta g_i, \tag{2.12}$$

kde g je gradient váhy v i-té iteraci a parametr η je koeficient učení (tzv. *learning rate*).

Na základě této metody existuje velké množství dalších metod, jako je například velmi používaná *stochastic gradient descent* (*SGD*) [37], nicméně nevýhodou této metody je nutnost experimentálně nalézt ideální hodnotu pro učící koeficient.

Existují však optimalizační algoritmy, které nalezení učícího koeficientu zajišťují samy. Jednou z těchto metod je například metoda *Adam* [16], což je zkratka pro *Adaptive moment estimation*.

Kapitola 3

Rozpoznávání tváří

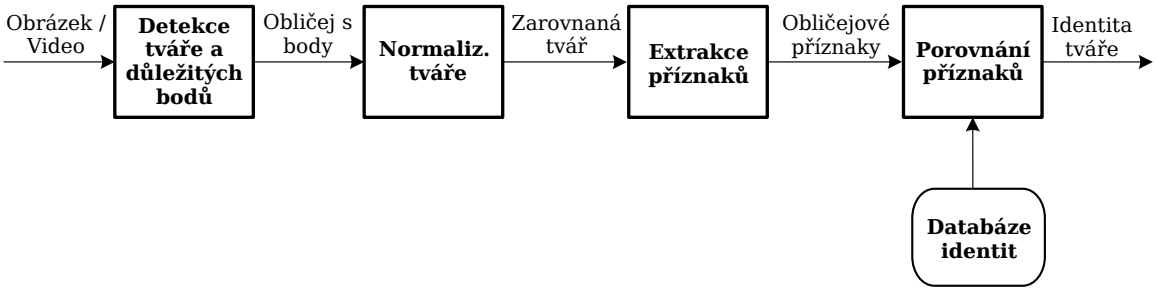
Rozpoznávání tváří lze obecně zařadit do oblasti rozpoznávání vizuálních vzorů. Jednotlivé tváře na obrázku, ač jsou to 3D objekty, tak jsou zachyceny jako 2D předměty v určitém čase při různém osvětlení. Tyto měnící se faktory (osvětlení, póza tváře, a další) nám výrazně ovlivňují výsledný obrázek, což samotný proces rozpoznání činí obtížnějším. Za účelem eliminace těchto měnících se faktorů jsou do procesu zpracování obrazu přidány moduly například pro zarovnání tváře. Pro lepší získání základního přehledu o problému rozpoznávání tváří odkazuji čtenáře na knihu pánů *A. K. Jain* a *S. Z. Li* [14].

Systém rozpoznávání tváří se obecně skládá ze 4 základních částí, které jsou zobrazeny na obrázku 3.1: lokalizace (detekce) obličeje, normalizace (ořezání a zarovnání), extrakce příznaků a porovnávání příznaků. Jednotlivé moduly jsou popsány níže.

Lokalizace tváře. Tato část procesu (například pomocí metody popsané v práci *P. Viola et al.* [34]) rozpoznávání tváře se zaměřuje na poznání obličeje od pozadí. V případě videa musí být tento detekovaný obrázek sledován skrze několik snímku. Až detekce obličeje nabízí informace ohledně přibližné polohy obličeje a jeho velikosti, následuje lokalizace významných bodů obličeje (jako například: oči, nos, pusa, obrys obličeje). Po nalezení těchto důležitých obličejových bodů je připraven obrázek s tváří na normalizaci.

Ořezání a zarovnání. Taky označovaná jako normalizace obličeje. Tento proces se provádí tak, aby výsledný obličej na obrázku byl zarovnán způsobem, jak to očekává modul pro extrakci příznaků. V tomto procesu dochází jak ke geometrickému zarovnání, tak i v některých případech k fotometrické normalizaci. Toto zarovnání je důležité a nezbytné, protože se od systému rozpoznání obličeje očekává, že bude schopen rozeznávat obličeje i při různých pózách a podmínkách osvětlení. Geometrická normalizace se nejčastěji provádí ořezáním, a zarovnáním významných bodů obličeje v očekávaném tvaru (je určena pozice očí, atd.). Fotometrická normalizace je proces, při kterém se mění vlastnosti fotky, jako například světelnost fotky.

Extrakce příznaků. Tento modul pracuje s normalizovanou tváří a účelem tohoto procesu je získání významných informaci z obličeje, které pak slouží k rozlišování obličejů a poté určení například shodnosti identity. Tyto metody musí být robustní vůči menším deformacím obličeje na obrázku, i přesto, že se v modulu před tímto procesem obrázky normalizují.



Obrázek 3.1: Zobrazení postupu při rozpoznávání tváře z obrázku, či videa. Převzato z [14].

Porovnávání příznaků. Při procesu porovnávání, či rozlišování tváří jsou reprezentace obličejů porovnávány za účelem získání nějaké informace. V tomto modulu rozlišujeme dvě základní úlohy a to *verifikaci* a *identifikaci*.

Verifikace je proces, při kterém se ověřuje, zda dvě reprezentace tváří se vztahují k jedné a téže identitě (taky označované jako ověřování 1:1). Výstupem je tedy vždy binární odpověď 'ano' či 'ne'. Porovnání se zde provádí v závislosti na určitém prahu (tzv. *threshold*). Hodnota prahu se může měnit a tedy je potřeba ho vždy za určitým účelem nastavit. Verifikace se například může používat k udělení přístupu na mobilní zařízení, kdy je ověřován uživatel daného zařízení (předpokládáme-li, že tímto uživatelem je právě jedna osoba).

Identifikace je na rozdíl od verifikace proces, při kterém se přiřazuje identita vstupnímu obličeji. Předpokládáme-li, že máme přístup k databázi (galerii) s N reprezentacemi jednotlivých identit, pak je vstupní obličej porovnán s každým záznamem v databázi. Výsledky porovnání lze poté seřadit podle míry podobnosti od nejpodobnějších až po ty nejméně podobné.

V případě této práce je použito pro *reprezentaci tváře* příznakového vektoru, což je v podstatě n-rozměrný vektor, jehož hodnoty na jednotlivých dimenzích jsou prvky z množiny reálných čísel.

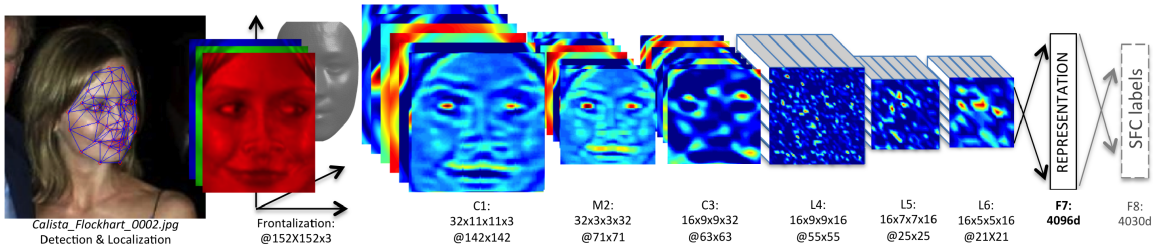
Tato reprezentace by nemusela být potřeba, pokud bychom porovnávali přímo ty obrázky vůči sobě. Nicméně pokud bychom chtěli identifikovat danou tvář na obrázku a existovala by databáze obličejů, pak při porovnávání každého páru by tato metoda byla značně neefektivní a zdlouhavá. Za tímto účelem se tedy tváře reprezentují jako vektor reálných čísel.

V následujících řádcích této kapitoly jsou popsány *state of the art* metody, které se zaměřují na jednotlivé části rozpoznávání obličejů.

3.1 Související práce

Existuje mnoho architektur sítí a metod, které se zabývají problémem rozpoznávání tváří, nebo podobným problémem klasifikace obrázků, avšak velký úspěch zaznamenávají metody, které jsou založené na „hlubokém učení“ (tzv. *deep learning*). Architektury těchto sítí obsahují velký počet konvolučních vrstev (označované jako konvoluční neuronové sítě *CNN*). Metody, které jsou založené na jiném principu zde nebudou zmiňovány.

Sítě pro rozpoznávání tváří. Jako zástupce těchto metod bych zde uvedl síť *DeepFace* [32]. Výsledky, které byli dosaženy pomocí neuronové sítě *DeepFace*, jsou značně srovnatelné



Obrázek 3.2: Ukázka architektury sítě *DeepFace*, převzato z [32].

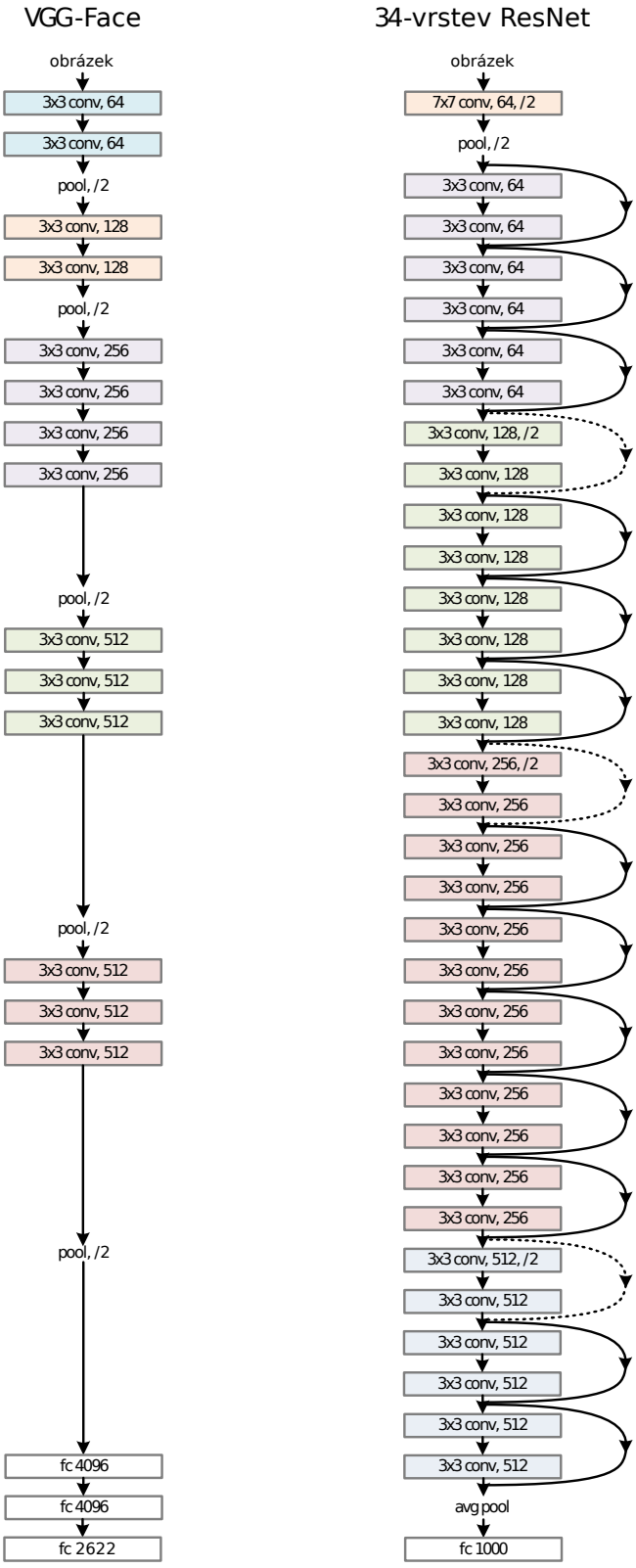
s výsledky lidí. Tyto výsledky byly dosaženy na datasetu *LFW* [13], kde prezentovaná síť *DeepFace* dosahovala úspěšnosti 97.35% a lidé 97.53%. Zajímavostí je i samotná architektura sítě (viz. obrázek 3.2), kde se kromě dvou konvolučních vrstev na začátku sítě nacházejí ještě vrstvy *lokálně-propojené*, které svojí funkcionalitou připomínají vrstvy konvoluční, avšak jejich konvoluční jádra pro jednotlivé lokální oblasti nesdílí váhy. Díky tomu se odlišné části obrázku nezpracovávají stejným způsobem. Z tohoto důvodu je zřejmé, že daná architektura sítě předpokládá vstupní obrázky zarovnány a to velmi kvalitně, konkrétně pak je v architektuře sítě *DeepFace* přítomný modul pro zarovnání, který detekovanou tvář převede na jednoduchý 3D model a dle tohoto modelu poté zarovná obličej na očekávanou pozici pro další zpracování. Architektura této sítě je zakončena dvojicí plně propojených vrstev. Celkový počet modifikovatelných parametrů na tuto architekturu je něco kolem 120 miliónů, avšak z toho 95% se nachází v lokálně a plně propojených vrstvách. Při učení tak velkého množství parametrů bylo použito přes 4 milióny obrázků, které obsahovaly více než 4000 různých identit.

Práce *DeepFace* byla rozšířena sérií prací *DeepId*, které byly prezentovány v mnoha článcích, nicméně jejich poslední síť *DeepID3* [31] je sítí, která kromě konvolučních vrstev využívá na konci sítě *Inception* bloky, což je blok složený z několika konvolucí a pooling vrstvy vedle sebe. S touto sítí se ještě více zvedla přesnost při rozpoznávání a to jak na *LFW* datasetu, tak i na *YouTube Faces Dataset* (*YTF*) [35]. Nicméně architektura této sítě je velmi komplikovaná. Jako příklad mohu uvést třeba fakt, že obsahuje kolem 200 konvolučních vrstev.

Další architekturou sítě, která řeší problém rozpoznávání tváří je síť *VGG-Face* [25]. Tato síť je schopna , i přes poměrně jednoduchou architekturu (viz obrázek 3.3), dosáhnout podobných výsledků jako metody popsané výše. Na rozdíl však od těchto metod byla použita při trénování chybová funkce založená na porovnávání trojic tzv. *triplet loss* funkce. Tato metoda byla poprvé představena výzkumníky z Google a to v článku *Facenet* [27]. Základem této metody je trojice tváří (a, b, c), kde dvojice tváří (a, b) jsou od stejné identity a tvář c je od jiné identity. Úkolem této chybové funkce bylo zmenšit vzdálenost obličejů a a b více než vzdálenost a a c.

Sítě pro klasifikaci obrázků. Z této kategorie sítí bych chtěl zmínit architekturu sítě, která byla vyvinuta výzkumníky z Microsoft, a nazývá se *ResNet* [12]. Architektura této sítě (viz obrázek 3.3), je specifická tzv. *reziduálními vazbami*, což jsou spoje mezi jednotlivými vrstvami, které umožňují lépe propagovat gradienty při učení i do hlubších vrstev sítě.

Díky této vlastnosti je možné učit i velmi hluboké sítě s několika set vrstvami.



Obrázek 3.3: Ukázka architektur sítí **vlevo:** *VGG-Face*, **vpravo:** *ResNet*. Převzato z [12]. Bloky představují jednotlivé vrstvy v neuronových sítích. Vrstvy označené jako *conv* jsou konvoluční a vrstvy označené jako *fc* jsou plně propojené. *Pool* je označení pro pooling vrstvu.

3.2 Agregační metody

Při rozpoznávání tváří z video sekvence není potřeba daný obličej rozpoznat na základě jedné fotky, ale lze využít informace z jednotlivých snímků videa a eliminovat tak stavy, při kterých je například špatná pozice obličeje vůči kameře, nebo špatné světelné podmínky. Jeden z dalších faktorů, který nelze zanedbat, je výpočetní složitost při identifikaci tváře na videu. Bez agregace je potřeba každý snímek videa porovnat s každým záznamem databáze a tedy pokud existuje databáze s K identitami a video s n snímky, pak porovnáváme-li každý snímek videa s každou identitou, bude celková výpočetní složitost $O(nK)$, což je nepřípustné při velkém počtu identit v databázi. Nicméně pokud dojde k agregaci videa na jeden vektor, pak celková výpočetní složitost je $O(K)$.

Pooling metody. Tyto metody jsou založeny na jednoduchých matematických metodách, které více příznakových vektorů sdruží do jednoho. Jeden z příkladů použití agregační metody založené na matematických operací je i práce *O. M. Parkhi et al.* [24]. Takovým příkladem pooling metody může být průměrování či výběr minimální, nebo maximální hodnoty na stejných indexech. Tyto metody jsou převážně jednoduché a výpočetně nenáročné.

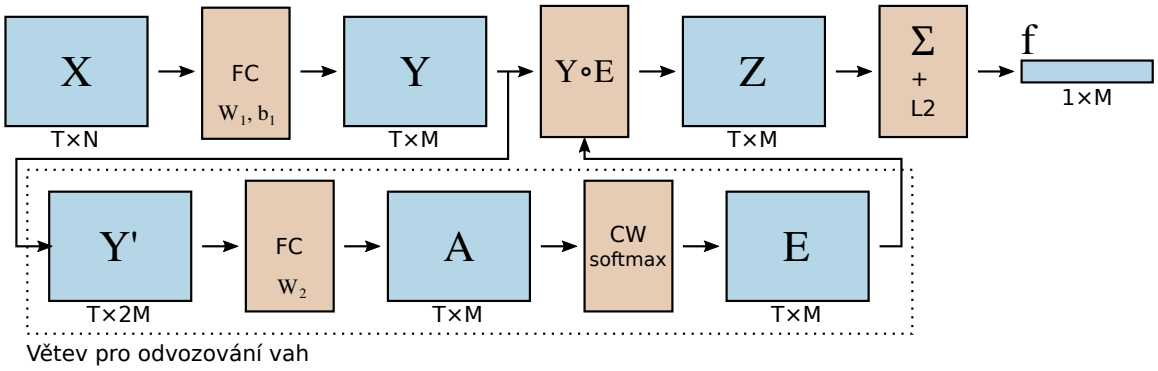
Agregační metody založené na neuronových sítích. Dalším způsobem pro agregování více příznakových vektorů v jeden je použití neuronových sítí. Obecně platí, že lze použít jakoukoliv neuronovou síť, jejíž vstupem je sada příznakových vektorů a výstupem je pak právě jeden agregovaný příznakový vektor. Nicméně některé architektury sítí vykazují lepší výsledky než jiné. Sdružování více příznakových vektorů se nezabývá jenom problematika rozpoznání tváří, ale také například odvětví re-identifikace chodců, nebo například re-identifikace aut.

Jedna z možných architektur sítí pro agregaci je popsána v práci od *N. McLaughlin et al.* [20], kde používají rekurentních neuronových sítí (*RNN*). Tato síť nepracuje pouze s jedním příznakovým vektorem, který má momentálně na vstupu, ale pracuje i s vektory, které už touto sítí prošly.

Struktura této sítě byla rozšířena výzkumníky *W. Zhang et al.* [38], kde představili obousměrnou (tzv. *bidirectional*) rekurentní neuronovou síť (*BRNN*), kterou používali pro problematiku re-identifikace lidí. Tato síť na rozdíl od jednoduché RNN, využívá dopředných vazeb, takže při zpracování konkrétního snímku videa je k dispozici temporální informace ze snímku předchozího, ale i budoucího.

Další architekturou neuronových sítí určenou pro agregaci je síť s názvem *Neural Agregation Network* (*NAN*) [36]. Tato síť byla použita v článku na rozpoznávání obličejů z videa. Funkcí agregačního modulu je ohodnotit příznakové vektory jednotlivých snímku videa váhami a s jejich pomocí vypočítat vážený součet těchto vektorů. Váhy jednotlivých snímků jsou vypočítány na základě skalárního součinu s dočasným vektorem o stejné délce jako příznakový vektor, který je výsledkem plně propojené vrstvy. Dále je výsledný vektor normalizován pomocí softmax funkce.

Jako poslední bych chtěl zde zmínit práci *J. Sochor et al.* [29], ve které byla představena architektura sítě *LFTD* (zkratka z *Learning Features in Temporal Domain*). Tato metoda je založena na ohodnocování příznaků u jednotlivých příznakových vektorů. Toto ohodnocování je na základě přidělování vah jednotlivým prvkům příznakových vektorů. V této práci byla metoda použita při úkolu re-identifikace aut, avšak její využitelnost sahá i do odvětví rozpoznávání tváří, či lidí.



Obrázek 3.4: Schéma agregační metody *LFTD*. Převzato z [29].

Vstupem do agregačního modelu (ilustrováno na obrázku 3.4), přestaveného v tomto článku je matice $\mathbf{X}$ o velikosti $T \times N$, kde T je počet snímku ve videu a N je dimenze příznakového vektoru pro jeden snímek videa. Hned na začátku tohoto modulu dochází ke kompresi velikosti příznakových vektorů z dimenze N na M. Toho je docíleno pomocí plně propojené vrstvy. Výstupem této vrstvy je matice $\mathbf{Y}$, jejíž velikost je $T \times M$.

Pro ohodnocení jednotlivých prvků v příznakových vektorech dané stopy videa je použita následující sekvence úloh. Jako vstup do této části agregačního modulu je matice $\mathbf{Y}$, která je popsaná výše. Pro lepší „komunikaci“ mezi jednotlivými příznaky je za každý příznakový vektor přidán průměrný příznakový vektor pro celou stopu, tento postup lze zapsat jako rovnici

$$\mathbf{y}'_\tau = \begin{bmatrix} \mathbf{y}_\tau \\ \frac{1}{T}\sum_{i=1}^{T} \mathbf{y}_i \end{bmatrix}, \tag{3.1}$$

odtud je poté vypočítána matice

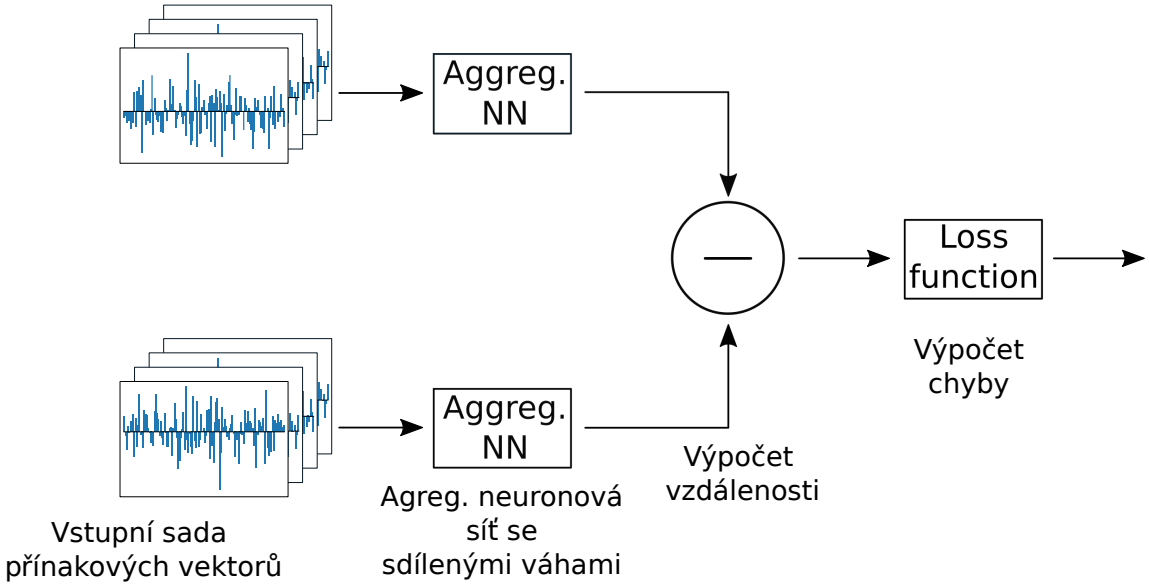
$$\mathbf{Y}' = [\mathbf{y}'_1, \mathbf{y}'_2, \mathbf{y}'_3, \ldots, \mathbf{y}'_T]^\top. \tag{3.2}$$

$\mathbf{y}_\tau$ je příznakový vektor pro každý snímek z matice $\mathbf{Y}$. Velikost matice $\mathbf{Y}'$ je $T \times 2M$. Dále je tato matice vstupem pro plně propojenou vrstvu, jehož výstupem je matice $\mathbf{A}$ o rozměru $T \times M$. Posledním blokem v této vyhodnocovací části agregačního modulu je softmax vrstva pro normalizaci(více o této vrstvě v sekci 2.2), nicméně tato normalizace není provedena podle řádků (po jednotlivých snímcích), avšak podle sloupců, kde se normalizují jednotlivé příznaky skrz celé video. Výstupem normalizace je matice $\mathbf{E}$, která má stejný rozměr jako matice $\mathbf{A}$.

V další fázi agregace se slučuje matice vah, tedy matice $\mathbf{E}$ se zmenšenou maticí $\mathbf{Y}$.Pro dané sloučení se používá tzv. Hadamardův součin (po jednotlivých složkách). Výsledný vektor $\mathbf{f}$ je pak sumou pro jednotlivé sloupce matice $\mathbf{Z}$. Dále je tento vektor *L2* normalizován. Vektor $\mathbf{f}$ po normalizaci je výsledným vektorem celé agregace a slouží poté k porovnávání s ostatními vektory.

3.3 Trénování sítí

Pro trénování neuronových sítí je možné použít *siamských sítí* (viz obrázek 3.5). Tato architektura sítě obsahuje jednu či více *identitckých* podsítí. Slovo *identitckých* je zde myšleno, že tyto sítě používají stejnou konfiguraci včetně stejně nastavených parametrů a vah. Tato architektura umožňuje jednotlivým sítím v této struktuře se učit společně i přesto, že na



Obrázek 3.5: Ukázka architektury siamské sítě.

jednotlivých vstupech jsou rozdílné data, v našem případě sekvence snímků. Na konci této sítě se pak měří vzdálenost skrze jednotlivé výstupy a výsledná hodnota je použita při určení chyby a to nejčastěji pomocí chybové funkce *Contrastive loss*, která je více popsaná v sekci 2.3.

Dalším způsobem pro zefektivnění trénování je metoda *hard - mining*. Tato metoda byla použita například v článku publikovaném výzkumníky *P. Felzenszwalb et al.* [5], kteří se zabývali problematikou detekce objektů na obrázku. Základní myšlenkou této metody je učení neuronové sítě na těžkých příkladech. Mezi tyto těžké příklady můžeme zařadit jak těžké pozitivní příklady (*hard pozitive mining*), nebo také těžké negativní příklady (*hard negative mining*). Termín těžké příklady je zde myšlena sada vstupních trénovacích dat, které jsou danou neuronovou sítí špatně vyhodnoceny. Ve výše zmíněném článku pracovali s *hard negative mining* metodou, jelikož více převažovalo negativních dat nad těmi pozitivními. Základem jejich metody jsou dvě třídy: třída H pro těžké (nesprávně vyhodnocené) příklady a třída E pro lehké (správně vyhodnocené) data - příklady s chybou 0 (určeno pomocí chybové funkce). Před každou trénovací iterací se vyberou náhodně data z trénovací sady D, které se posléze uloží do třídy C. Po každé iteraci jsou ze třídy D vyjmuty příklady, které byly správně určeny a tedy se nacházeli ve třídě E. Celý postup se znovu opakuje vybíráním dat z D a následné uložení do třídy C

Kapitola 4

Návrh řešení a práce s datasety

Problematika rozpoznávání tváří se skládá z několika základních kroků. Jako první je potřeba získat data na trénování neuronových sítí a také na ověření jejich funkcionality. Získaná data musí být dále vhodným způsobem upravena (ořezána a zarovnána), aby jednotlivé tváře na obrázcích byly konzistentní.

Dalším krokem následuje výběr sítě pro extrakci příznakových vektorů z obrázků. Tento krok je velmi důležitý, protože pokud je přítomna velmi dobrá síť na extrakci příznaků, pak agregační síť pracuje s kvalitními daty, podle kterých se může naučit rozeznávat nějaké vzory. Na internetu existuje několik již natrénovaných sítí, takže lze vynechat fázi trénování, avšak je potřeba ověřit si jejich kvalitu.

Dále následuje otázka, jakou zvolit metodu agregace. Agregační síť by měla být schopná z více příznakových vektorů určit jeden příznakový vektor, charakterizující daný obličej na snímcích. Je potřeba takovou agregační síť sestavit, dále natrénovat a ověřit její funkčnost.

Tento obecný postup byl použit i v této práci a v následujících řádcích najdete podrobnosti ohledně jednotlivých kroků, výběru metod a popřípadě implementační detaily.

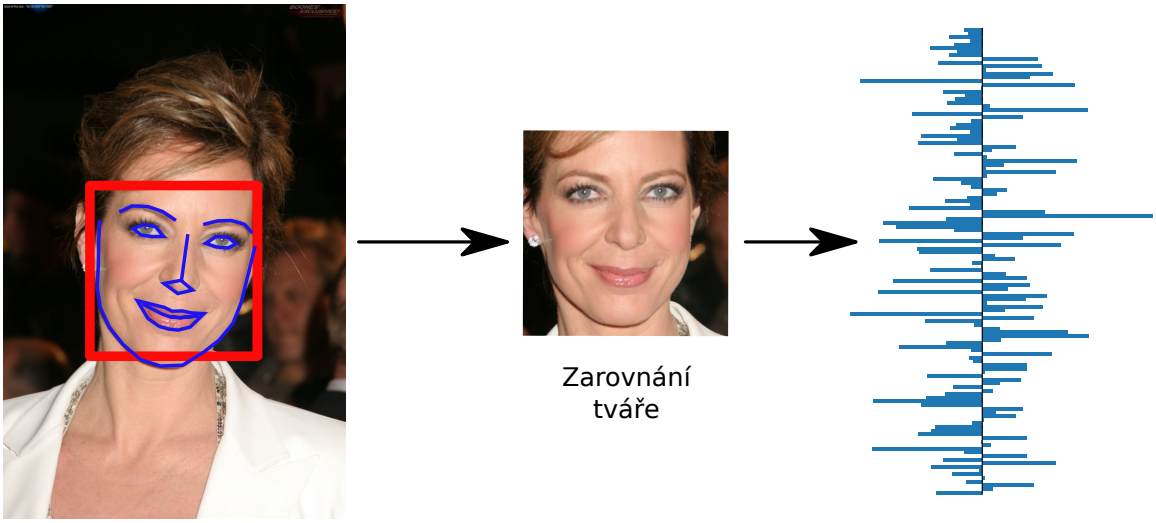
4.1 Datové sady

V této sekci jsou popsány použité video datasety pro trénování a také ověřování funkcionality neuronových sítí. Na konci této sekce je zde popsán způsob přípravy dat Kromě níže zmíněných datových sad existuje například video dataset *IARPA Janus Benchmark A* (*IJB-A*), který však se svou velikostí (cca 20 000 snímků z 2 000 videí) je poměrně malým datasetem. Níže vybrané datasety nabízejí daleko větší počet snímků na video a také obsahují méně anotačních chyb.

UMDFaces dataset. Tento dataset [2] byl představen v roce 2016 a obsahuje jak obrázky, tak videa. Konkrétně obrázková část pak obsahuje 367 888 obrázků od 8 277 různých osob. Video dataset je tvořen z 3 735 476 video snímků, které byly extrahovány z 22 075 videí, které byli posbírány z YouTube. Video dataset je tvořen z 3 107 různých identit.Tento dataset obsahuje obrázky a videa v poměrně dobré kvalitě, například oproti YTF datasetu. Dále se při tvorbě tohoto datasetu zaměřili na co nejvíce rozmanitější náklony a pozice obličejů na obrázcích, takže daný dataset je vhodný pro trénování sítí.

YTF dataset. Tento video dataset [35] je tvořen z 3 425 videí od 1 595 různých osob. Celkově je tento dataset tvořen přibližně 620 000 snímky. Jednotlivé videa jsou staženy z

Obrázek 4.1: Ukázka obrázků z datasetů. **nahoře:** dataset *UMDFaces*, **dole:** *YTF* dataset.



Detekce tváře a lokalizace významncých bodů

Extrakce příznakového vektoru

Obrázek 4.2: Ukázka postupu při extrahování příznakového vektoru pomocí *Dlib* knihovny.

YouTube, takže jednotlivé snímky jsou bohaté na různé pozice obličejů. Tento dataset se standardně používá při vyhodnocování úspěšnosti v oblasti rozpoznání obličejů.

Příprava dat. Pro přípravu dat je použita knihovna *Dlib*, která poskytuje jak natrénovanou síť na detekci tváře a lokalizaci 68 obličejových bodů, tak samotné zarovnání obličeje (viz obrázek 4.2).

4.2 Sítě pro extrakci příznakových vektorů

Jednou ze stěžejních částí celého procesu rozpoznávání je i síť, která extrahuje příznaky z tváří na obrázku. Pokud by existovala dokonalá síť pro tuto extrakci, pak by nebyla potřeba agregovat vektory a stačil by použít pouze jeden zástupný snímek z videa pro reprezentaci celého videa. Nicméně taková síť neexistuje a proto se zabývám otázkou agregace.

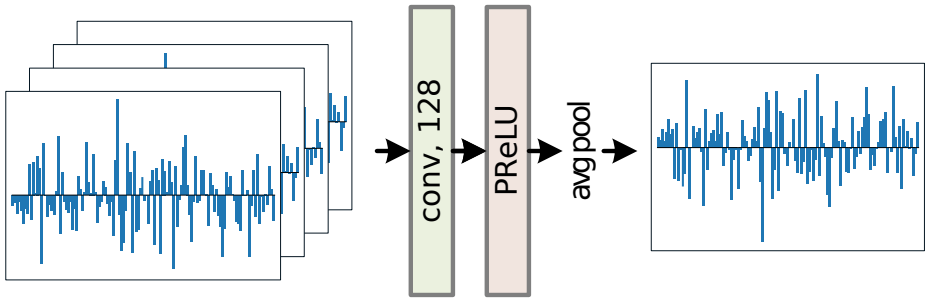
Cílem této práce není navrhnout ani ověřit nerunovou síť pro extrakci příznakových vektorů, a proto byly pro dané účely použity již natrénované neuronové sítě, které jsou veřejně dostupné. Bylo však potřeba ověřit zda tyto sítě jsou dostatečně kvalitní na získávání příznakových vektorů.

Výsledné příznakové vektory jsou v této práci vždy normalizovány a to *L2* normalizací, která je definován vztahem

$$y = \frac{x}{||x||_2} = \frac{x}{\sum_{i=1}^{n} x_i^2}, \tag{4.1}$$

kde y je výsledný normalizovaný vektor, x je vstupním vektorem, n je dimenze vektoru x.

VGG-Face. Jako první neuronová síť pro extrakci příznaků byla zvolena síť *VGG-Face*. Podrobná architektura této sítě je popsaná v sekci 3.1. Tato síť generuje příznakové vektory velikosti 2622, jelikož je zakončena plně propojenou vrstvou s 2622 umělými neurony.



Obrázek 4.3: Ukázka architektury agregační neuronové sítě *small*.

Dlib. Tato síť [15] pro extrakci příznakových vektorů je postavena na architektuře *ResNet* s 29 konvolučními vrstvami. Jedná se v podstatě o upravenou verzi sítě *ResNet-34*, která je podrobněji popsaná v sekci 3.1. Další úpravou, která byla provedena na sítí *ResNet-34*, je snížení počtu filtrů na polovinu u každé konvoluční vrstvy. Trénování této sítě probíhalo na datasetu s okolo 3 milióny obličejů, které pochází od 7 485 různých identit. Při trénování byla použita metoda *hard-negative mining*. Úspěšnost této natrénované sítě na *LFW* datasetu je 99.38%.Tato síť generuje příznakové vektory o velikosti 128.

4.3 Agregační metody

Jako hlavní agregační metodou byla zvolena neuronová síť s architekturou popsanou v článku *Learning Feature Aggregation in Temporal Domain for Re-Identification* [29] (dále tato síť bude označována jako síť LFTD). Architektura sítě je podrobně popsána v sekci 3.2. Jeden z hlavních důvodů je, že tato metoda nebyla zatím ověřena na rozpoznávání tváří či re-identifikací chodců. Výhodou této sítě je také její přístup k agregaci. Tato metoda ohodnocuje příznaky u jednotlivých snímků, které se projeví ve výsledném příznakovém vektoru.

Při implementaci této metody jsem provedl několik změn oproti originální *LFTD* síti. V originální síti jsou použity plně propojené vrstvy (viz obrázek 3.4), které zpracovávají jednotlivé snímky. Tyto vrstvy byli při implementaci nahrazeny konvolučními vrstvami (1D konvoluce s M filtry), které plní stejný účel jako zmíněné originální plně propojené vrstvy.

Další změnou, kterou jsem provedl na architektuře sítě *LFTD*, je použití $PReLU$, jakožto aktivačních funkcí. Tato funkce je založena na přechodové funkci $ReLU$, která je popsána v sekci 2.1, avšak je doplněna o adaptivní parametry pro negativní část $ReLU$. Více o této metodě si lze přečíst v práci *K. He et al.* [11].

Jako další agregační metodou byla použita neuronová síť s velmi jednoduchou architekturou (viz obrázek 4.3). Tato neuronová síť se skládá pouze z jedné konvoluční vrstvy se 128 filtry, která zpracovává každý snímek zvlášť, a dále s aktivační funkcí $PReLU$. Poté jsou jednotlivé příznakové vektory průměrovány do jednoho vektoru. Tato metoda je dále označována jako metoda *small*.

Poslední metodou, která je použita při agregování příznakových vektorů je *pooling* metoda, která pouze vytváří průměrný příznakový vektor přes všechny snímky videa. Tato metoda je značena jako *avg*.

4.4 Trénování

V této sekci jsou popsány způsoby, jak se trénovaly dané sítě, na kterých datech, a blíže je zde popsaný *hard-negativ mining* algoritmus, který je použit při trénování sítí.

Trénovaní neuronových sítí probíhalo tak, jak je to normální v oblasti rozpoznávání tváří. Byla použita architektura siamských sítí, pro výpočet vzdálenosti dvou vektorů se použila *euklidovská metrika* a jako chybová funkce byla použita *contrastive loss*. Pozitivní a negativní páry v trénovacích datech byly vždy generovány ve stejném poměru a to 1:1. Počáteční hodnota *learning rate* byla stanovena na 0.0001 a byl použit adaptivní optimalizátor *Adam*, který je podrobněji popsán v sekci 2.3.

Trénovací a validační data. Trénování agregačních sítí probíhalo na datech z *UMDFaces* datasetu, který byl rozdělen v poměru 2:1 na část trénovací a část validační. Validační část dat byla dále rozdělena v dalším poměru 2:1 na data A a B, kde data z třídy A se používali pro validaci v trénovacím procesu a data ze třídy B byli označené jako *unseen* a tyto data se použila pouze pro vyhodnocování daných metod, aby byla dosažena co nejreálnější hodnota přesnosti dané metody. Dataset *UMDFaces* byl rozdělen do těchto tříd náhodným výběrem identit.

Dataset *YTF* byl rozdělen v poměru 2:1 na data validační a data trénovací. Validační data sloužili stejně jako data *unseen* z *UMDFaces* datasetu na tzv.*benchmark* testy daných metod v porovnání s ostatními metodami. Trénovací data z *YTF* byli použiti na jemné doladění trénované neuronové sítě, aby daná síť mohla efektivně nakládat s daty jiné kvality (především jiného rozlišení). Validační data z *YTF* neobsahovala žádné identity, které se nacházely v trénovacích datech z *UMDFaces*.

Hard-negative mining. Při trénovaní neuronových sítí, tedy konkrétně agregačních modulů, bylo použito metody *hard-negative mining*, která zajišťuje vybírání učících dat při trénovací fázi a zefektivňuje tak učení. Data, která se používala při trénování, obsahoval průměrně 7 různých videí na identitu, což znamená, že daná síť je schopna projít všechny možné kombinace pozitivních párů při učení, avšak negativních párů byl daleko víc a tedy je síť nebyla schopna projít všechny při učení. Tato vlastnost trénovacích dat je celkem dobrá, protože nebude docházet k přetrénování (tzv. *overfitting*) daných sítí, avšak velká část negativních párů je snadno rozeznatelná (například při porovnání muže snědé pleti s ženou světlé pleti), a tedy by se daná neuronová síť nenaučila efektivně rozeznávat menší rozdíly mezi různými, avšak podobnými identitami. Z tohoto důvodu byla aplikována metoda *hard-negative mining* při trénovaní.

Tato metoda vyhodnocuje podobnost všech identit vůči všem identitám v trénovacím datasetu a na základě toho se sestavují pravděpodobnostní vektory pro každou identitu, které jsou použity při generování negativních párů na trénovaní. Po určitém počtu iterací dochází k novému sestavení těchto pravděpodobnostních vektorů. Díky této metodě nedochází k přetrénování sítě a zároveň dochází k efektivnímu učení nad těžšími páry, než by docházelo při náhodném výběru negativních dvojic.

Kapitola 5

Výsledky experimentů

V této kapitole jsou popsány experimenty za použití různých architektur neuronových sítí, které jsou blíže popsány v předešlé kapitole (4). Jednotlivé experimenty s výsledky jsou blíže popsány v daných sekcích.

Ze začátku jsou popsány experimenty nad natrénovanými neuronovými sítěmi, které slouží pro extrakci příznakových vektorů a dále jsou v této kapitole popsány experimenty s agregačními sítěmi. Za účelem co nejefektivnějšího přístupu byly z jednotlivých snímků z obou video datasetů předem extrahovány příznakové vektory, které sloužili pro trénování a testování agregačních neuronových sítí.

Na konci této kapitoly se nachází srovnání našich agregačních sítí s architekturami významných sítí v problematice rozpoznávání tváří ve videu. Toto srovnání probíhá na datasetu *YTF*.

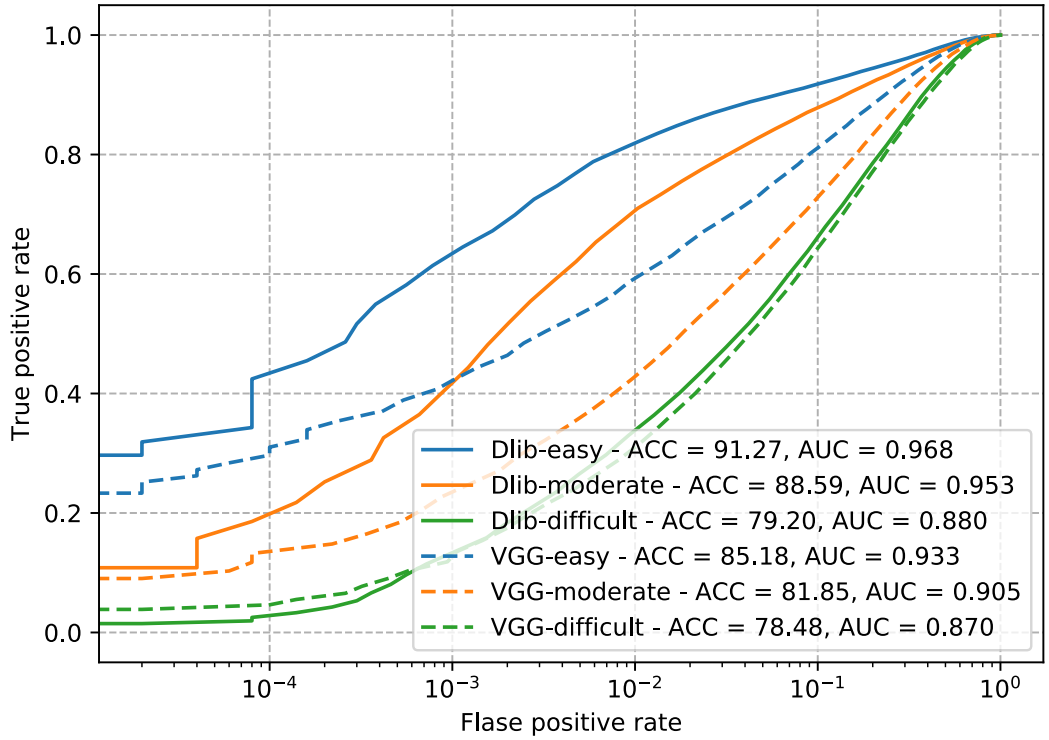
5.1 Srovnání sítí pro extrakci příznaků

V tomto experimentu se nachází srovnání dvou natrénovaných sítí, které slouží pro extrakci příznakových vektorů. Pro co nejrelevantnější výsledek byly použity testy ze stránek *UMDFaces*[1]. Tyto testy jsou rozděleny do třech skupin dle obtížnosti.

Jak bylo psáno v předchozí kapitole, byly použity sítě *Dlib* a *VGG-Face* (dále označovaná jako *VGG*). U první zmíněné sítě z knihovny (*Dlib*), bylo použito zarovnání obličeje dle příslušných významných bodů, nicméně u sítě *VGG* toto zarovnání neproběhlo. Důvodem k tomuto rozhodnutí byl předpoklad, že v příznakových vektorech o velikosti 2622 generovaných sítí *VGG* se nachází informace o náklonu tváře. Tato informace by mohla posloužit agregačním sítím při ohodnocování jednotlivých snímků a jejich příznaků a ve výsledku by tedy zarovnání nemuselo být potřeba.

Z výsledků tohoto experimentu (viz obrázek 5.1) lze si všimnout rozdílu mezi sítěmi *Dlib* a *VGG* a to hlavně v oblasti s menší změnou pózy tváře. Tento rozdíl je především způsoben 2D zarovnáním obličejů u *Dlib* sítě. Jak si lze dále povšimnout, tak při velkých natočeních tváří (test *difficult*) jsou neuronové sítě velmi vyrovnané. Nejspíše je to způsobeno tím, že 2D zarovnání již nedokáže takové velké natočení tváře eliminovat (jedná se například o profil).

[1] http://www.umdfaces.io/



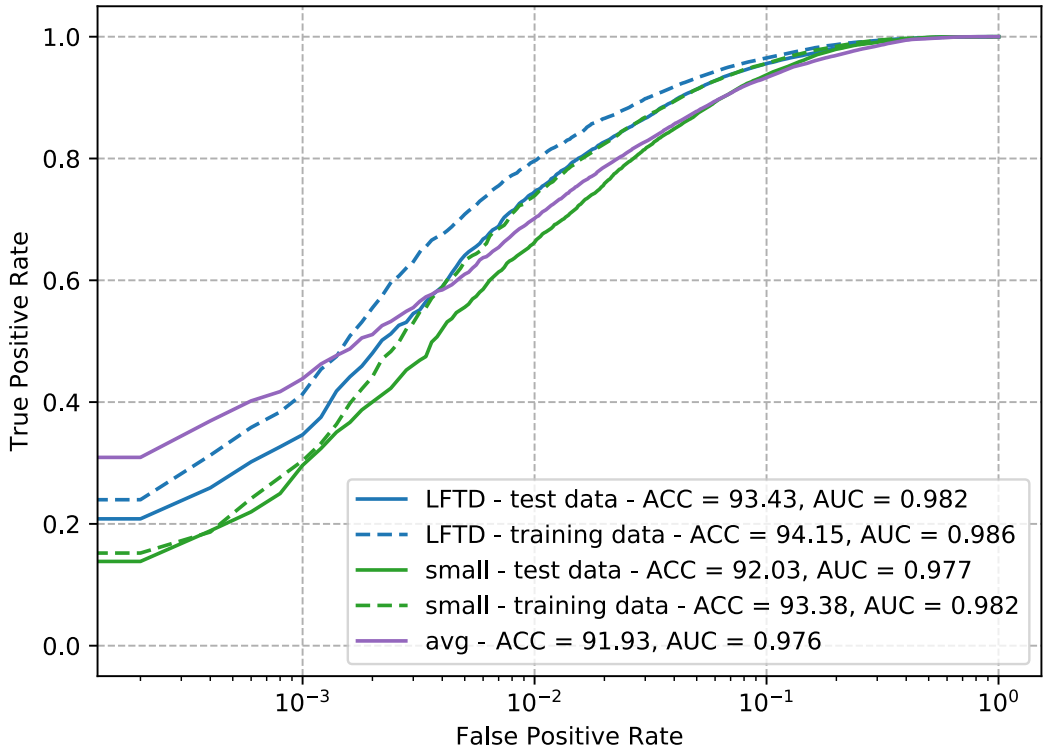
Obrázek 5.1: Srovnání neuronových sítí určených pro extrakci příznakových vektorů. Byli použity testy nad datasetem *UMDFaces*. Testy jsou rozděleny do třech skupina podle míry natočení hlavy na jednotlivých obrázcích.

5.2 Experimenty s agregačními metodami

Pro experimenty s agregováním příznakových vektorů byly vybrány dvě architektury neuronových sítích, které jsou více popsány v předchozí kapitole (4.3). Agregační síť postavená na architektuře *LFTD*, bude v experimentech označována jako *LFTD* a další agregační sítí je jednoduchá síť popsaná taktéž v předešle kapitole, konkrétně pak síť s jednou konvoluční vrstvou. Tato sít bude v experimentech dále označována jako *small*. Poslední agregační metodou, která je také použita v experimentech je průměrování sloupců přes všechny snímky. Tato agregační metoda je dále označována jako *avg*.

Při agregování příznakových vektorů může vyvstat otázka kolik vektorů je potřeba pro co nejlepší výsledky agregačních metod, nebo jaký vliv má na učení jednotlivých agregačních sítí hodnota *margin*, která je nastavitelná v chybové funkci *contrastive loss*. Na tyto otázky a další bych chtěl v této sekci odpovědět. Začneme však s otázkou, zda trénovací dataset je dostatečně velký.

Trénování. Jednou z důležitých otázek při trénování neuronových sítí je zda, trénovací dataset je dostatečně velký na samotné trénovaní neuronových sítí. Pokud by totiž trénovací dataset neobsahoval dostatečné množství dat, mohlo by se stát, že by se neuronové sítě přetrénovaly (tzv. *overfitting*) a na validačních datech by prokazovaly menší úspěšnost než na datech trénovacích.

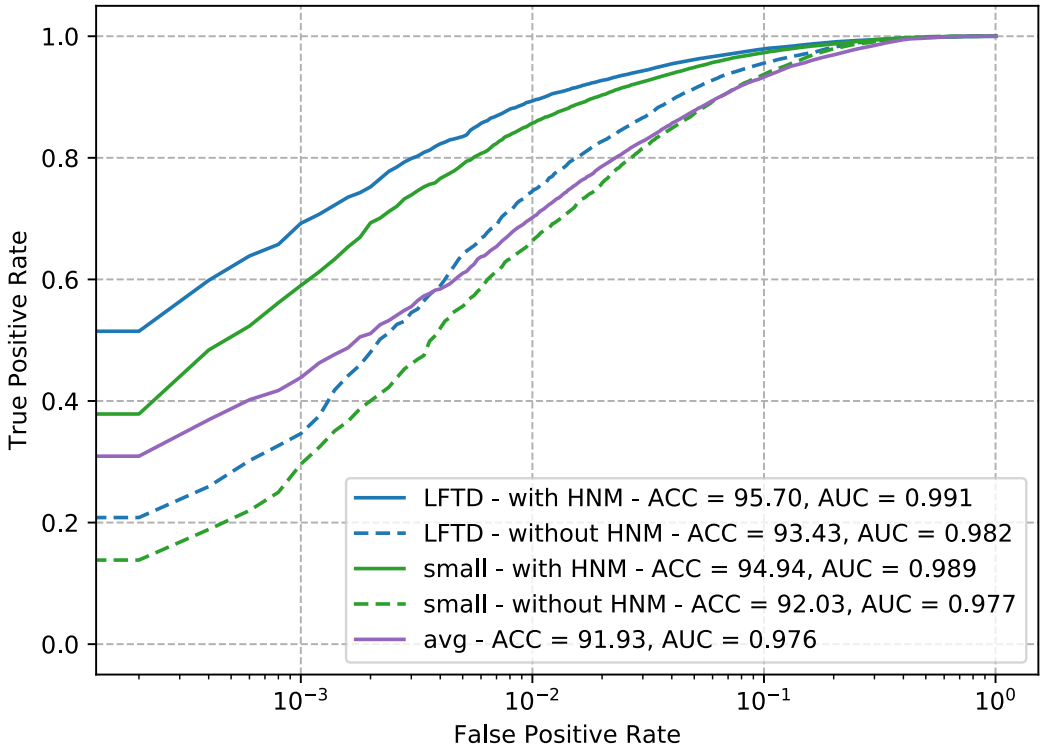


Obrázek 5.2: Porovnání úspěšnosti agregačních neuronových sítí na testovacích a trénovacích datech. Pro srovnání je zde uveden i výsledek agregační metody *avg*.

Z obrázku 5.2 je zřejmé, že k výraznému přetrénování u agregačních neuronových sítí nedochází, avšak k překrytí dvou křivek nedošlo. Lze tedy považovat velikost trénovacího datasetu za uspokojivou.

Jedna z dalších informací, kterou je dobré znát, je časová náročnost daných metod při učení. Doba učení jednotlivých metod se moc neodvíjela od použité agregační metody, avšak od toho jaké délky byly použity příznakové vektory. Konkrétně pak při použitý příznakových vektorů extrahovaných knihovnou *Dlib* byla jejich kompaktnost příslibem rychlé manipulace s trénovacími daty, což se také odrazilo v rychlosti trénování. Agregační sítě nad těmito příznakovými vektory dosahovaly rychlosti 25 - 35 sekund na učící epochu (cca 10 000 trénovacích párů). V porovnání s příznakovými vektory generovanými sítí *VGG* byla trénovací rychlost 60 - 80 sekund na epochu. Tyto časy nejsou nikterak velké, avšak při použití metody *hard-negative mining* v průběhu trénování se časová náročnost výrazně zvýší. Konkrétně jedna iterace ohodnocování podobnosti identit trvá přibližně 1.5 hodiny u *Dlib* vektorů a 3.5 hodiny u *VGG* vektorů, s tím, že dochází k ohodnocování identit 10× v průběhu trénování. Celková doba trénování agregačních metod je tedy přibližně 16 hodin u příznakových vektorů vygenerovaných *Dlib* sítí a 37 hodin u příznakových vektorů *VGG* sítě.

Učení s hard-negative miningem. Na obrázku 5.2 je dále vidět, že agregační metoda *avg*, je přesnější ve správně určených pozitivních vzorcích, při nízkém *false positive rate*, než agregační neuronové sít. Tento jev může být způsoben výskytem rozdílných dvojic identit, které jsou si podobné, avšak sítí jsou vyhodnoceny jak stejné. Aby se neuronové sítě naučili zacházet s podobnými identitami, tak byla použita při trénování metoda *hard-negative*



Obrázek 5.3: Porovnání agregačních neuronových sítí, které byli učené metodou *hard-negative mining*.

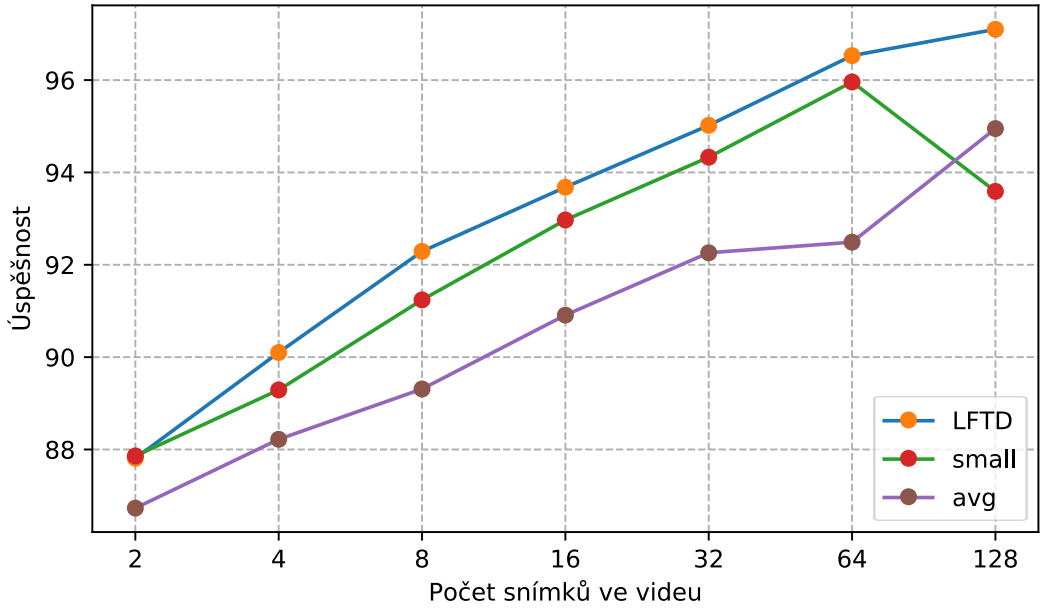
mining (v experimentech označená jako *HNM*). Tento přístup k zefektivnění trénovací fáze dokáže ohodnotit v jedné iteraci až $\frac{1}{7}$ celkového počtu trénovacích videí (konkrétně pro dataset *UMDFaces*). Toto číslo je závislé na počtu videí od každé identity.

Použitím této metody při trénovaní došlo ke značnému zlepšení (viz obrázek 5.3). V dalších experimentech se bude vždy při trénování používat metoda *hard-mining*.

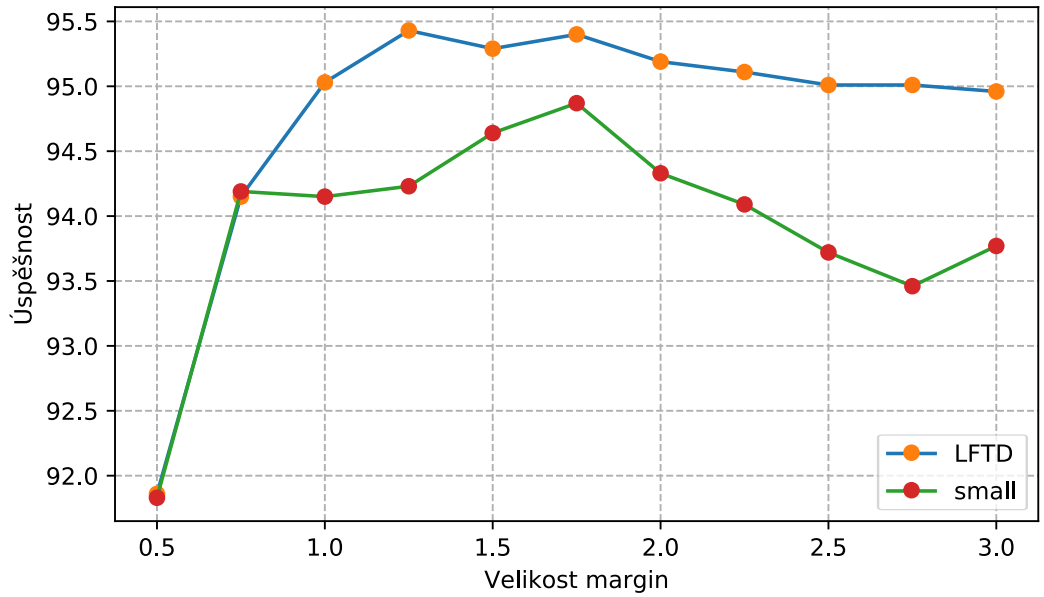
Počet snímků ve videu. Další zkoumanou vlastností je závislost počtu snímků ve videu na úspěšnosti agregačních metod. Z logického hlediska je pravděpodobnější, že čím více snímků má konkrétní agregační metoda k dispozici, tím více informací má o dané identitě na videu.

Z obrázku 5.4, lze vidět, že s exponenciálním růstem počtu snímků ve videu roste lineárně i přesnost agregačních metod. Tento lineární růst přesnosti za použití vždy dvojnásobného počtu snímků je především způsoben tím, že video obsahuje snímky z podobného prostředí a tedy lineární přidání snímků nepřináší lineární informaci daným agregačním metodám. U snímků videa nedochází k rychlým změnám prostředí a ani osvětlení, tak jako například u jednotlivých obrázků daných osob. Oproti ostatním metodám je metoda *LFTD* velmi konzistentní. Pro další experimenty se bude pracovat vždy s 32 snímky na video.

Velikost hodnoty *margin*. Mezi další experimenty, které byli provedeny při trénovaní agregačních neuronových sítí, byly i experimenty s nastavitelnou hodnotou *margin* v chybové funkci *contrastive loss*. Tato chybová funkce je podrobněji vysvětlena v sekci 2.3. Hodnota *margin* udává ideální vzdálenost dvou negativních identit. Z tohoto hlediska je logické, že čím vyšší je tato hodnota, tím přesnější bude natrénovaná neuronová síť.



Obrázek 5.4: Zobrazení závislosti počtu snímků ve videu na přesnosti agregačních metod.



Obrázek 5.5: Zobrazení závislosti velikosti *margin* u chybové funkce *contrastive loss* na úspěšnosti agregačních neuronových sítí.

Avšak dle obrázku 5.5, je patrné, že v přibližné hodnotě $m \approx 1.75$ se snižuje přesnost neuronových sítí s rostoucím *margin*. Tento jev je především způsoben optimalizací chybové funkce a chybovou funkcí samotnou, jelikož chybová funkce přiřazuje větší váhu negativním párům, než párům pozitivním. Dále byla nastaveny hodnota *margin* na 1.75.

5.3 Výsledky na testovacích datasetech

Vyhodnocení probíhalo na datasetech *YTF* a *UMDFaces*. Video dataset *UMDFaces* nelze porovnat s jinými metodami, protože nebyli nalezeny experimenty na tomto vcelku mladém datasetu. Naproti tomu dataset *YTF* je považován za takzvaný *benchmark* dataset, na kterém existuje spoustu výsledků různých metod.

Testování probíhalo nad identitami, které byli rozdílné od trénovacích identit. Dále se vybral 5000 náhodných pozitivních a 5000 náhodných negativních párů, nad kterými probíhalo vyhodnocení. Toto vyhodnocení bylo opakováno 10× a výsledný graf byl průměrem jednotlivých grafů. Účelem tohoto testování je dostat co nejrelevantnější informaci o přesnosti daných metod.

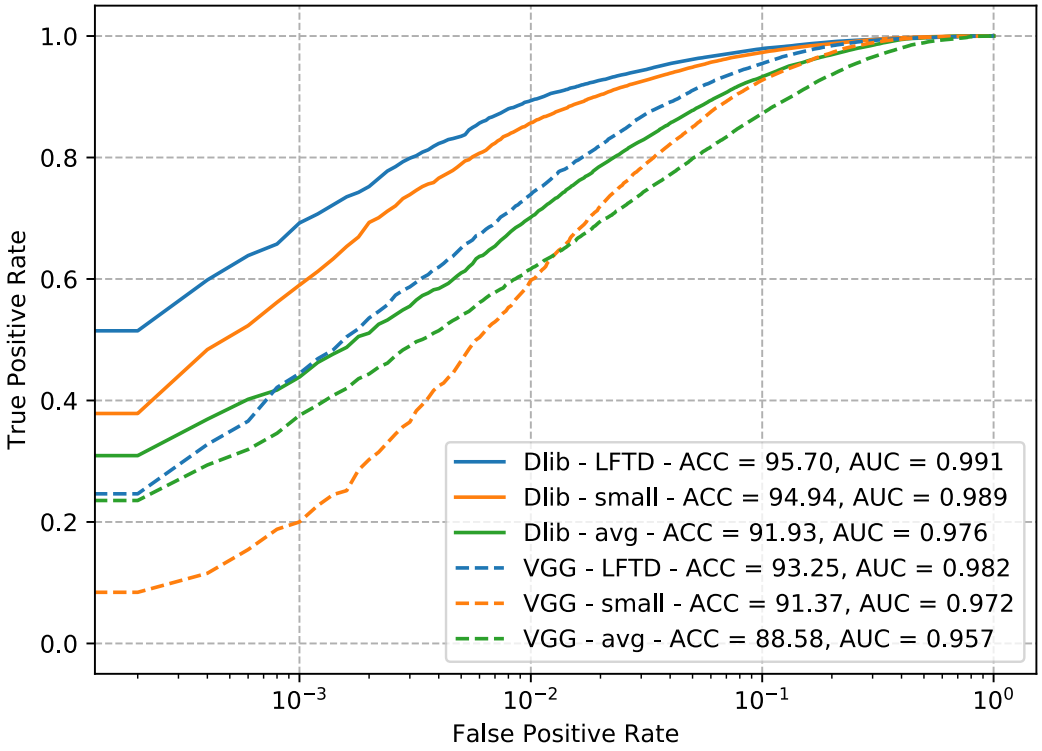
UMDFaces. Výsledky metod nad datasetem *UMDfaces* je zobrazen pomocí ROC křivek na obrázku 5.6. Nejlépe se umístila agregační metoda *LFTD*, která k agregaci využívala příznakové vektory z *Dlib* sítě. Její přesnost na tomto datasetu je $100\% - EER = 95.70\%$. Přesnost této metody za použití příznakových vektorů ze sítě *VGG* je nižší, konkrétně je $100\% - EER = 93.25\%$, avšak stále lepší než pooling metoda nad zarovnanými obličeji ze sítě *Dlib*.

Z tohoto grafu lze říci, že zarovnání obličejů při extrahování příznakových vektorů značně přispěje k celkové přesnosti rozpoznávání tváří na videu.

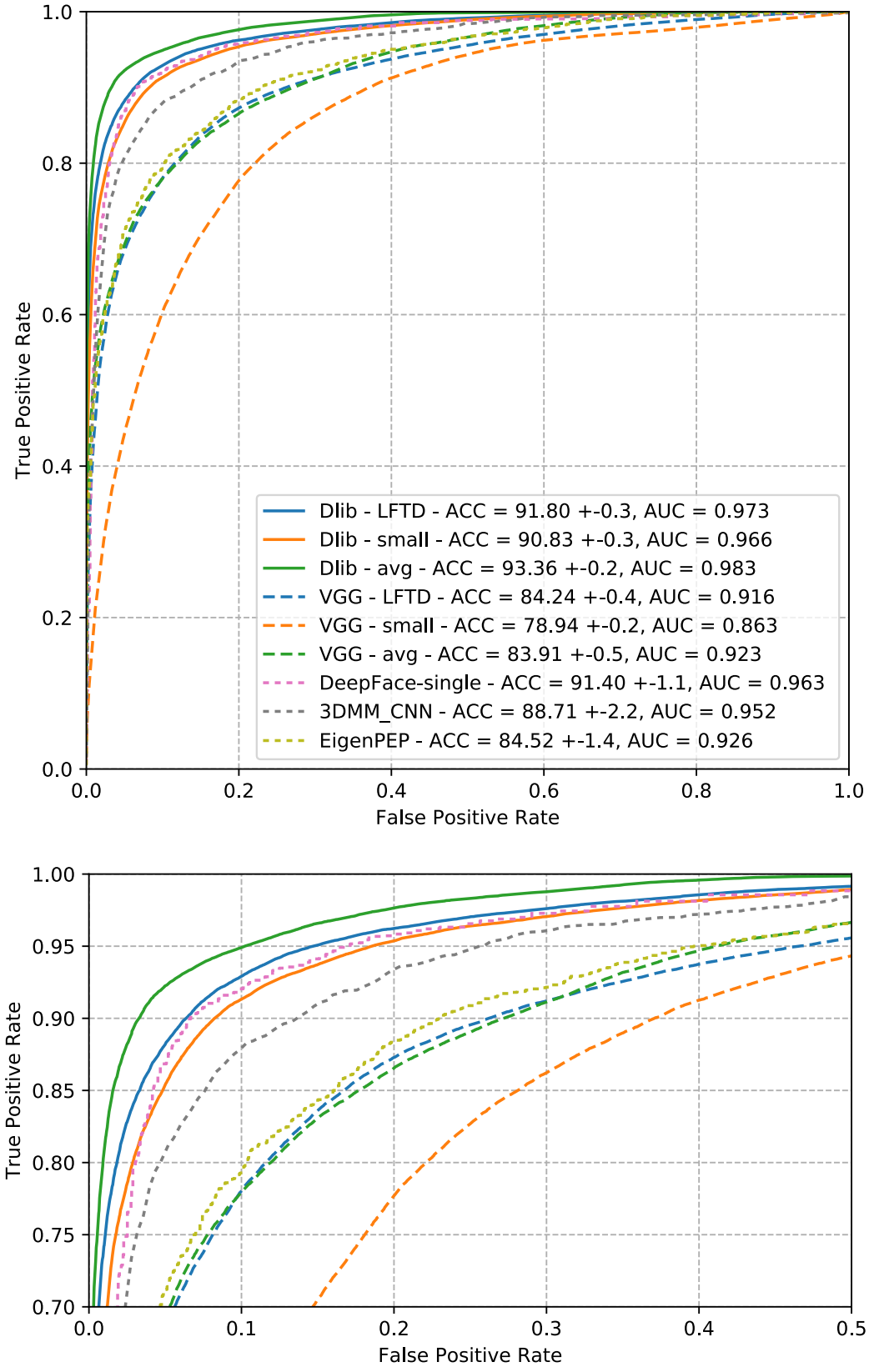
YouTube Faces. Výsledky použitých agregačních metod v porovnání s ostatními metodami je zobrazen na obrázku 5.7. Mezi *state-of-the-art* řešení byly vybrány 3 metody. Jako první byla vybrána neuronová síť *DeepFace* [32], která je podrobněji popsaná v sekci 3.1. Další metoda, která byla použita pro srovnání, je i metoda publikovaná v *Eigen-PEP* [18], která je zástupcem metod pro rozpoznávání tváří bez použití neuronových sítí. Jako poslední byla ke srovnání zvolena síť *CNN-3DMM* [33]. Tato metoda vytváří pomocí konvoluční neuronové sítě 3D modely tváří, která poté navzájem porovnává.

Agregační metody využívající vektory generované sítí *Dlib* se zarovnáním obličejů jsou srovnatelné s metodami *state-of-the-art*, nicméně metody využívající příznakové vektory ze sítě *VGG* jsou značně podprůměrné, avšak stále silně lepší než metody, které nevyužívají neuronových sítí pro extrakci příznakových vektorů.

Jako nejpřesnější agregační metodou je na daném datasetu vyhodnocena pooling metoda *avg*, s úspěšností $100\% - ERR = 93.36$ se standardní odchylkou 0.2%. Výrazný úspěch této pooling metody je nejspíše z důvodu velmi kvalitní sítě pro extrakci příznakových vektorů a také tím, že agregační sítě se při mírném „dotrénování“ nestihly přizpůsobit na jiná data.



Obrázek 5.6: Výsledky nad daty z datasetu *UMDFaces*.



Obrázek 5.7: Výsledky nad daty z datasetu *YTF* v porovnání s dalšími metodami.

Kapitola 6

Závěr

Hlavním cílem této práce bylo navrhnout a implementovat metodu pro rozpoznávání tváří na videu.

Tento cíl byl rozdělen do dvou částí. První částí bylo nalezení vhodné konvoluční neuronové sítě pro extrakci příznakových vektorů a druhou částí bylo vytvořit agregační metodu, která ze sady příznakových vektorů vygeneruje právě jeden vektor reprezentující identitu v dané sekvenci snímků, neboli na videu.

Byla nalezena velmi dobrá neuronová síť pro extrakci příznakových vektorů a to síť z knihovny *Dlib*, jejíž přesnost na datasetu *LFW* je 99.38%. Tato síť obsahuje moduly s detekcí obličeje, lokalizací významných bodů, zarovnáním i ořezáním. Dále byla použita pro extrakci příznakových vektorů také neuronová síť *VGG-Face*, u které však nebyly připravené data zarovnány a to se poté ukázalo na přesnosti sítě při generování příznakových vektorů.

Dále se povedlo natrénovat agregační neuronovou síť, která je postavena na architektuře *LFTD*. Tato metoda byla srovnána na *YTF* datasetu se *state-of-the-art* metodami a dosáhla lehce lepších výsledků. Tato metoda byla také vyhodnocena nad daty z datasetu *UMDFaces*, což je poměrně nový video dataset. Při trénování těchto agregačních sítí se používalo i metody *hard-mining* pro generování negativních trénovacích párů.

Z pohledu budoucího vývoje by bylo vhodné, optimalizovat trénování neuronových sítí, konkrétně pak čas, který byl zabrán ohodnocováním identit pro metodu *hard-negative mining*. Tento čas by šel výrazně zkrátit a to například při průběžném ohodnocování jednotlivých identit vůči sobě. Dále by bylo vhodné otestovat agregační metody nad obrázkovými datasety jako například *LFW*. A v neposlední řadě by nebylo špatné získat více trénovacích dat, nebo vytvořit vlastní video dataset, který by obsahoval více videí (i kratších) na jednu identitu.

Literatura

[1] Bahdanau, D.; Cho, K.; Bengio, Y.: Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Bansal, A.; Nanduri, A.; Castillo, C. D.; aj.: Umdfaces: An annotated face dataset for training deep networks. 2017: s. 464–473.

[3] Boureau, Y.-L.; Ponce, J.; LeCun, Y.: A theoretical analysis of feature pooling in visual recognition. 2010: s. 111–118.

[4] Dahl, G. E.; Sainath, T. N.; Hinton, G. E.: Improving deep neural networks for LVCSR using rectified linear units and dropout. 2013: s. 8609–8613.

[5] Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; aj.: Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, ročník 32, č. 9, 2010: s. 1627–1645.

[6] Forsyth, D. A.; Ponce, J.: *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002, ISBN 0130851981.

[7] Fukushima, K.: Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron. *IEICE Technical Report, A*, ročník 62, č. 10, 1979: s. 658–665.

[8] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] Hadsell, R.; Chopra, S.; LeCun, Y.: Dimensionality reduction by learning an invariant mapping. ročník 2, 2006: s. 1735–1742.

[10] Haykin, S. S.: *Neural networks : a comprehensive foundation*. Prentice Hall, 1998, ISBN 81-7808-300-0.

[11] He, K.; Zhang, X.; Ren, S.; aj.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 2015: s. 1026–1034.

[12] He, K.; Zhang, X.; Ren, S.; aj.: Deep residual learning for image recognition. 2016: s. 770–778.

[13] Huang, G. B.; Ramesh, M.; Berg, T.; aj.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. 2007.

[14] Jain, A. K.; Li, S. Z.: *Handbook of face recognition*. Springer, 2011.

[15] King, D.: *High Quality Face Recognition with Deep Metric Learning*. 2017, [Online; navštíveno 2.05.2018].
URL http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html

[16] Kingma, D. P.; Ba, J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] LeCun, Y.; Bengio, Y.; Hinton, G.: Deep learning. *nature*, ročník 521, č. 7553, 2015: str. 436.

[18] Li, H.; Hua, G.; Shen, X.; aj.: Eigen-pep for video face recognition. 2014: s. 17–33.

[19] McCulloch, W. S.; Pitts, W.: A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, ročník 5, č. 4, 1943: s. 115–133.

[20] McLaughlin, N.; del Rincon, J. M.; Miller, P.: Recurrent convolutional network for video-based person re-identification. 2016: s. 1325–1334.

[21] Munakata, T.: *Fundamentals of the new artificial intelligence*. Springer, 1998, ISBN 978-1-84628-838-8.

[22] Nair, V.; Hinton, G. E.: Rectified linear units improve restricted boltzmann machines. 2010.

[23] Panfilov, P.: *Introduction to neural networks*. 2001, [Online; navštíveno 18.04.2018].
URL http://download.virtuosclub.ru/lib/nyeyroni.pdf

[24] Parkhi, O. M.; Simonyan, K.; Vedaldi, A.; aj.: A compact and discriminative face track descriptor. 2014: s. 1693–1700.

[25] Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; aj.: Deep Face Recognition. ročník 1, č. 3, 2015: str. 6.

[26] Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, ročník 65, č. 6, 1958: str. 386.

[27] Schroff, F.; Kalenichenko, D.; Philbin, J.: Facenet: A unified embedding for face recognition and clustering. 2015: s. 815–823.

[28] Shang, Y.; Wah, B. W.: Global optimization for neural network training. *Computer*, ročník 29, č. 3, 1996: s. 45–54.

[29] Sochor, J.; Špaňhel, J.; Herout, A.; aj.: Learning Feature Aggregation in Temporal Domain for Re-Identification. *to be published in ECCV 2018*, 2018.

[30] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, ročník 15, č. 1, 2014.

[31] Sun, Y.; Liang, D.; Wang, X.; aj.: Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015.

[32] Taigman, Y.; Yang, M.; Ranzato, M.; aj.: Deepface: Closing the gap to human-level performance in face verification. 2014: s. 1701–1708.

[33] Tran, A. T.; Hassner, T.; Masi, I.; aj.: Regressing robust and discriminative 3D morphable models with a very deep neural network. 2017: s. 1493–1502.

[34] Viola, P.; Jones, M. J.: Robust real-time face detection. *International journal of computer vision*, ročník 57, č. 2, 2004: s. 137–154.

[35] Wolf, L.; Hassner, T.; Maoz, I.: Face recognition in unconstrained videos with matched background similarity. 2011: s. 529–534.

[36] Yang, J.; Ren, P.; Chen, D.; aj.: Neural aggregation network for video face recognition. *arXiv preprint*, 2017.

[37] Zeiler, M. D.: ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[38] Zhang, W.; Yu, X.; He, X.: Learning bidirectional temporal cues for video-based person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

Příloha A

Obsah přiloženého paměťového média

doc	obsahuje zdrojové soubory k technické dokumentaci bakalářské práce a již přeložené PDF
video	obsahuje video, které bylo součástí zadání práce
networks	obsahuje natrénované neuronové sítě
src	obsahuje zdrojové soubory, které byly použity k přípravě dat a trénování neuronových sítí
README.txt	textový soubor s podrobnějším popisem jednotlivých souborů na disku DVD