SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-72758

KLASIFIKÁCIA TVÁRÍ POMOCOU KONVOLUČNÝCH SIETÍ PRE ANDROID OS

DIPLOMOVÁ PRÁCA

2018 **Bc. Ján Polaček**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5384-72758

KLASIFIKÁCIA TVÁRÍ POMOCOU KONVOLUČNÝCH SIETÍ PRE ANDROID OS

DIPLOMOVÁ PRÁCA

Študijný program:	Aplikovaná informatika
Číslo študijného odboru:	2511
Názov študijného odboru:	9.2.9 Aplikovaná informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	Ing. Dominik Sopiak

Bratislava 2018 **Bc. Ján Polaček**

Slovenská technická univerzita v Bratislave
Ústav informatiky a matematiky

Fakulta elektrotechniky a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Autor práce:	Bc. Ján Polaček
Študijný program:	aplikovaná informatika
Študijný odbor:	9.2.9. aplikovaná informatika
Evidenčné číslo:	FEI-5384-72758
ID študenta:	72758
Vedúci práce:	Ing. Dominik Sopiak
Miesto vypracovania:	Ústav informatiky a matematiky FEI STU
Názov práce:	**Klasifikácia tvárí pomocou konvolučných sietí pre Android OS**
Jazyk, v ktorom sa práca vypracuje:	slovenský jazyk

Špecifikácia zadania:

Cieľom tejto práce je natrénovanie konvolučnej siete a jej aplikácie na problém klasifikácie ľudskej tváre v neriadených podmienkach. Ako prvé je dôležité naštudovať si problematiku konvolučných sietí. Následne je potrebné tieto znalosti aplikovať pri trénovaní a aplikácie konvolučnej siete na zariadení Android OS.

Úlohy:
1. Naštudovať teóriu konvolučnej siete.
2. Oboznámiť sa s problematikou klasifikácie tvárí.
3. Navrhnúť štruktúru konvolučnej siete.
4. Vytvoriť navrhnutú konvolučnú sieť.
5. Diskusia získaných výsledkov.

Literatúra:

- JAIN, A. *Handbook of Biometrics.* London: Springer-Verlag, 2008. ISBN 978-0-387-71040-2.
- GOODFELLOW, M. -- WILLIAMS, S. -- MORDAESKI, M. *Actinomycetes in Biotechnology.* San Diego : Academic Press, 1988. 501 p.
- Vincent Dumoulin, Francesco Visin: "A guide to convolution arithmetic for deep learning", Cornell University, 23. Mar 2016

Dátum zadania:	18. 09. 2017
Dátum odovzdania:	11. 05. 2018

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Bc. Ján Polaček
Diplomová práca:	Klasifikácia tvárí pomocou konvolučných sietí pre Android OS
Vedúci záverečnej práce:	Ing. Dominik Sopiak
Miesto a rok predloženia práce:	Bratislava 2018

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean et est a dui semper facilisis. Pellentesque placerat elit a nunc. Nullam tortor odio, rutrum quis, egestas ut, posuere sed, felis. Vestibulum placerat feugiat nisl. Suspendisse lacinia, odio non feugiat vestibulum, sem erat blandit metus, ac nonummy magna odio pharetra felis. Vivamus vehicula velit non metus faucibus auctor. Nam sed augue. Donec orci. Cras eget diam et dolor dapibus sollicitudin. In lacinia, tellus vitae laoreet ultrices, lectus ligula dictum dui, eget condimentum velit dui vitae ante. Nulla nonummy augue nec pede. Pellentesque ut nulla. Donec at libero. Pellentesque at nisl ac nisi fermentum viverra. Praesent odio. Phasellus tincidunt diam ut ipsum. Donec eget est. A skúška mäkčeňov a dĺžnov.

Kľúčové slová: Android

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Bc. Ján Polaček
Master's thesis:	Face classification using CNN for Android OS
Supervisor:	Ing. Dominik Sopiak
Place and year of submission:	Bratislava 2018

On the other hand, we denounce with righteous indignation and dislike men who are so beguiled and demoralized by the charms of pleasure of the moment, so blinded by desire, that they cannot foresee the pain and trouble that are bound to ensue; and equal blame belongs to those who fail in their duty through weakness of will, which is the same as saying through shrinking from toil and pain. These cases are perfectly simple and easy to distinguish. In a free hour, when our power of choice is untrammelled and when nothing prevents our being able to do what we like best, every pleasure is to be welcomed and every pain avoided. But in certain circumstances and owing to the claims of duty or the obligations of business it will frequently occur that pleasures have to be repudiated and annoyances accepted. The wise man therefore always holds in these matters to this principle of selection: he rejects pleasures to secure other greater pleasures, or else he endures pains to avoid worse pains.

Keywords: Android

Poďakovanie

I would like to express a gratitude to my thesis supervisor.

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek

ICA	Indenpendent Component Analysis
LDA	Linear Discriminant Analysis
LFA	Local Feature Analysis
NS	Neurónové siete
PCA	Principal Component Analysis
ReLU	Rectified linear unit
SGD	Stochastic Gradient Descent
SVM	Support Vector Machines

Zoznam algoritmov

Zoznam výpisov

Úvod

Tu bude krasny uvod s diakritikou atd.
A mozno aj viac riadkovy uvod.

1 Rozpoznávanie tváre

V tejto časti práce sa budeme venovať problematike rozpoznávania tváre, popíšeme ako je možné rozdeliť rozpoznávanie tváre z pohľadu porovnávania, ukážeme z akých krokov vo všeobecnosti pozostáva proces rozpoznávania tváre a nakoniec popíšeme niektoré známe techniky používané pri rozpoznávaní tváre.

Rozpoznávanie tváre je jedným z úkonov, ktoré človek robí pravidelne a bez námahy v každodennom živote. Široká dostupnosť silných počítačov za nízku cenu vytvára obrovský záujem o automatické spracovanie digitálneho obrazu v širokom spektre aplikácií, ako sú napríklad biometrická autentifikácia, monitorovanie osôb, interakcia s počítačom či spravovanie multimédií. Výskum a vývoj v oblasti rozpoznávania tvárí nasleduje tento trend automaticky.

Hlavnými výhodami využívania rozpoznávania tvárí voči iným biometrickým metódam ako napríklad odtlačky prstov či dúhovka, je ich prirodnené a nerušivé používanie, ale najmä možnosť použitia na väčšiu vzdialenosť. Zo šiestich biometrických metód (tvár, odtlačky prstov, dlaň, hlas, dúhovka, podpis) je podľa The International Civil Aviation Organization(ICAO)[1] rozpoznávanie tváre primárnou metódou pri kontrole identity na letisku.

Prvým automatickým systémov na rozpoznávanie tvárí bol podľa[2], systém navrhnutý Takeo Kanadem v jeho práci[3] z roku 1974. Za ním nasledovalo obdobie bez výraznejšieho pokroku v automatickom rozpoznávaní tváre, až do roku 1990, kedy Sirovich a Kirby zverejnili článok[4], v ktorom popisujú využitie nizko dimenzionálnych reprezentácií tváre odvodenej z Karhunen-Loevovej transformácie alebo Principal Component Analysis PCA. Podľa Jaina v[2] bol ďalším veľkým míľnikom práca[5] od Turka a Pentlanda na vlastných vektorov tváre (eigenface), ktorá znovu naštartovala výskum v oblasti rozpoznávania tváre. Medzi ďalšie míľniky tiež Jain zaraďuje prácu na Fisherovej metódae[6], ktorá aplikuje Linear Discriminant Analysis(LDA) po aplikovaní PCA k dosiahnutiu vyššej presnosti, výskum na lokálnych Gaborových filtroch[7] k dosiahnutiu efektívnejších príznakov tváre a prístup AdaBoost učenia, založeného na architektúra kaskádneho klasifikátora pre detekciu v reálnom čase[8].

Od obdobia kedy bola narhnutá Eigenface metóda nastal veľký pokrok v oblasti rozpoznávania tváre. V kontrolovaných podmienkach, kde je možné ovládať svetelnosť, postoj osoby či výraz tváre, prekonáva automatické rozpoznávania tváre ľudí, a to najmä pokiaľ databáza obsahuje veľké množstvo snímkov tváre. Napriek tomu rozpoznávanie

tváre stále čelí mnohým výzvam, najmä problematike rozpoznávania tváre v neriadených podmienkach.

1.1 Rozdelenie

Ako aj ostatné biometrické systémy, aj rozpoznávanie tváre funguje v jednom alebo oboch z režimov:

- Verifikácia (autentifikácia)
- Identifikácia tváre

Pri verifikácii tváre ide o porovnanie jedna k jednej, čo znamená, že jeden snímok tváre sa porovnáva len jedným záznamom identity v databáze, za ktorú sa prehlasuje. Typickým využitím tohoto režimu je samoobslužná kontrola identity prostredníctvom elektronického pasu[2].

Identifikácia tváre zahŕňa porovnanie jedna k mnohým, čo znamená, že jeden snímok tváre sa porovnáva s viacerými záznamami identít v databáze a vyberie jednu[2]. V niektorých prípadoch využitia je postačujúce nájsť len najpodobnejšiu identitu. V iných prípadoch, ako napríklad sledovanie podozrivých osôb, je okrem nájdenia najpodobnejšej tváre potrebné zaviesť aj prah spoľahlivosti, a tie tváre ktoré dosiahli mieru podobnosti väčšiu ako je prah, sú zaznamenané.

Úspešnosť systému na rozpoznávanie tváre závisí vo veľkej miere na množstve variabilných faktorov, ako je osvetlenie, poloha tváre, mimika, vek, make-up, účes, brada či pohyb tváre. Na základe týchto faktorov Jain rozdeľuje[2] na 2 kategórie vzhľadom na miery spolupráce užívateľov:

- Scenár so spolupracujúcim užívateľom
- Scenár s nespolupracujúcim užívateľom

Prípad spolupracujúceho užívateľa je využívaný napríklad pri prihlasovaní do počítača, riadenie fyzického prístupu, elektronické pasy (e-passport), teda prípady kedy má užívateľ záujem spolupracovať na správnom zoznímaní tváre (napr. pod správnym zoznímaním tváre môžeme rozumieť napríklad snímok tváre z predu s neutrálnym výrazom a otvorenými očami) kvôli prístupu alebo povoleniu.

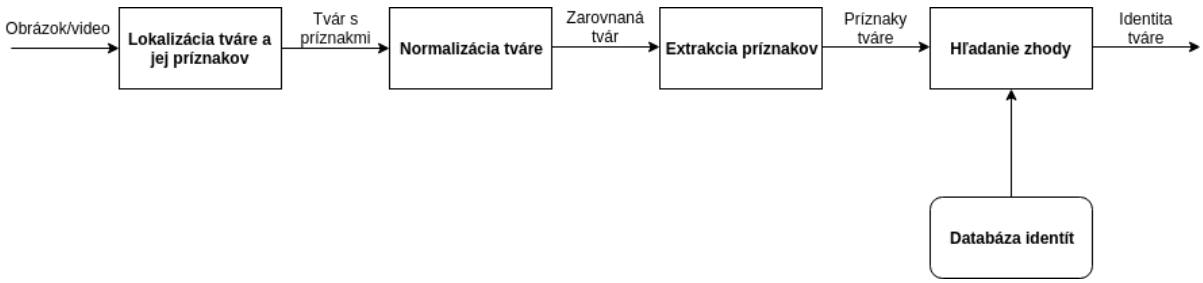
V prípade nespolupracujúceho užívateľa, ktorý je typické pre už spomínané sledovanie podozrivých, si osoba nie je vedomá toho, že podlieha identifikácii. Čo sa týka vzdialenosti medzi tvárou osoby a kamerou, sa v spolupracujúcom scenári využíva krátka vzdialenosť,

typicky o jedného metra, a ide teda o oveľa jednoduchšiu úlohu v porovnaní s identifikáciou nespolupracujúcej osoby často na väčšiu vzdialenosť.

1.2 Proces identifikácie

Jain popisuje rozpoznávanie tváre takto: "Rozpoznávanie tváre sa zaraďuje medzi problémi rozpoznávania vzorov, kde tvár, ktorá je reprezentovaná ako trojzormerný objekt, podlieha odlišnostiam vo svetle, postoji, výraze a iných faktoroch, potrebuje byt identifikovaná na zákle zozbieraných obrázkov"[2]. Zatiaľ čo roznávanie tváre z dvojrozmerného snímku je dnes bežne používané vo vačšine prípadov, v niektorých, najmä tých, ktoré vyžadujú vyššiu bezpečnosť, sa využíva trojrozmerný snímok tváre, prípadne snímky tváre mimo bežne viditeľného spektra - napr. termografický snímok tváre. Systém na rozpoznávanie tváre sa podľa Jaina[2] vo všeobecnosti skladá zo štyroch základných častí, ako je ukázané na obrázku1:

- Detekcia tváre a lokalizácia bodov tváre
- Normalizácia tváre
- Extrakcia príznakov
- Hľadanie zhody tváre



Obr. 1: Proces rozpoznávania tváre

Pri detekcií tváre ide primárne o oddelenie oblasti tváre od pozadia snímku. V prípade videa je potrebné sledovať (track) detekovanú tvár naprieč niekoľkými snímkami videa pomocou komponentu na sledovanie pohybu tváre. Zatiaľ čo detekcia tváre poskytuje len hruby odhad polohy a veľkosti tváre, lokalizácia bodov tváre najde už konkrétne časti tváre, ako sú napríklad oči, nos, obrys tváre a podobne. Lokalizácia bodov tváre je zväčša vykonanávaná osobitným komponentom na lokalizáciu bodov alebo komponentom na vyrovnanie (alignment) tváre.

Normalizácia tváre ide o normalizáciu tváre v geometrickom a fotometrickom zmysle. Tento krok je nevyhnutný, pretože sa od najnovších a najlepších (state-of-the-art) rozpoznávacích metód očakáva, že dokážu rozpoznať tvár v rôznych polohách a rôznom svetle. Geometrická normalizácia vykonáva transformáciu tváre do štandartného formátu snímku pomocou orezania (crop) tváre [2]. Snímok je následne zakrivený (warp) a upravený (morph) kvôli ešte lepšej a presnejšej normalizácií tváre [2]. Úlohou fotometrickej normalizácie je spracovanie snímku na základe osvetlenie či farebnej škály.

Extrakcia príznakov je vykonávaná na normalizovanom snímku tváre, s cieľom vybrať charakteristické infomácie, ktoré sú užitočné pri rozlišovaní tvárí rozdielnych osôb, pričom odolný voči odchýlkam v geometrickej a fotometrickej normalizácii [2]. Extrahované príznaky sú nasledne použité pri hľadaní zhody s identitou.

Pri hľadaní zhody tváre sa porovnávajú extrahované priznaky zo vstupnej tváre s jednou alebo viacerými tvárami ktoré už sú zapísané v databáze. Výsledkom porovnania s jednou tvárou je výsledkom odpoveď áno alebo nie (verifikácia). V prípade porovnávania s viacerými tvárami je výsledkom indetita vstupnej tváre, za predpokladu, že identita ktorá je nájdená presiahne prah spoľahlivosti, inak vráti informáciu, že tvár je neznáma. V súčastnosti je v tejto oblasti najväčšou výzvou nájsť spoľahlivé meranie vhodné na určenie podobnosti príznakov tváre.

Presnosť systému na rozoznávanie tváre vo veľkom závisí na správnosti extrahovaných príznakov tváre, ktoré zase závisia od správnej lokalizácii a normalizácii tváre.

1.3 Techniky rozpoznávania tvári

Zhao rozdeľuje[9] algoritmy rozponávania do dvoch základných kategórií, vzhľadom na spôsob extrakcie príznakov:

- Metódy založené na príznakoch (feature-based)
- Medódy založené na vzhľade (appearance-based)

Metódy založené na príznakoch využívajú rôzne vlastnosti a geometrické atribúty na popis tváre, ako sú napríklad vzdialenosti či uhly medzi bodmi tváre. Na druhej strane, metódy založené na vhľade využívajú globálne vlastnosti tvárového vzoru. Typickou črtou algoritmov založených na vhľade je výpočet bázových vektorov, na ktoré je následne tvár premietnutá. Koeficienty takejto projekcie sú potom použité k efektívnej reprezentácii údajov tváre[10]. Obľúbené algoritmy ako sú Principal Component Analysis PCA, Linear

Discriminant Analysis LDA, Indenpendent Component Analysis ICA, Local Feature Analysis LFA, Manifolds, Correlation Filters alebo Tensorfaces sú založené práve na vhľade tváre. Holistický prístup k rozpoznávaniu tváre má však často problémy pri rôznych polohách tváre[10].

1.3.1 Eigenfaces

Podstatou metódy Eigenfaces navrhnutej Turkom a Pentlandom[5], tiež známej ako PCA, je hľadanie najmenšej kvadratickej chyby lineárneho podpriestoru, ktorý mapuje dáta z originálneho N-rozmerného priestoru na M-rozmerný priestor príznakov, kde M « N . Týmto dosiahneme redukciu dát do M rozmeného priestoru využitím M vlastných vektorov matice kovariance, ktoré korešpondujú s najväčšími hodnotami vlastných čísel[10]. Konečné bázové vektory ktoré sú najvhodnejšie na popis dát, sú nájdené pri procese optimalizácie, ktorej podstatou je maximalizácia variancie premietnutých dát. Jain popisuje[10] proces výberu bázových PCA vektorov W optimalizačnou funkciou(1), kde S_T označuje úplne maticu rozptylu, ktorá obsahuje kovariancie dát tváre.

$$W_{PCA} = argmax|W^T S_T W| = [w, w2, \ldots, wm] \tag{1}$$

Obr. 2: Prvých 6 bázových vektorov Eigenfaces, té z[10, p. 45]

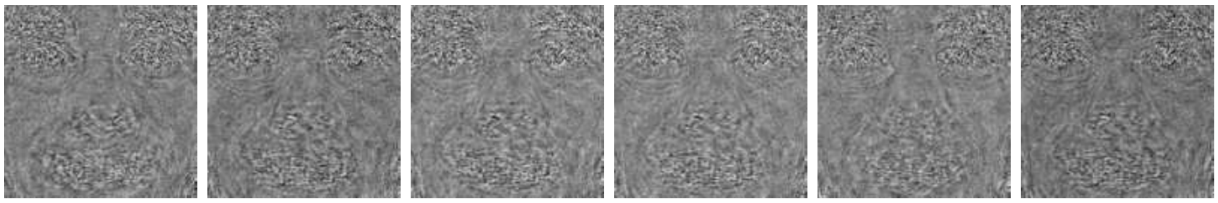
Na obrázku2 vidíme príklad vlastných vektorov vybraných metódou Eigenfaces, na obrázkoch po procese normalizácie. Metóda PCA je vhodná na reprezentáciu dát, čo neznamená, že je vhodná aj na rozdeľovanie do tried.

1.3.2 Linear Discriminant Analysis, Fisherfaces

Matóda LDA[11] je oveľa vhodnejšia k hľadaniu projekcií, ktoré dobre oddeľujú rozdielne triedy. Táto metóda je založená na hľadaní optimálnych projekčných vektorov, ktoré optimalizujú pomer medzitriednych (within class) a vnútrotriednych (between class) vzdialeností - maximalizuje oddelenie tried v premietnutom priestore. Optimálne bázové vektory LDA, môžu byť podľa[10] definované ako rovnica(2), kde S_B značí medzitriednu maticu rozptylu a kde S_W vnútrotriednu maticu rozptylu.

$$W_{LDA} = argmax \frac{|W^T S_B W|}{|W^T S_W W|} \tag{2}$$

Zvyčajne pri riešení problematiky rozpoznávania tváre (a väčšine ostatných problémov rozpoznávania vzorov obrázkov) je množstvo tréningových obrázkov menší ako počet pixelov (dimenzionality dát), čo znamená, že vnútrotriedová matica rozptylu S_W je singulárna, čo je problémom pre LDA. Na vyriešenie tohoto problému singulárnej matice sa na najprv vykonáva PCA na redukovanie dimenzionality dát a až následne sa aplikuje LDA v menej rozmernom podpriestore PCA. NA základe týchto zmien bolo dosiahnuté zlepšenie výsledkov v porovnaní s tradičnou PCA metódou. Projekčné vektory, ktorých príklad môžeme vidiet na obrázku3 Fisherfaces sú tie, ktorú spĺňajú maximalizujú vysledok optimalizačnej funkcie(3).



Obr. 3: Prvých 6 bázových vektorov Fisherfaces, prebraté z[10, p. 46]

$$W_{LDA} = argmax \frac{|W^T W_{PCA}^T S_B W_{PCA} W|}{|W^T W_{PCA}^T S_W W_{PCA} W|} \tag{3}$$

1.3.3 Indenpendent Component Analysis

Podstatou metódy ICA, je hľadanie takých neortogonálnych báz, aby boli na nich založené transormácie príznakov boli štatisticky nezávislé, zatiaľ čo metóda PCA hľadá také ortogonálne bázy tváre, aby príznaky po transformácii neboli korelované[10]. Bázové vektory tvárí metódy PCA sú založené len na štatisike druhého rádu. ICA zovšeobecňuje koncept PCA a vytvára tak model so vzťahmi štatisticky vyššie rádu. Pôvodná motivácia tejto metódy východza z potreby zaradiť zvukové vlny do nezávislých zdrojov, bez priamej znalosti procesu spájania (mixovania) týchto vĺn[10]. Bartlett vo svoje publikácií[12], v ktorej aplikoval metódu ICA na rozpoznávanie tváre uvádza, že ICA dosahuje oveľa lepšie výsledky v porovnaní s metódou PCA, pri rozponávaní tváre v rôznych častiach dňa a rôznych výrazoch tváre. prebra

1.3.4 Local Feature Analysis

LFA vytvára skupinu lokálne prepojených detektorov príznakov, založených na rozložení charakteristického podpriestoru. Vzniknú tak minimálne kolerované a topograficky indexované

podskupiny príznakov, ktoré definujú podpriestor záujmu. [10]

Lokálna reprezentácia vytvára odolnosť voči zmenám v lokalizovaných oblastiach objektov. Príznaky použité metódoou LFA sú menej náchylné na zmenú vo svetle a zároveň je vďaka nim jednoduchšie odhadnúť rotáciu objektu. Algoritmus LFA bol pouitý ako kľučový komponent algoritmu FaceIt, ktorý je jedným z komerčne využívaných systémov na rozpoznávanie tváre[10].

1.3.5 Neurónové siete a Support Vector Machines

Neurónové siete NS a Support Vector Machines SVM sú zvyčajne využívané v nízkodimenzionálnych priestoroch príznakov, najmä kvoli výpočtovej náročnosti spracovania, v prípade viacdimenzionálnych dát tváre[10]. Neurónové siete podliehaju širokému záujmu o výskum, najmä čo sa týka lepšej reprezentáciu príznakov tváre a rozpoznávania tváre. Avšak, so zvyšujúcim sa množstvom osôb, ktoré sa neurónová sieť učí rozpoznávať, narastá výpočtová zložitosť exponencionálne. Spojením viacerých neurónových sietí sa podarilo zlepšiť celkový výkon pri rozpoznávaní tváre.

Vo všobecnosti nevieme povedať, čo presne sa neurónová sieť naučila, alebo akým spôsobom bude fungovať. Neurónová sieť zvyčajne vyžaduje veľké množstvo tréningových dát kvôli dobrej generalizácii dát, s čím zároveň súvisí značné množstvo výpočtov uskotočnených v procese trénovania. SVM sa ukazuje ako úspešná metóda pri rozpoznávaní objektov, použitím kernelového triku (kernel trick), ktorý mapuje dáta do viacdimenzionálneho priestoru príznakov[10]. SVM v tomto priestore potom nájde nadrovinu, ktorá maximalizuje šírku hranice oddeľujúcej triedy klasifikácie, a znižuje tak riziko zlej klasifikácie nielen pre trénovacie dáta, ale tiež k dosiahnutiu lepšej generalizácie neznámych dát.[10]

Neurónovú siete v kombinácii s SVM použijeme v praktickej časti tejto práce, ktorou detekujeme príznaky tváre a následne klasifikujeme.

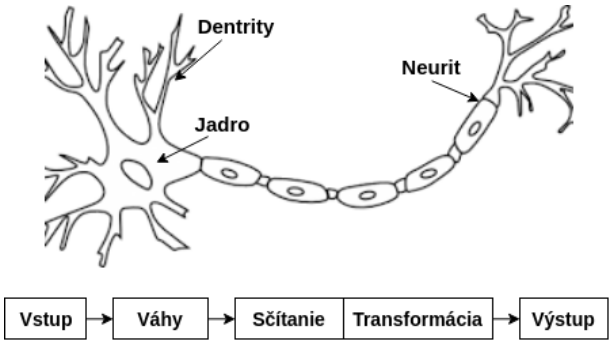
2 Neurónové siete

V tejto kapitole popíšeme neurónové siete, ktoré sme si zvolili ako techniku na rozpoznávanie tváre. Začneme hlavnou stavebnou jednotkou neurónových sietí - neurónom, následne pokračujeme teóriou trénovania neurónových sietí a optimalizáciám ako je stochastic gradient descent či regularizácie. Nakoniec tejto kapitoly popíšeme konvolučné neurónové siete a jej základné princípy, ktoré použijeme pri rozpoznávaní tváre v praktickej časti tejto práce.

2.1 Neurón

Základnou jednotkou z ktorej sa skladá mozog je neurón. Kúsok mozgu, o veľkosti malého zrnka ryže, obsahuje približne 10000 neurónov, pričom každý neurón vytvára v priemere 6000 prepojení s inými neurónmi, a ide teda v podstate o veľkú biologickú sieť prepojení, ktorá nám dovoľuje rozumieť svetu okolo nás[13].

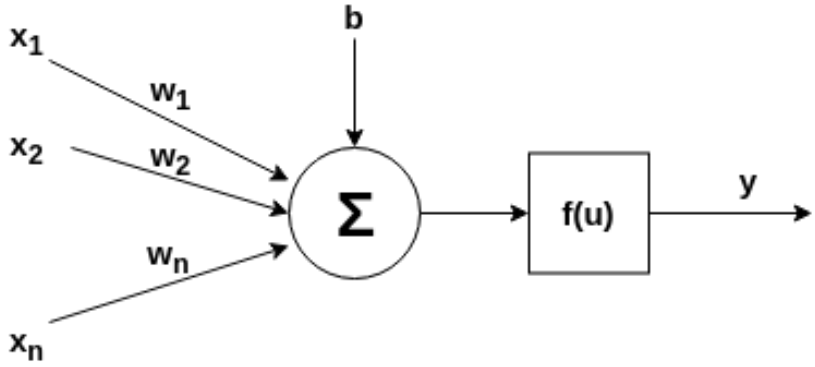
V samotnej podstate neurónu, prichádza k príjmu informácií z ostatných neurónov, ich spracovaniu a odoslaní výsledku spracovania ďalším bunkám. Neurón prijíma vstupné informácií pomocou dendritov, čo sú v výbežky na konči neurónu ktoré sa správajú ako ako antény a zachytávajú signály. Každé z týchto prepojení je dynamicky zosilnené alebo zoslabené, závislosti od frekvencie používania daného spojenia, pričom sila prepojenia definuje, aký veľký prínos má daný dendrit k výstupu neurónu[13]. Po zvážení sily(váhy) prepojení sú vstupné informácie sčítané v jadre neurónu a tranformované na nový signá(výsledok), ktorý je cez neurit odoslaný ďalším neurónom. Zjednodušený proces fungovania biologického neurónu vidíme na obrázku4.



Obr. 4: Zjednodušený popis fungovania biologického neurónu

Fungovanie biologického neurónu sa stalo inšpiráciou pre návrh umelého neurónu. Rovnako ako biologický neurón, aj umelý neurón prijíma vstupné údaje - x_1, x_2, ..., x_n ktoré sú násobené špecifickou váhou w_1, w_2, ..., w_n prislúchajúcou k neurónu. Váhované

vstupy sú podobne ako pri biologickom neuróne sčítané, je k ním pripočítaný základ(bias), a následne transormované aktivačnou funkciou, ktorej výsledok je odoslaný ďalším neurónom. Fungovanie neurónu sa dá zapísať ako rovnica(4), kde vstup, a jeho schému vidíme na obrázku5.



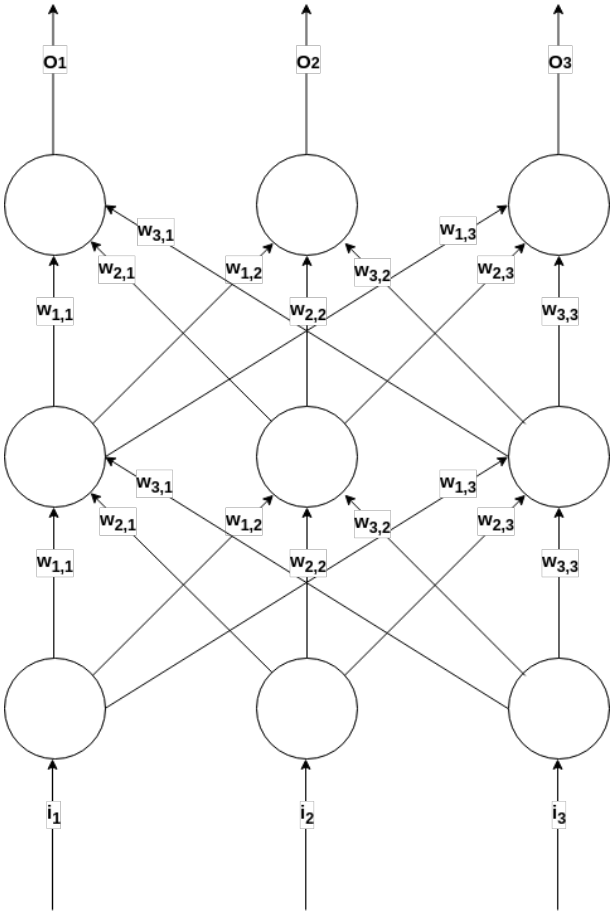
Obr. 5: Schéma fungovania umelého neurónu

$$y = f(\sum_{i=1}^{n} x_i w_i + b) \tag{4}$$

Jedným neurónom vieme aproximovať jednoduchú lineárnu funkciu. Na aproximáciu zložitejších problémov, napríklad na rozpoznanie ručne písaných znakov, ich však potrebujeme viac. Ľudský mozog má usporiadané neuróny vo vrstvách. Buduma tvrdí, že mozgová kôra, ktorá je zodpovedná za ľudské myslenie, je tvorená šiestimi takýmito vrstvami[13], a informácia sa prenáša z jednej vrstvy do druhy, až kým vstupná informácia nedáva zmysel. Príkladom je ľudské videnie, kde najspodnejšia vrstva prijíma informácie zo sietnice oka, táto informácia sa presúva vrstvami mozgovej kôry, až nakoniec v poslednej vrstve rozpozná, o aký predmet ide.

Aplikovaním rovnakého konceptu spájania neurónov do vrstiev, výtvárame umelé neurónové siete. Obrázok jednduchej neurónovej siete vidíme na obrázku6, kde spodná vrstva prijíma vstupné dáta a vrchná vrstva neurónov počíta finálny výsledok. Stredná vrstva neurónov je nazývaná skrytá vrstva, kde $w_{i,j}^{(k)}$ označuje váhu prepojania medzi í-tym v k-tej vrstve a j-tym neurónom v k+1 vrstve, pričom schopnosť neurónovej siete riešiť problémi priamo závisí na nájdení optimálnych hodnôt váh, pričom vďaka skrytej vrstve je možné vypočítať komplexnejšie vzťahy vo vstupných dátach. V ukážkovej sieti sú existujú prepojenia neurónov v smere do nasledujúcej vrstvy, bez prepojení medzi neurónmi v rovnakej vrstve alebo do predchádzajúcej, tento typ siete sa označuje ako

dopredná - feed forward. Lineárne neuróny sú nenáročné na výpočet, avšak akákoľvek dopredná sieť len s lineárnymi neurónmi môže byť vyjadrená ako sieť bez skrytej vrstvy, a teda je limitovaná na naučenie sa zložitejších vzťahov. Riešením je vnesenie nelineárnych vzťahov do neurónovej siete, prostredníctvom aktivačných funkcií.[13]

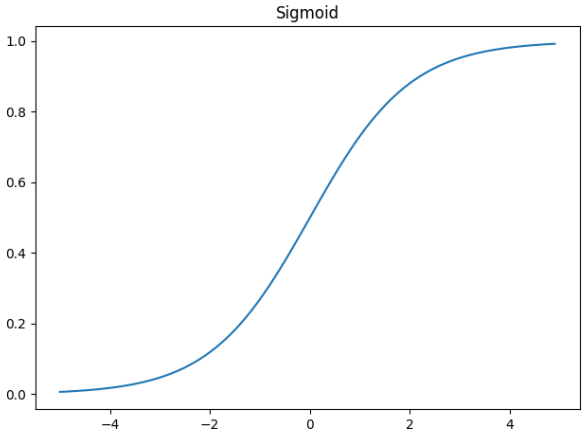


Obr. 6: Príklad doprednej neurónovej vrstvy skladajúcej sa z troch vrstiev a tromi neurónmi vo vrstve.

2.2 Aktivačné funkcie

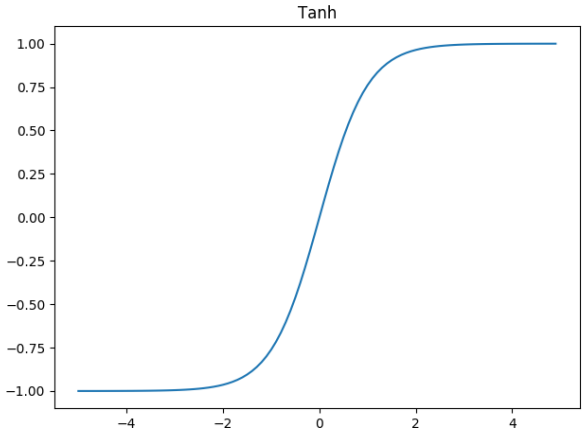
Poznáme 3 základné nelineárne aktivačné funkcie - sigmoid, tanh a relu. Sigmoidná funkcia, je definovaná ako(5), jej priebeh7 je podobný písmenu s, čo spôsobuje, že ak je vstup veľmi malý, výstup bude blízky 0, a pokiaľ je veľmi veľký, výsledok sa blíži k 1.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{5}$$



Obr. 7: Priebeh sigmoidnej funkcie

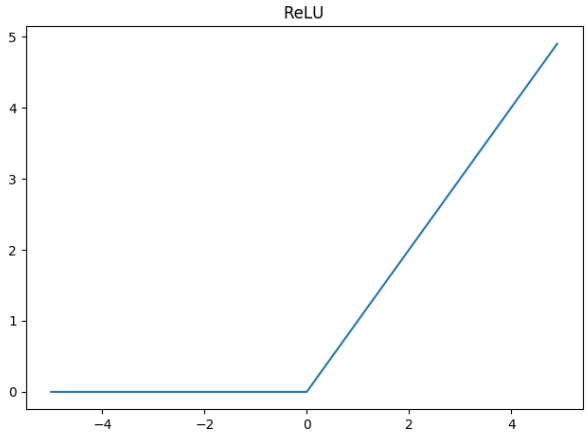
Funkcia tanh - hyperbolický tangens, je podobná sigmoidnej čo sa týka priebehu8, s tým že jej rozsah je od -1 do 1, má centrum priebehu v 0, kvôli čomu je často preferovanou aktivačnou funkciou.



Obr. 8: Priebeh funkcie tanh

Iným tipom nelineárnej funkcie je ReLU, vyjadrený rovnicou (6), a priebehom ako na obrázku9 a je používanou funkciou najmä v oblasti strojového videnia.

$$f(z) = max(0, z) \tag{6}$$



Obr. 9: Priebeh ReLU funkcie

2.3 Trénovanie neurónovej siete

Pod pojmom trénovania neurónovej siete rozumieme proces optimalizácie vektora váhových prepojení medzi neurónmi. Počas trénovania sa zvyčajne na vstup neurónovej siete prináša veľké množstvo tréningových vzoriek, pričom opakovane upravujeme hodnoty váh s cieľom minimalizovať chybné výsledky neurónovej siete. Chybu výsledku neurónovej siete vypočítame chybovou funkciou(objective function, error function loss function) (7), a kde E je výsledná chyba pre i-tu tréningovú vzorku, t^i označuje pravdivú hodnotu výsledku pre vzorku i a y^i označuje výstup neurónovej siete.

$$E = \frac{1}{2}\sum_i (t^{(i)} - y^{(i)})^2 \tag{7}$$

Výsledok chybovej funkcie je rovný nule, pokiaľ náš model neurónovej siete vracia perfektné výsledky pre každú trénovaciu vzorku. Cieľom trénovania je teda optimalizovať váhy neurónov tak, aby sa výsledok chybovej funkcie E čo najviac približoval k E.

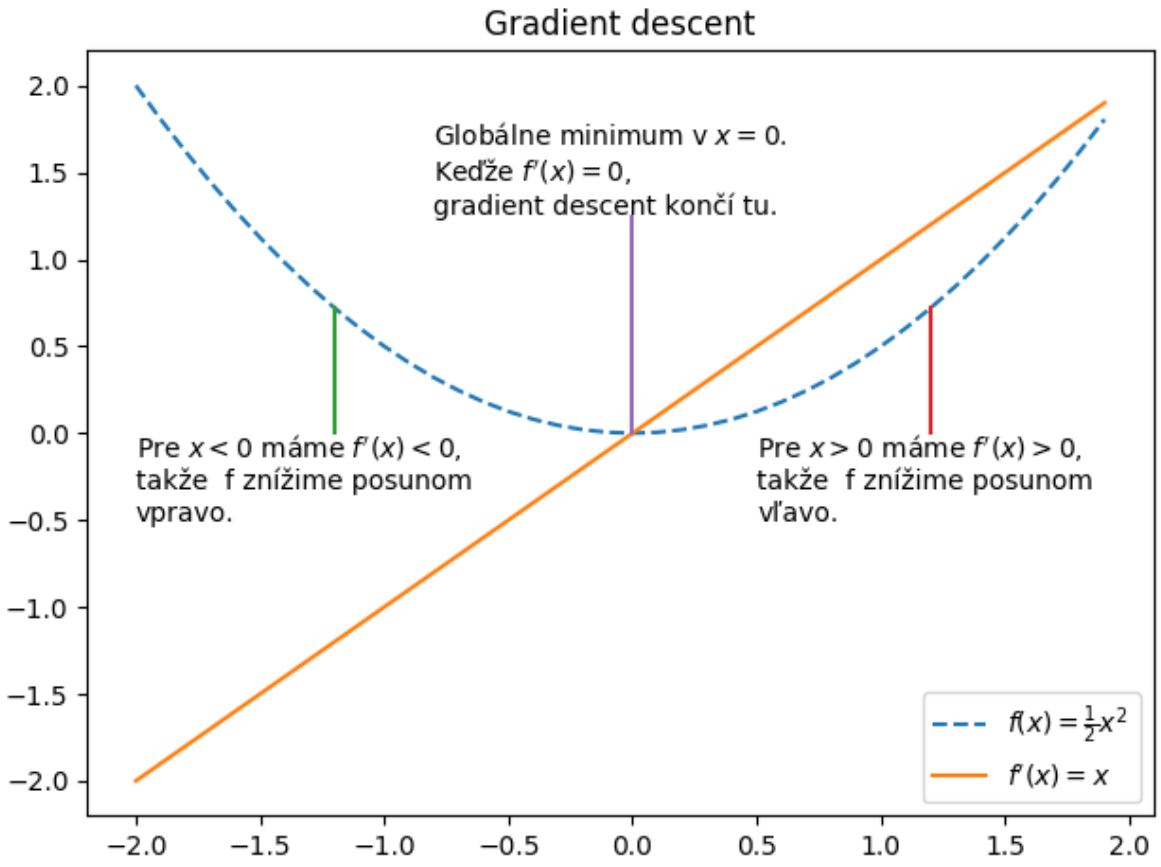
2.4 Gradient descent

Po vypočítaní chybovej funckie je potrebné upraviť samotné hodnoty váh, pomocou metódy gradient descent. Predpokladajme, že máme funkciu $y = f(x)$, kde obe x a y sú reálne čísla a derivácia tejto funckie je označená ako $f'(x)$. Derivácia $f'(x)$ nám dáva sklon alebo smerovanie funcie $f(x)$ v bode x, inými slovami nám udáva, ako upraviť malú zmenu vo vstupe tak, aby sme dosiahli požadovanú funckiu na výtupe (8). [14]

$$f(x + e) \approx f(x) + ef'(x) \tag{8}$$

Derivácie sú vhodným prostriedkom na minimalizáciu funcií, pretože nám udáva, ako

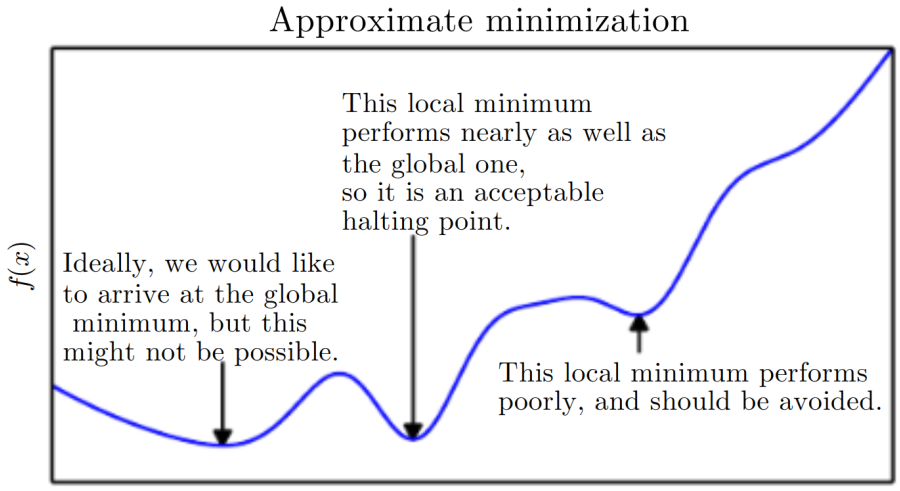
uraviť x tak, aby sme dosiahli zlepšenie v y. Teda pokial vieme, že $f(x-\epsilon(znamienko)(f'(x)))$ je menšie ako $f(x)$ pre malé ϵ. Môžeme teda zmenšiť $f(x)$ úpravou x v malých krokoch v opačnom smere ako je *znamienko* derivácie. Táto technika sa nazýva gradient descent, a jej príklad vidíme na obrázku 10. [14]



Obr. 10: Ukážka, ako derivácie funkcie môže byť použitá na hľadanie minima funkcie (gradient descent)

V bodoch, kde $f'(x) = 0$, nemáme informáciu o tom, ktorým smerom sa má hodnota upraviť, a sú označované ako kritické alebo stacionárne body [14]. Lokálne minimum je bod, kde $f(x)$ je menšia ako všetky okolité body, takže nie je viac možné znížiť hodnotu $f(x)$ vykonaním nekonečne malých krokov v jednom alebo druhom smere. Analogicky lokálne maximum je body, kde $f(x)$ je väčšia ako všetky okolité body, takže nie je viac možné zvýšiť hodnotu $f(x)$ vykonaním nekonečne malých krokov v jednom alebo druhom smere. Niektoré kritické body niesu ani minimom ani maximom a nazývajú sa sedlové body.

Bod, kde $f(x)$ dosahuje absolútne najnižšiu hodnotu sa nazýva globálne minimum. Je možné, aby mala funkcia jedno, alebo aj viac globálnych miním. Tiež je možné, aby existovali lokálne minimá, ktoré niesu globálne optimálne. V kontexte učenia neurónových sietí optimalizujeme funkcie, ktoré môžu mať veľké množstvo lokálnych miním, veľké množstvo sedlových bodov so širokými plochými oblasťami. Toto všetko robí trénovanie neurónovej siete náročným, najmä pokiaľ ide o n-rozmerné dáta. Kvôli tomu sa trénovanie siete často ukončí pri hodnote funkcie $f(x)$ ktorá je veľmi malá, ale nie nutne minimálne z formálneho hľadiska, ako vidíme na obrázku 11. [14]



Obr. 11: Ukážka funkcie s niekoľkými lokálnymi minimami, prebraté z[14, p. 85]

2.5 Stochastic gradent descent

Stochastic gradient descent SGD dôležitým a často používaným algoritmom pri učení neurónových sietí. Ide o rozšírenie gradient descent algoritmu, ktorý sme popísali v 2.4.[14]

Pri učení neurónových sietí sa snažíme dosiahnuť čo najlepšiu generalizáciu problému, k čomu je potrebné veľké množstvo trénovacích dát (dataset), ktoré sú však výpočtovo náročné. Chybová funkcia používaná v strojovom učení je často založená na vypočítaní chyby pre každý prvok z trénovacích dát a ich následnom sčítaní, náročnosť tohoto výpočtu je $O(m)$ pre m prvkov. SGT tento problém rieši používaním menšieho počtu vzoriek (minibatch) pri výpočte chybovej funkcie, ktorých množstvo postačuje na určenie smeru optimalizácie chybovej funkcie. Použitie menšieho počtu vzoriek je kľúčovým aspektom pri trénovaní veľkých modelov neurónových sietí s veľkými trénovacími datasetmi. Ak má model siete fixnú veľkosť, potom náročnosť SGD pri výpočte chyby v jednom cykle nezáleží

od veľkosti trénovacieho datasetu m.

Počet výpočtov chybovej funkcie pri klasickom gradient descent algoritme zvyčajne rastie v závislosti od počtru trénovacích prvkov. S množstvom prvkov m blížiacemu sa k nekonečnu, dosiahne model najlepší možný výsledok ešte predtým, než SGD použije všetky prvky trénovacieho datasetu. Zvyšovaním množstva prvkov v datasete v takomto prípade už nenavýši množstvo času potrebného na dosiahnutie najlepšieho možného výsledku. Z tohoto uhla pohľadu je možné tvrdiť, že približná náročnosť trénovania modelu pomocou SGD sa rovná $O(1)$, ako funkcií m.[14]

2.6 Generalizácia, overfitting a underfitting

Hlavným cieľom strojového učenia aby model fungoval dobre nielen na trénovacích dátach, ale najmä na úplne nových vstupných dátach. Schopnosť neurónovej siete správne fungovať aj na nových dátach nazývame generalizácia.

Pri trénovaní neurónovej siete sa využíva trénovací dataset, na ktorom sa počíta trénovacia chyba, ktorú sa snažíme minimalizovať. Tento proces môžeme zaradiť zaradiť medzi jednoduché optimalizačné problémi. Čo však oddeľuje strojové učenie od optimalizácie je snaha dosiahnuť aj nízku generalizačnú chybu, tiež označovanej ako testovacia chyba a je definovaná ako hodnota chyby na nových dátach.[14].

Testovaciu chybu modelu neurónovej siete získavame meraním úspešnosti na testovacom datasete, pričom vzorky z testovacieho datasetu sa nenachádzajú v trénovacom datasete - teda sú úplne nezávislé, rozdelené rovnomerným spôsobom. Pri výpočte testovacej chyby chyby sa do neurónovej siete privedú vzorky z trénovacieho datasetu, vypočíta sa trénovacia chyba, optimalizujú sa parametre - váhy tak aby sa znížila trénovacia chyba - tento proces sme popísali v kapitole 2.4, potom sa na vstup neurónovej siete privedú vzorky z testovacieho datasetu a vypočíta sa testovacia chyba, ktorá je väčšia alebo rovná trénovacej chybe[14].

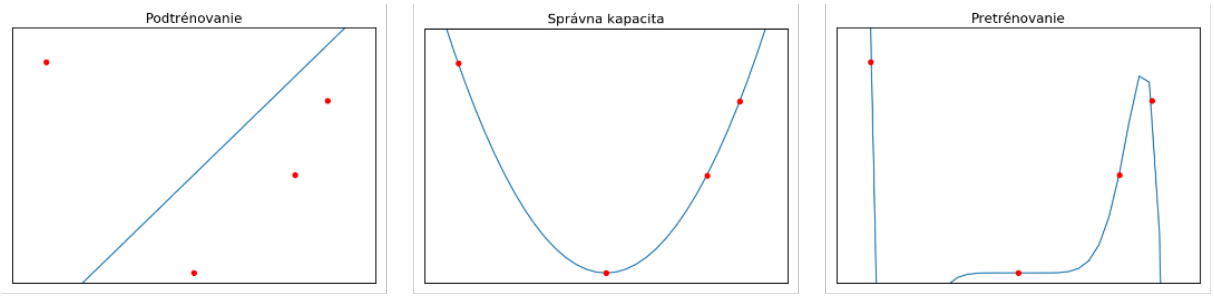
Faktory, ktoré určujúc úspešnosť neurónovej siete, sú schopnosti:

- Minimalovať trénovaciu chyby
- Udržať čo najmenšiu vzdialenosť medzi testovacou a trénovacou chybou

[14]

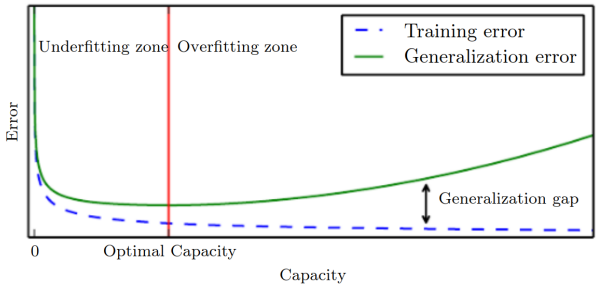
Tieto dva faktory faktory korešpondujú dvom problémom strojového učenia - nedotrénovanie (underfit) a pretrénovanie(overfit). Undefitting znamená, že modelej siete nie je schopný dosiahnuť požadovanú hodnotu chyby na trénovacom datasete. Pretrénovanie označuje situáciu, kedy je rozdiel medzi trénovacou a testovacou chybou príliš veľký. Kapacitou modelu (capacity), je možné kontrolovať, či bude sieť náchylnejšia k nedotrénovaniu alebo pretrénovaniu. Kapacitou modelu myslíme schopnosť modelu obsiahnuť širokú škálu funkcií[14]. Modely s nižšou kapacitou môžu mať problém pri naučení sa zložitejších funkcií trénovacích dát, zatiaľ čo modely s vyššou kapacitou sú náchylné k naučeniu sa takých vlastností trénovacieho datasetu, ktoré zvyšujú chybovosť testovacieho datasetu. Vo všeobecnosti teda platí, že model funguje najlepšie v prípade, ak kapacita modelu korešponduje so skutočnou komplexitou problému, ktorý sa sieť model snaží naučiť [15].

Na obrázku 12 vidíme názornú ukážku týchto princípov, na ktorom porovnávame model s kapacitou na predikovanie lineárnych, kvadratických a 9 dimenzionálnych funkcií, pričom skutočným cieľom je predikovanie kvadratických funkcií. Linerána kapacita nie je schopná poňať celú podstatu kvadratickej funckie, a preto je model podtrénovaný. Model o kapacite 9 je schopný poňať kvadratickú funkciu, avšak jeho rozsah dovoľuje natrénovať nekonečné množstvo iných funkcií so správnymi hodnotami pre trénovacie dáta, pretože má model viac parametrov než máme trénovacích dát.



Obr. 12: Kapacita siete a jej vzťah k nedotrévaniu a pretrénovaniu

Ukážku trénovacaj a testovacej chyby a ich vzťahu s podtrénovaním a pretrénovaním vidíme na obrázku 13.



Obr. 13: Vzťah medzi kapacitou a chybou trénovania, prebratá z z[14, p. 115]

2.7 Regularizácia

Ako sme už spomenuli, hlavným cieľom pri strojovom učení je vytvoriť taký model, ktorý bude fungovať správne nielen na trénovacích dátach, ale taktiež na nových neznámych vzorkách. Veľké množstvo algoritmov strojového učenia je špeciálne navrhnutých tak, aby minimalizovali testovaciu chybu, hoci za cenu výššej trénovacej chyby. Tieto prístupy pri učení neurónových sietí označujeme ako regularizácia[16]. Najčastejšie ide o pridanie nových pravidiel do modelu neurónovej siete, ako napríklad obmedznie možných hodnôt parametrov neurónovej siete či pridanie pravidiel do chybovej funkcie, čo môžeme tiež chápať ako nepriame obmedzenie hodnôt parametrov v neurónovej sieti [14]. Za predpokladu, že sú tieto pravidlá a obmedzenia vybrané vhodne, môžu viesť k lepším výsledkom na testovacom datasete.

2.7.1 L^2 regularizácia

L^2, tiež známa ako úpadok váh(weight decay), je regularizáciou, ktorá približuje váhy smerom k pôvodnej hodnote - najčastejšie k 0 tým, že do chybovej funkcie zavádza vzťah (9) [17].

Tento vzťah vyjadruje, že L^2 spôsobuje, že stupné dáta si zachovávajú vyššiu varianciu, čím znižuje hodnoty váh príznakov, ktorých kovariancia na cieľovom výstupe je nízka v porovnaní s varianciou vstupu. [18].

$$L^2 = \frac{1}{2} \parallel w \parallel_2^2 = \frac{1}{2} \sum_{i=0}^{n} w_i^2 \tag{9}$$

Tento vzťah môžeme tiež chápať tak, že váhy ktorých hodnota je blízka nule majú nízky vplyv na komplexitu modelu, zatiaľ čo vzdialenejšie hodnoty majú obrovský vplyv [17].

2.7.2 L^1 regularizácia

L^2 regularizácia je pravdepodobne najpoužívanejšia pri redukovaní hodnôt váh, okrem nej však existujú aj iné druhy regularizácií ktoré zmenšujú veľkosť modelu - napríklad L^1, ktorá je definovaná vzťahom (10) [14].

$$L^1 = \| w \| + = \sum_{i=0}^{n} |w_i|_i \tag{10}$$

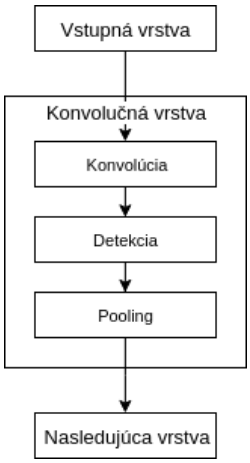
V porovnaní s L^2 je výsledkom L^1 regularizácie redší model, čo znamená, že väčšie množstvo parametrov môže dosiahnuť presnú hodnotu 0, na rozdiel od L^2, kde sa hodnota parametrov len približuje k 0.[19]. Toto môžeme chápať ako mechanizmus selekcie použitých príznakov, čím sa zjednodušuje proces učenia neurónovej siete, keďže sa zníži množstvo parametrov, ktoré sa použijú.

2.8 Konvolučné neurónové siete

Konvolučné neurónové siete sú špeciálnym druhom neurónových sietí ktoré majú topológiu podobnú mriežke, ako napríklad time-series dát, reprezentovateľné ako 1-rozmerná mriežka, alebo obrázkové dáta, v ktorých sú pixely usporiadané ako 2-rozmerná mriežka[14].

Vrstvy konvolučnej neurónovej siete sa v bežnom prípade skladajú z troch základných krokov, ako vidíme aj na obrázku14, v literatúra sa často uvádzajú tiež ako osobitné vrstvy:

- Konvolúcia
- Detekcia (ReLU)
- Pooling



Obr. 14: Diagram vrstvy konvolučnej neurónovej siete

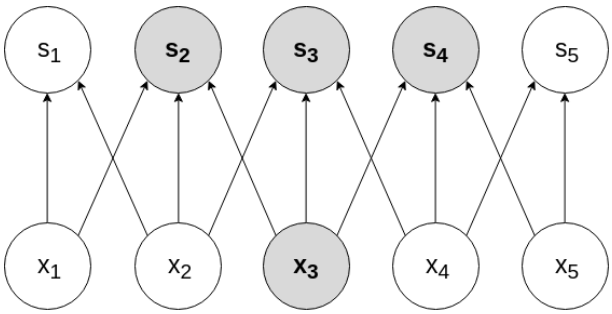
2.8.1 Konvolúcia

Názov konvolučná neurónová sieť je založený na matematickej operácii - konvolúcii. Tá je založená na troch základných bodoch, ktoré zlepšuju učenie neurónovej siete:

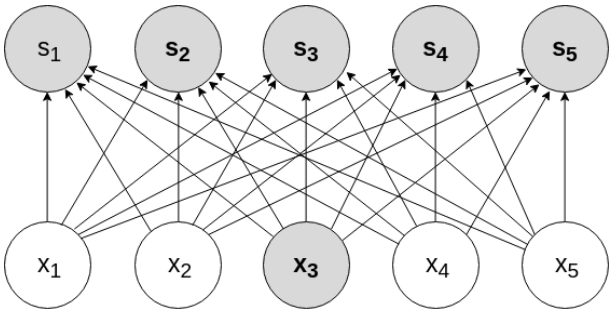
- Redšie prepojenia (sparse interactions)
- Zdieľanie parametrov (parameter sharing)
- Ekvivariantné vyjadrenia (equivariant representations)

[14].

Tradičné vrstvy neurónové siete sú postavené na násobení matice parametrov s maticou parametrou (váhami), medzi každým vstupným a každým výstupným neurónom - čo znamená, že každý neurón stupnej vrstvy ovplyňuje všetky neuróny výstupnej vrstvy. Konvolučné neuróné siete sa v tomto odlišujú, pretože používajú redšie prepojenia, čo je dosiahnuté použitím menšieho jadra než je vstup (kernel). Napríklad, pri spracovaní obrázkových dát, ktoré pozostávajú z miliónov pixelov, detekujeme menšie príznaky ako napríklad hrany, v podoblasti len niekoľkých stoviek pixelov. [14] Teda, pokiaľ máme m vstupných neurónov a n výstupných, potom maticové násobenie vyžaduje $m \times n$ parametrov, a teda náročnosť takejto operácie je $O(m \times n)$. Ak však zmenšíme množstvo prepojení na k, potom je potrebných $k \times n$, s náročnosťou $O(k \times n)$, pričom v praxi je možné dosiahnuť dobré výsledky trénovania pri použití k, ktoré je omnoho menšie ako m. To znamená, že je potrebný menší počet parametrov, čo šetrí pamäťové prostriedky, znižuje množstvo operácií, ktoré je potrebné vykonať, a vo všeobecnosti výrazným spôsobom zlepšuje efektivitu trénovania modelu. Názornú demonštráciu redších prepojení vidíme na obrázku 15 a ukážku plný prepojení na obrázku 16.



Obr. 15: Ukážka redšieho prepojenia neurónov so zvýraznením prepojení, ktoré má neurón x_3



Obr. 16: Ukážka plných prepojení neurónov so zvýraznením prepojení, ktoré má neurón x_3

Zdieľanie parametrov znanemná, že rovnaké parametre sú použíté vo viacerých funkciách modelu. Pri klasických neurónových sieťach je každý element matice váh použitý práve jeden krát, s jedným elementom zo vstupu, pri výpočte výstupu pre nasledujúcu vrstvu. Pri konvolučných neurónových sieťach sa každý prvok jadra použije spolu s každým elementom zo vstupu (okrem hraničných v závislosti od návrhu modelu - padding, stride ...). Zdieľanie parametrov teda znamená, že namiesto učenia osobitných skupín parametrov pre každú pozíciu zo vstupu, sa sieť naučí len jednu, ktorá je použítá pre všetky lokácie.[14]

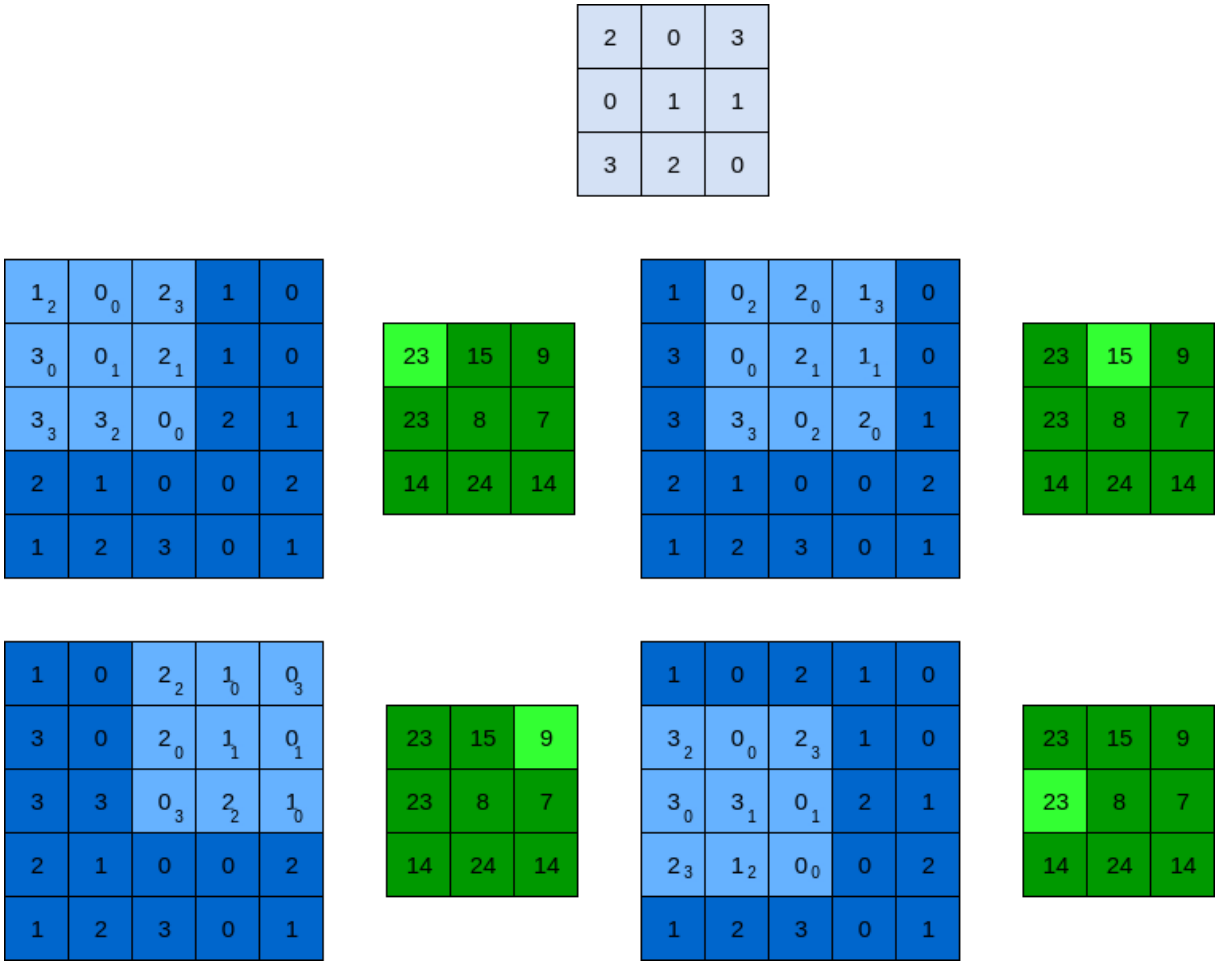
Vďaka zdieľaniu parametrov nadobúdajú vrstvy neurónovej siete vlastnosť ekvivariancie voči presunom. Funkcia je ekvivariantná, pokiaľ zmeny vstupu ovplyvňujú zmeny vo výstupe rovnakou mierou, a teda, funkcia $f(x)$ je ekvivariantná k funkcii g, ak $g(f(x)) = f(g(x))$. V prípade konvolúcie, nech g je funkcia ktorá, sa presúva po vstupe, je funkcia g ekvivariantná funkcií konvolúcie. Táto vlastnosť je obzvlásť výhodná, pokiaľ chceme na celom vstupe detekovať rovnaké príznaky - napríklad hrany, noapak nemusí pokiaľ chceme v rôznych častiach obrázka detekovať iné príznaky - napríklad vo vrhnej časti obrázka detekovať obočie, a v spodnej bradu. [14]

Na obrázku17 vidíme jednoduchý príklad operácie konvolúcie, kde tmavomodrá mriežka predstavuje vstupnú mapu príznakov (input feature map). Pre zjednodušenie použijeme príklad len s jednou mapou príznakov, avšak v bežnej praxi ich môže byť niekoľko naskladaných jedna na druhej. Šedá oblasť ktorá sa označuje ako jadro, posúvame naprieč mapou príznakov. Pre každú podoblasť mapy príznakov je vypočítaný výstup vychádzajúci z danej podloblasti, ktorých sčítanie tvorí výstup danej podoblasti.[20] Tento postup môže byť zopakovaný $n-krát$ použitím rôznych jadier, čím vznikne n výstupných máp priznakov, výstupná mapa príznakov je označená zelenou farbou. Pokiaľ máme niekoľko vstupných

máp vstupujúcich do konvolúcie, potom je potrebné použiť 3-rozmerné jadro, alebo, každá zo vstupných máp použije odlišné jadro, ktorých výsledok bude následne sčítaný do výstupnej mapy príznakov.

Príklad na obrázku 17 zobrazuje 2-rozmernú konvolúciu, avšak môže byť použitý na generalizovanie aj n-rozmernej konvolúcie. V prípade 3-rozmernej konvolúcie by jadro predstavovalo kváder, ktorým by sme prechádzali naprieč výškou, šírkou a hĺbkou vstupnej mapy príznakov.[20]

Súbor jadier predstavujúcich konvolúcie má tvar korešpondujúci permutácii $(n, m, k_1, \ldots, k_N)$ kde n označuje počet výstupných máp príznakov, m označuje počet vstupných máp príznakov a k_j označuje veľkosť jadra pozdĺž osi j.[20]



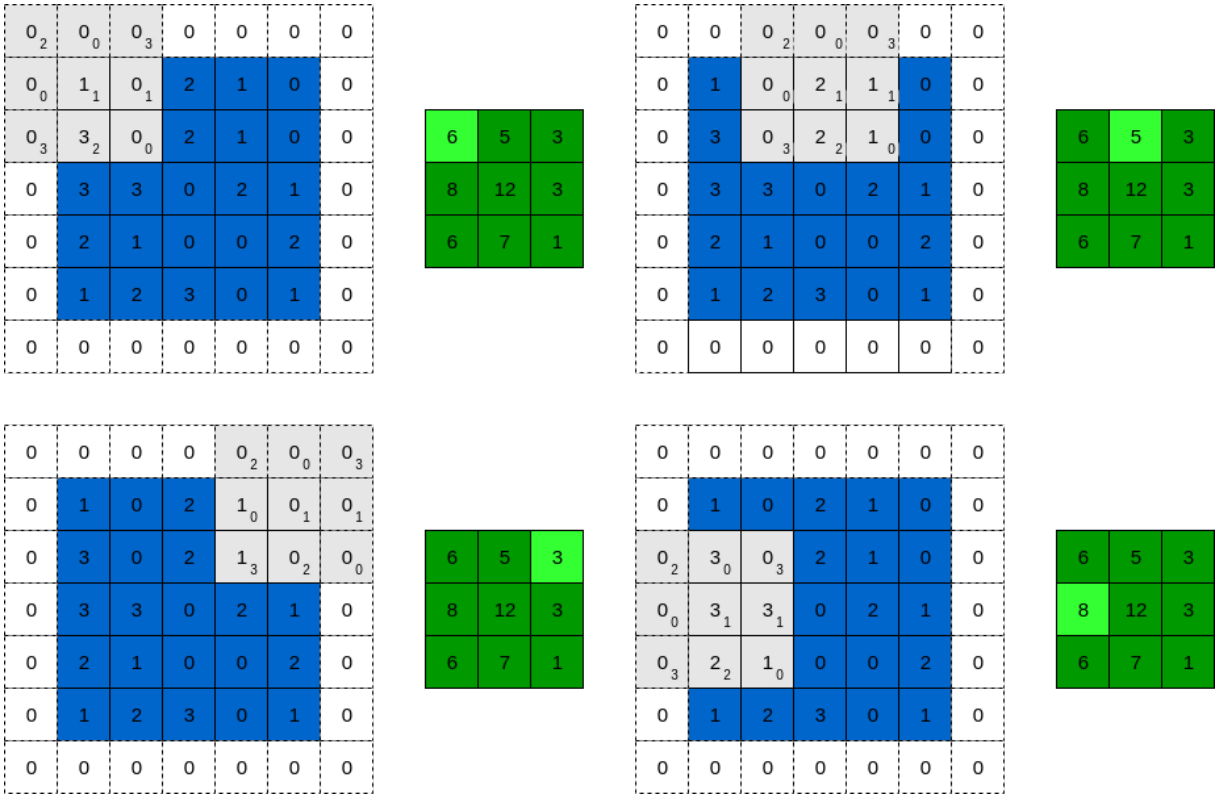
Obr. 17: Výpočet výstupných hodnôt operácie konvolúcie

Výstupnú veľkosť o_j konvolučnej vrstvy pozdĺž osi j ovplyvňuje niekoľko parametrov:

- i_j: vstupná veľkosť (input size) pozdĺž osi j

- k_j: veľkosť jadra (kernel size) pozdĺž osi j
- s_j: vzdialenosť medzi dvomi po sebe nasledujúcimi podoblasťami (stride) pozdĺž osi j
- p_j: veľkosť okraja (padding) pozdĺž osi j

Na obrázku 18 ukazuje použitie jadra o veľkosti 3×3, použitého na vstupe o veľkosti 5×5, okrajom 1×1 s hodnotami 0 a stridom 2×2.



Obr. 18: Výpočet výstupných hodnôt konvolúcie s použitím paddingu a stridu

2.8.2 Pooling

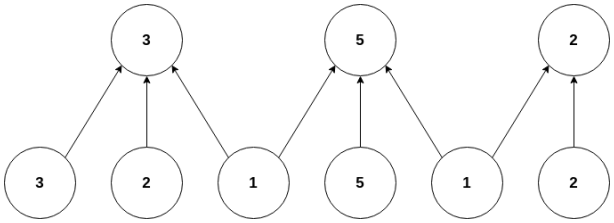
Vo fáze detekcie prejde výstup konvolúcie aktivačnou funkciou ReLU, ktorú sme spomenuli v časti2.2. Výstup z fázy detekcie pokračuje do funkcie poolingu, ktorá ďalej upraví výstup do ďalšej vrstvy.

Ide o aplikáciu poolingovej funkcie na vstupné dáta, čím dôjde k zmenšeniu výsledku. Podobne ako pri konvolúcií, prechádzame vstup krok po kroku, avšak namiesto lineárnej operácie používanej pri konvolúcii, použijeme štatistickú funkciu, hľadajúci napríklad maximálne, priemerné či minimálne hodnoty v danej oblasti [21]. Vo všetkých týchto

prípadoch, napomáha pooling vytvoriť výstup, ktorý je invariantný voči malým zmenám pozičným posunom vstupu - to znamená, že pokiaľ sa sa vstup do funkcie poolingu zmení len o malú hodnotu, potom sa väčšina dát vo výstupe poolingu nezmení. Invariancia voči malým lokálnym zmenám môže byť veľmi užitočná, pokiaľ je väčším cieľom detekovať, či sú príznaky vôbec prítomné, než detekovať kde presne sú[14]. Napríklad, ak je naším cieľom detekovať, či obrázok obsahuje tvár, potom nepotrebujeme zisťovať presnú polohu očí na pixel presne, stačí, že vieme zistiť, že tvár má oči na oboch stranách.

Poolingová funkcia použitá na riedke oblasti vytvára invarianciu voči posunom, avšak ak použijeme pooling na nezávislé parametrizované konvolúcie, potom je možné, že sa príznaky samé naučie, voči ktorým príznakom sa majú stať invariantnými[14].

Pretože pooling sumarizuje výsledok z celej oblasti, potrebujeme menšie množstvo jednotiek vykonávajúcich pooling než jednotiek detektora, vďaka tomu, že bedú do úvahy oblasti o veľkosti k pixelov, namiesto 1 pixelu, ako vidíme aj na obrázku 19. Tým sa zvyšuje výpočtová efektivita vrstvy neurónovej siete, pretoťe ďalšia vrstva má k krát menší vstup, ktorý potrebuje spracovať.[14]

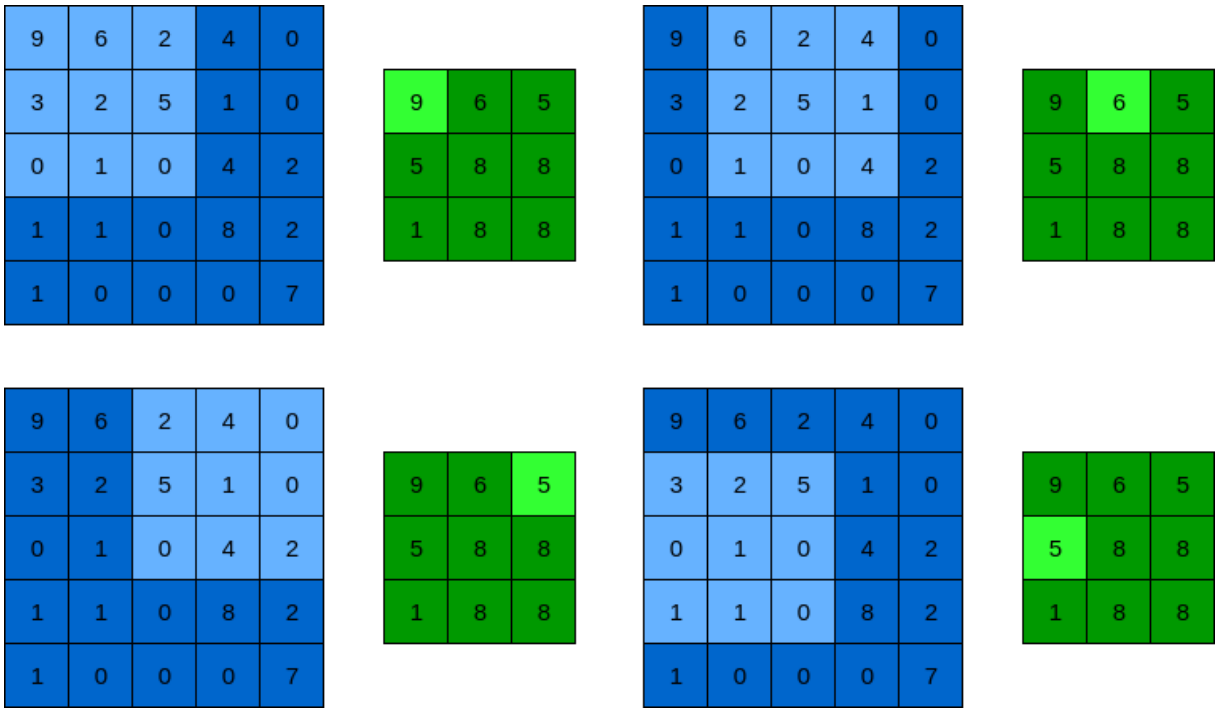


Obr. 19: Max pooling spolu s downsamplingom z pohľadu prepojení neurónov

Veľkosť výstupu o_j a fungovanie poolingu ovplyvňuje niekoľko parametrov.

- i_j: vstupná veľkosť (input size) pozdĺž osi j
- k_j: veľkosť polingovej podoblasťami (pooling window) pozdĺž osi j
- s_j: vzdialenosť medzi dvomi po sebe nasledujúcimi podoblasťami (stride) pozdĺž osi j

Príklad ich použitia vidíme na obrázku 20, kde $i = 5 \times 5$, $k = 3 \times 3$ a $s = 1 \times 1$:



Obr. 20: Výpočet výstupných hodnôt použitím max poolingu

3 Model neurónovej siete na rozpoznávanie tvárí

Cieľom tejto práce je aplikovať natrénovaný model konvolučnej neurónovej siete pri rozpoznávaní tváre na zariadení Android OS. Tento cieľ je teda možné rozdeliť na dve základné časti:

1. Natrénovanie modelu neurónovej siete
2. Aplikácia modelu na zariadení Android OS.

Táto kapitola sa bude venovať prvému budu - vytvoreniu natrénovaného modelu neurónovej siete. V našom prípade sme netrénovali celú sieť na rozpoznávanie tváre, ale použili sme model neurónovej siete Facenet, ktorá je schopná extrahovať príznaky tváre, ktoré budú použité pri klasifikácii.

3.1 Facenet

Pri popise tohoto modelu neurónovej siete sa budeme opierať najmä o článok [22], v ktorom je detailne popísaná štruktúra Facenetu a problémov ktoré rieši. Taktiež využijeme poznatky Sandberga, ktorý vytvoril voľne dostupnú implementáciu Facenetu na `https://github.com/davidsandberg/facenet`, a ktorý sa ďalej vyvíja. Chceme upozorniť, že v tejto práci budeme popisovať implementáciu modelu k marcu 2018, vhľadom k tomu, že v súčastnosti sa jeho implementácia pozmenila.

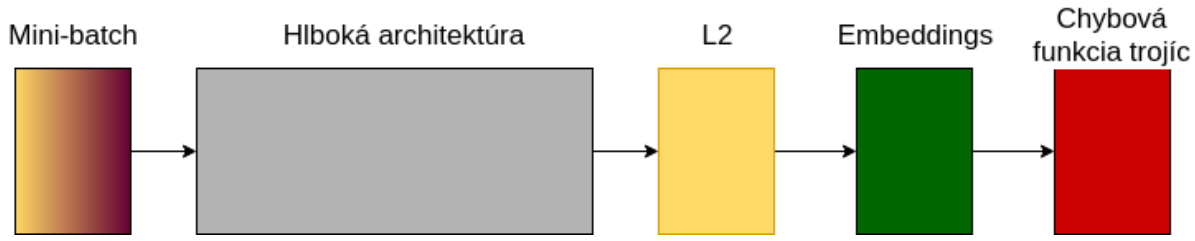
Ide o model neurónovej siete, navrhnutý na rozpoznávanie tvárí pre rôzne potreby - identifikácie, verifikácie či clusteringu (hľadanie a spájanie rovnakých tvári). Je založený na učení sa príznakov v Euklidovskom priestore (Euclidean embedding) obrázkov prostredníctvom hlbokej konvolučnej siete. Sieť je trénovaná tak, aby umocnené L^2 vzdialenosti v podpriestore príznakov priamo poukazovali na podobnosť tvárí - teda podobné tváre majú nižšiu vzdialenosť než odlišné tváre [22]. Po vypočítaní príznakov tváre je ďalší krok rovnaký ako pri iných klasifikačných problémoch - vypočítanie vzdialenosti medzi príznakmi dvoch tvári a určeniu či ide o rovnaké tváre na základe thresholdu v prípade verifikácie alebo k-NN (k nearest neighbors) klasifikačný problém v prípade identifikácie.

Facenet sa odlišuje od ostatných prístupov k rozpoznávaniu tvárí pomocou hlbokých neurónových sietí, menovite článok [22] porovnáva prístup voči [23], kde bola použitá stredná vrstva neurónovej siete natrénovanej na známych inentitách, na všeobecné rozpoznívanie

tvárí mimo trénovacieho datasetu. “Nevýhodou tohoto prístupu je však nepriamosť a neefektivita: je nutné dúfať, že reprezentácie tváre strednou vrstvou je dostatočne generalizovaná voči novým tváram; použitím strednej vrstvy na vznikajú veľké (1000 - rozmerné) reprezentácie tvárí“[22]. Túto dimenzionalitu je možné síce redukovať pomocou PCA podobne ako v [23], avšak ide len o lineárnu transformáciu príznakov, čím nemusíme nutne zaručiť dostatočnú generalizáciu na nové tváre.

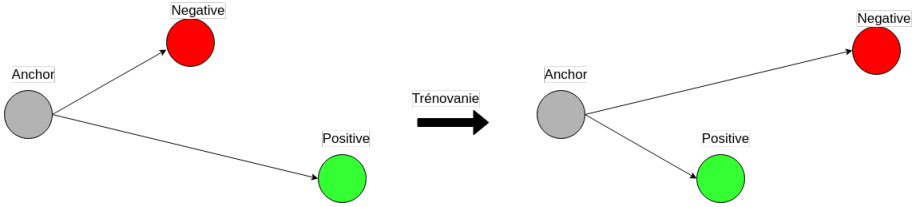
Prístup Facenetu je odlišný v tom, že priamo trénuje neurónovú sieť do 128-rozmerného výstupu, pričom trénovanie je založené chybovej funkcií využívajúcej trojice založenej na LMNN (Large margin nearest neighbor)[24]. Chybová funkcia Facenetu teda pracuje s trojicami, ktoré pozostavajú z dvoch tvárí rovnakej identity a jednej odlišnej, čím dosahuje väčšiu vzdialenosť v priestore medzi tvárami odlišných identít a naopak nižšiu medzi tvárami rovnakej identity. Na trénovanie využíva orezané obrázky tvárí, na ktoré nebolo aplikované žiadne špeciálne 2-rozmeré alebo 3-rozmerné zarovnanie, okrem škálovania a posunu.

Je postavený na architektúre neurónových sietí typu Inception, ktoré si popíšeme neskôr. Zjednodušený diagram fungovania Facenetu vidíme na obrázku 21



Obr. 21: Diagram fungovania Facenetu

Prostredníctvom chybovej funkcie trojíc sa sieť usiluje o zapuzdrenie $f(x)$, vstupného obrázka x do d-rozmerného Euklidovského priestoru príznakov R^d, $f(x) \in R^d$, aby umocnená vzialenosť medzi všetkými tvárami, nezávisle na variácií póz, osvetlenia či iných podmienok, medzi rovnakými tvárami bola nízka, zatiaľ čo vzialenosť medzi obrázkami tvárí rozdielnych bola veľká, 22[22]. Okrem toho zavádza ďalšie pravidlo $\| f(x) \|_2 = 1$ čím omedzuje vnorenie na d-rozmernú hypeguľu.



Obr. 22: Ukážka princípu úpravy vzdialeností medzi vzorkami pri chybovej funkcii trojíc

Túto chybovú funkciu vieme vyjadriť ako rovnicu (11) (12), kde x_i^a označuje vstupnú vzorku (anchor), x_i^p označuje zhodnú identitu (positive), ktorú chceme priblížiť vstupu, x_i^n inú osobu (negative), α vyjadruje vynútený rozostup medzi pozitívnym a negatívnym párom a τ označuje všetky možné trojice obrázkov s kardinalitou N [22].

$$\| f(x_i^a) - f(x_i^p) \|_2^2 + \alpha < \| f(x_i^a) - f(x_i^n) \|_2^2 \tag{11}$$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \tau \tag{12}$$

Výsledná chybová funkcia má potom tvar (13) [22].

$$\sum_i^N [\| f(x_i^a) - f(x_i^p) \|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2 + \alpha]_+ \tag{13}$$

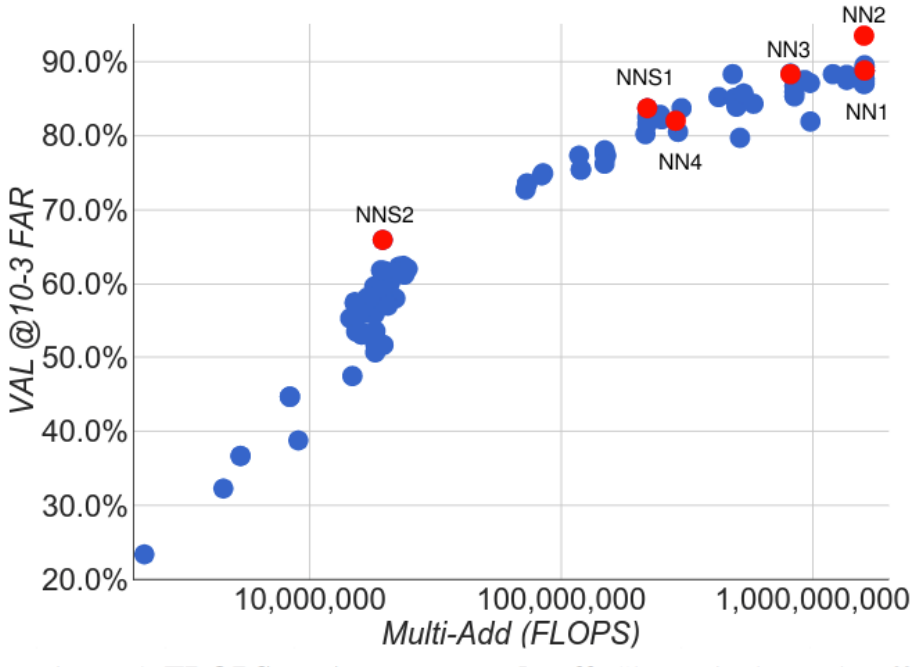
Vhodný výber trojíc sa ukázal ako kľúčový pre rýchlu konvergenciu k dobrým výsledkom, a teda vybrať také trojice, aby porušovali rovnicu (11), čo znamená, že pre zvolený vstup x_i^a chceme vybrať také x_i^p, aby sme dosiahli $argmax_{x_i^p} \| f(x_i^a) - f(x_i^p) \|_2^2$ a podobne x_i^n také aby spĺňalo $argmin_{x_i^n} \| f(x_i^a) - f(x_i^n) \|_2^2$.

Autori článku[22] o Facenete udávajú, že je nemožné počítať *argmin* a *argmax* cez celý tréningový dataset kvôli výpočtovej náročnosti, a navrhujú dve možné riešenia výberu trojíc:

- Generovať trojice offline každých n krokov a počítať *argmin* a *argmax* na podmnožine dát
- Generovať trojice online, výberom pozitívnej a negatívnej vzorky z mini-batchu (dávky v jednom cykle)

Z týchto dvoch možností autofi zvolili možnosť online generovania s použitím väčších - niekoľko-tisícových mini-batchov, v ktorých sa nachádza približne 40 vzoriek pre jednu identitu.

Model bol trénovaný pomocou SGD algoritmu, ktorý sme popísali v 2.5. Autori testovali 4 rôzne architektúry typu Inception, líšiacich sa vo veľkosti vstupu (220x220 - 96x96) či hĺbke siete a množstve potrebnej výpočtovej sily(220 mil. FLOPS - 20 mil. FLOPS), a ich porovnanie výsledkov vidíme na obrázku 23, a dosiahli úspešnosť na LFW datasete $99,63\% + -0.09$ s použitím zarovnani tvárí. Z výsledkoch ich pokusov je tiež zaujimavé, že dimenzionalita príznakov na ktoré trénovali neurónové siete je prekvapivo najvhodnejšia pri 128 rozmeroch, kde dosiahli najlepšie výsledky v porovnaní so 64, 256 a 512 dimezií. Tieto výsledky dosiahli na veľkosti snímkov o rozmeroch 220×220, avšak poukázali na to, že menšie snímky o veľkosti 120×120 dosahujú len o málo horšie výsledky.



Obr. 23: Porovnanie výpočtovej náročnosti v závislosti od úspešnosti siete, prebraté z[22, p. 5]

V ďalších čatiach si popíšeme konkrétnu implementáciu od Facenetu Sandberga [25].

3.2 Tensorflow

Záver

Conclusion is going to be where?
Here.

Zoznam použitej literatúry

1. HEADQUARTERS, ACI WORLD. *The Application of Biometrics at Airports*. 2005. (Accessed on 04/04/2018).
2. JAIN, Anil K a LI, Stan Z. *Handbook of face recognition*. Springer, 2011.
3. KANADE, Takeo. Picture processing system by computer complex and recognition of human faces. 1974.
4. KIRBY, Michael a SIROVICH, Lawrence. Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern analysis and Machine intelligence*. 1990, roč. 12, č. 1, s. 103–108.
5. TURK, Matthew a PENTLAND, Alex. Eigenfaces for recognition. *Journal of cognitive neuroscience*. 1991, roč. 3, č. 1, s. 71–86.
6. BELHUMEUR, Peter N., HESPANHA, João P a KRIEGMAN, David J. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*. 1997, roč. 19, č. 7, s. 711–720.
7. WISKOTT, Laurenz, KRÜGER, Norbert, KUIGER, N a VON DER MALSBURG, Christoph. Face recognition by elastic bunch graph matching. *IEEE Transactions on pattern analysis and machine intelligence*. 1997, roč. 19, č. 7, s. 775–779.
8. VIOLA, Paul a JONES, Michael. Rapid object detection using a boosted cascade of simple features. 2001, roč. 1, s. I–I.
9. ZHAO, Wenyi, CHELLAPPA, Rama, PHILLIPS, P Jonathon a ROSENFELD, Azriel. Face recognition: A literature survey. *ACM computing surveys (CSUR)*. 2003, roč. 35, č. 4, s. 399–458.
10. JAIN, Anil K, FLYNN, Patrick a ROSS, Arun A. *Handbook of biometrics*. Springer Science & Business Media, 2007.
11. DUDA, Richard O, HART, Peter E a STORK, David G. *Pattern classification*. John Wiley & Sons, 2012.
12. BARTLETT, Marian Stewart, MOVELLAN, Javier R a SEJNOWSKI, Terrence J. Face recognition by independent component analysis. *IEEE Transactions on neural networks*. 2002, roč. 13, č. 6, s. 1450–1464.
13. BUDUMA, Nikhil a LOCASCIO, Nicholas. *Fundamentals of deep learning: designing next-generation machine intelligence algorithms*. Ö'Reilly Media, Inc.", 2017.

14. GOODFELLOW, Ian, BENGIO, Yoshua, COURVILLE, Aaron a BENGIO, Yoshua. *Deep learning*. MIT press Cambridge, 2016.

15. LAWRENCE, Steve, GILES, C Lee a TSOI, Ah Chung. Lessons in neural network training: Overfitting may be harder than expected. In: *AAAI/IAAI*. 1997, s. 540–545.

16. NIELSEN, M. Chapter 3, Improving the Way Neural Networks Learn. *Neural Networks and Deep Learning. Available online: http://neuralnetworksanddeeplearning. com/chap3. html (accessed on 8 May 2017)*. 2015.

17. *Regularization for Simplicity: L2 Regularization | Machine Learning Crash Course | Google Developers* [`https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization`]. (Accessed on 05/04/2018).

18. LAARHOVEN, Twan van. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*. 2017.

19. *Regularization for Sparsity: L1 Regularization | Machine Learning Crash Course | Google Developers* [`https://developers.google.com/machine-learning/crash-course/regularization-for-sparsity/l1-regularization`]. (Accessed on 05/06/2018).

20. DUMOULIN, Vincent a VISIN, Francesco. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*. 2016.

21. CS231N. *CS231n Convolutional Neural Networks for Visual Recognition* [`http://cs231n.github.io/convolutional-networks`]. 2018. (Accessed on 05/01/2018).

22. SCHROFF, Florian, KALENICHENKO, Dmitry a PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, s. 815–823.

23. WST, Web-Scale Training. Deeply learned face representations are sparse, selective, and robust. *perception*. 2008, roč. 31, s. 411–438.

24. WEINBERGER, Kilian Q a SAUL, Lawrence K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*. 2009, roč. 10, č. Feb, s. 207–244.

25. *davidsandberg/facenet: Face recognition using Tensorflow* [`https://github.com/davidsandberg/facenet`]. (Accessed on 05/06/2018).

Prílohy