Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Klasifikácia obrazu ľudských tvári pomocou hlbokých neurónových sietí

Diplomová práca

2016 Bc. Martin Slavkovský

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Klasifikácia obrazu ľudských tvári pomocou hlbokých neurónových sietí

Diplomová práca

|  |  |
| --- | --- |
| Študijný program: | Špecializáciaaplikovaná informatika |
| Študijný odbor: | 2511 aplikovaná informatika |
| Školiace pracovisko: | Katedra aplikovanej informatiky FMFI |
| Vedúci práce: | RNDr. Elena Šikudová, PhD. |
| Konzultant: | Ing. Marián Beszédeš, PhD. |

2016 Bc. Martin Slavkovský

Čestné prehlásenie

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov a za odborného vedenia môjho vedúceho diplomovej práce.

.................................................

Bc. Martin Slavkovský

Poďakovanie

Chcem sa poďakovať svojej školiteľke RNDr. Elena Šikudovej, PhD a konzultantovi Ing. Marián Beszédešovi, PhD. za cennú pomoc, rady, konzultácie a čas, ktorý mi venovali počas písania tejto diplomovej práce.

Abstrakt

Úlohou diplomovej práce je štúdium problematiky neurónových sietí so zameraním sa na hlboké konvolučné siete a ich využitie na klasifikáciu ľudských tvárí. Teoretická časť práce sa do hĺbky zaoberá architektúrou konvolučných sietí, tým ako fungujú a ako ich vieme použiť na úlohu rozpoznávania veku a pohlavia.

Praktickou časťou tejto práce je zostavenie vhodnej databázy pre experimenty, zoznámenie sa s opensource Caffe frameworkom umožňujúcim pokročilé techniky učenia konvolučných neurónových sietí. Cieľom práce je implementácia rozpoznávania veku a pohlavia pomocou Caffe, integrácie natrénovaného modelu neurónovej siete do demonštračnej aplikácie,  vysvetlenie štatistických metód použitých na testovanie, podrobné vyhodnotenie dosiahnutých výsledkov a diskusia o možnosti ich rozvoja.

**Kľúčové slová :** Caffe, neurónová sieť, konvolučná neurónová sieť, trénovacia množina, testovacia množina, testovacia chyba, feature mapa

Abstract

The goal of the master’s thesis is the study of neural networks with a focus on convolution deep neural networks and their use for a classification of human faces. In the theoretical part of the thesis we analyze and explain the architecture of convolutional neural networks, how they work and how we can use them for the purpose of age and gender recognition.

Practical part of the thesis consists of creating suitable database for experiments along with getting to know the opensource Caffe framework which allows to work with advanced learning techniques of convolutional neural networks. The goals are: implementing age and gender recognition by using Caffe, integration of the trained model to the demo application, explaining the statistical methods used for testing, evaluation of the test results and a discussion about the future improvements.

**Key words:** Caffe, neural network, convolutional neural network, train set, test set, error, accuracy

Obsah

[Úvod 4](#_Toc450087535)

[1 Umelé neurónové siete 5](#_Toc450087536)

[1.1 Viacvrstvová plne prepojená neurónová sieť 5](#_Toc450087537)

[1.1.1 Učenie pomocou algoritmu spätnej propagácie 6](#_Toc450087538)

[1.1.2 Algoritmus spätnej propagácie 10](#_Toc450087539)

[1.2 Konvolučné siete ( CNN ) 10](#_Toc450087540)

[1.2.1 Konvolúcia 11](#_Toc450087541)

[1.2.2 Architektúra konvolučnej siete 12](#_Toc450087542)

[1.2.3 Vstupná vrstva 12](#_Toc450087543)

[1.2.4 Konvolučná vrstva 13](#_Toc450087544)

[1.2.5 Pooling vrstva ( sub-sampling ) 18](#_Toc450087545)

[1.2.6 Rectified-Linear Unit ( ReLU ) 19](#_Toc450087546)

[1.2.7 LRN vrstva ( Local Response Normalization ) 20](#_Toc450087547)

[1.2.8 Dropout vrstva 20](#_Toc450087548)

[1.2.9 Plne prepojená vrstva 21](#_Toc450087549)

[1.2.10 Vrstva chybovej funkcie 22](#_Toc450087550)

[1.3 Caffe 24](#_Toc450087551)

[1.3.1 Použitie Caffe 24](#_Toc450087552)

[1.4 Podobné práce 26](#_Toc450087553)

[1.4.1 Klasifikácie veku a pohlavia použitím konvolučných sietí 26](#_Toc450087554)

[1.4.2 Estimácia pohlavia a veku nefiltrovaných tvárí 26](#_Toc450087555)

[1.4.3 Odhadovanie pohlavia použitím Adaboostu 27](#_Toc450087556)

[2 Implementácia 28](#_Toc450087557)

[2.1 Príprava dát 28](#_Toc450087558)

[2.1.1 Použité databázy, filtrácia obrázkov 28](#_Toc450087559)

[2.1.2 Normalizácia obrázkov 31](#_Toc450087560)

[2.1.3 Zväčšenie trénovacej množiny 33](#_Toc450087561)

[2.2 Trénovanie 34](#_Toc450087562)

[2.2.1 Príprava trénovania, architektúry sietí 34](#_Toc450087563)

[2.2.2 K-fold cross validácia 35](#_Toc450087564)

[2.3 Testovanie 36](#_Toc450087565)

[2.3.1 Klasifikátor pohlavia 37](#_Toc450087566)

[2.3.2 Regresor veku 38](#_Toc450087567)

[2.4 Integrácia natrénovaných modelov do aplikácie 39](#_Toc450087568)

[2.4.1 Nájdenie a frontalizácia tváre pomocou OpenCV 40](#_Toc450087569)

[2.4.2 Implementácia klasifikátora, regresora pomocou Caffe API pre C++ 40](#_Toc450087570)

[3 Experiment a výsledky 41](#_Toc450087571)

[3.1.1 Klasifikátor pohlavia 41](#_Toc450087572)

[3.1.2 Regresor veku 47](#_Toc450087573)

[4 Záver 52](#_Toc450087574)

[Zoznam použitej literatúry 53](#_Toc450087575)

Zoznam obrázkov

Obrázok 1 – Ukážka viacvrstvovej neurónovej siete 6

Obrázok 2 – ilustrácie aktivácie neurónu 7

Obrázok 3 – matica váh 9

Obrázok 4 – grafické vysvetlenie konvolúcie 11

Obrázok 5 – ukážka architektúry konvolučnej siete 12

Obrázok 6 – ilustrácia posuvu konvolučného filtra 14

Obrázok 7 – ukážka konvolučných filtrov a feature máp 15

Obrázok 8 – obyčajné ReLU, Leaky ReLU, Parametrizované ReLU 20

Obrázok 9 – príklad dropoutu 21

Obrázok 10 – porovnanie dĺžky trénovania GPU vs CPU 24

Obrázok 11 – filtrácia natočenia tvárí 29

Obrázok 12 – príklad frontalizácie tváre 32

Obrázok 13 – príklad vyhodnotenia testovacej chyby veku 39

Obrázok 14 – screenshot z GUI aplikácie 40

Obrázok 15 – východisková architektúra siete pre rozpoznávanie pohlavia 42

Obrázok 16 – vývoj estimačnej a validačnej chyby pre klasifikátor pohlavia 45

Obrázok 17 – falošné negatíva, anotovaní muži klasifikovaní ako ženy 45

Obrázok 18 – falošné pozitíva, anotované ženy klasifikované ako muži 46

Obrázok 19 – vizualizácia konvolučných filtrov a feature máp 1. konvolučnej vrstvy pre klasifikátor pohlavia 46

Obrázok 20 – východisková architektúra siete pre rozpoznávanie veku 47

Obrázok 21 – vývoj estimačnej a validačnej chyby pre regresor veku 49

Obrázok 22 – subjekty s najlepšie odhadnutým vekom 50

Obrázok 23 – subjekty s najhoršie odhadnutým vekom 51

Obrázok 24 – vizualizácia konvolučných filtrov a feature máp 1. konvolučnej vrstvy pre regresor veku 51

Prevzaté obrázky

Zdroj:https://upload.wikimedia.org/wikipedia/commons/6/67/Convolution\_of\_spiky\_function\_with\_box.gif

Obrázok 5 :

Zdroj: http://devblogs.nvidia.com/parallelforall/wp-content/uploads/sites/3/2015/11/fig1.png

Obrázok 7 :

http://cs.nyu.edu/~yann/research/sparse/conv-mp-face.png

Obrázok 10:

http://devblogs.nvidia.com/parallelforall/wpcontent/uploads/sites/3/2015/03/cuDNN\_Perf\_chart011.png

Obrázok 11 https://i-msdn.sec.s-msft.com/dynimg/IC584331.png

Zoznam tabuliek

Tabuľka 1 – počet obrázkov a subjektov s pohlavím 30

Tabuľka 2 – rozdelenie obrázkov a subjektov s vekom 31

Tabuľka 3 – histogramy trénovacej množiny vek pred a po zväčšení 33

Tabuľka 4 – confusion matica 37

Tabuľka 5 – príklad vybalansovanej chyby podľa vekových kategórií 39

Tabuľka 6 – výsledky sietí s dvoma konvolučnými vrstvami pre klasifikátor pohlavia 43

Tabuľka 7 – výsledky sietí s troma konvolučnými vrstvami pre klasifikátor pohlavia 44

Tabuľka 8 – vypočítané priemerné hodnoty chýb pre východiskový model 44

Tabuľka 9 – confusion matica pre fold 0 pre východiskového modelu 44

Tabuľka 10 – výsledky sietí pre rozpoznávanie veku 48

Tabuľka 11 – výsledky sietí pre rozpoznávanie veku po pridaní GROUPS databázy 48

Tabuľka 12 – podrobné výsledky pre vekové kategórie regresora veku 50

# Úvod

Slávny český spisovateľ Karel Čapek v knihe R.U.R prvýkrát v  histórií zadefinoval pojem „*robot“.* Roboti pracovali v továrňach, aby ľudia nemuseli, ale jedného dňa sa vzbúrili, zničili ľudstvo a ovládli svet. Tento koncept sa s nástupom počítačov stal veľmi obľúbený a filmári sa už 30 rokov točia okolo myšlienky, že sa stroje, ktoré nám budú robiť sluhov, raz povstanú a ovládnu nás.

Laika by rapídny nástup a  rozvoj umelej inteligencie za posledných 10 rokov mohol znepokojovať, ale my sme sa napriek všetkých hrozbám ( ktorým neveríme ) rozhodli prípadným novým vládcom Zeme pomôcť a v tejto práci sa venujeme téme klasifikácie obrazu ľudských tvárí pomocou konvolučných neurónových sietí.

Umelé neurónové siete posledných 10 rokov zažívajú svoje zlaté obdobie. Veľké technologické korporácie ako Google, Microsoft alebo Facebook investujú milióny dolárov do ich výskumu a ukazuje sa, že toto zjednodušené napodobenie ľudského mozgu, je jednou z najefektívnejších a najuniverzálnejších metód strojového učenia. Posledné roky sa stali populárne konvolučné siete určené na rozpoznávanie obrazu. Môžeme nájsť desiatky vedeckých publikácií, ktoré dokazujú, že sa jedná o veľmi efektívny nástroj.

V práci sa zaoberáme tým, ako pomocou konvolučných neurónových sietí implementovať rozpoznávanie pohlavia a veku u ľudí. V prvej kapitole si priblížime teóriu a vysvetlíme si čo sú obyčajné a konvolučné neurónové siete.

V druhej kapitole si ukážeme ako sme na trénovanie neurónových sietí použili populárny Caffe [6] framework, ako s ním efektívne pracovať a ako použiť natrénované siete do GUI aplikácie. V závere práce si rozoberieme, akým spôsobom sme robili experiment a čo jeho výsledky znamenajú.

# Umelé neurónové siete

Umelé neurónové siete ( artificial neural networks ) sú momentálne jednou z najpopulárnejších a najpoužívanejších metód strojového učenia. Tieto matematické modely sú inšpirované biologickými neurónovými sieťami v mozgu. Keďže stále presne nevieme ako funguje učenie biologických neurónových sietí, nejedná sa o presnú simuláciu reálnych biologických procesov, ale len o ich čiastočné napodobenie a niektoré typy sietí vôbec nemajú prírodný náprotivok.

V práci si vysvetlíme jednoduchú plne prepojenú neurónovú sieť učiacu sa pomocou algoritmu spätnej propagácie, čo neskôr rozšírime o hlboké učenie ( deep learning ) a konvolučné siete. Nakoniec si priblížime si, ako sme ich použili na riešenie problému rozpoznávanie pohlavia a veku. [1]

## Viacvrstvová plne prepojená neurónová sieť

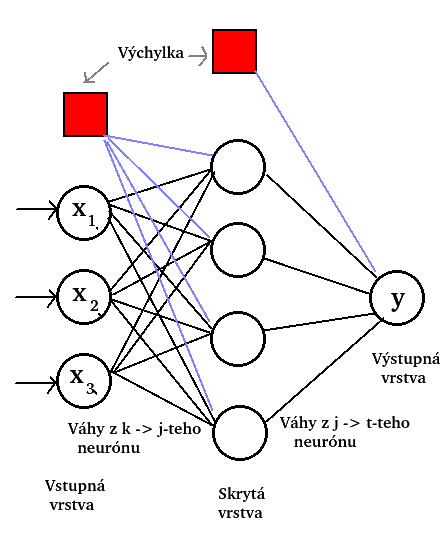
V tejto časti si rozoberieme učenie s učiteľom plne prepojenej neurónovej siete pomocou algoritmu spätnej propagácie. Pri tomto type učenia potrebujeme mať trénovaciu množinu, ktorá sa delí na vstupy a výstupy. Formálne trénovaciu množinu môžeme definovať ako pole *n* rozmerných vektorov, pričom výstupom je skalár.

Formálnejší zápis :

Kde sú trénovacie príklady a sú ich príslušné výstupy. Čo tieto vstupy a výstupy reprezentujú záleží od typu úlohy. V prípade klasifikácie chceme určiť do akej triedy patrí trénovací vzor. Ako príklad môžeme uviesť túto prácu, ktorej cieľom bolo natrénovať sieť schopnú určiť pohlavie človeka, teda rozlíšiť medzi dvoma triedami : mužom a ženou. Pre túto úlohu by vstupom do siete bol vektorizovaný obrázok.

Plne prepojenú sieť môžeme chápať aj ako orientovaný acyklický graf ( existujú aj cyklické siete ), kde uzly predstavujú jednotlivé neuróny a váha hrany ( zväčša na intervale <0,1> ) reprezentuje relatívnu dôležitosť informácie putujúcej z jedného neurónu do druhého. Neuróny sú delené do vrstiev, pričom vždy existuje jedna vstupná, jedna výstupná a nejaký počet skrytých vrstiev medzi nimi. S narastajúcim počtom neurónov alebo vrstiev by mala narastať aj množina hypotéz a zložitosť trénovania.

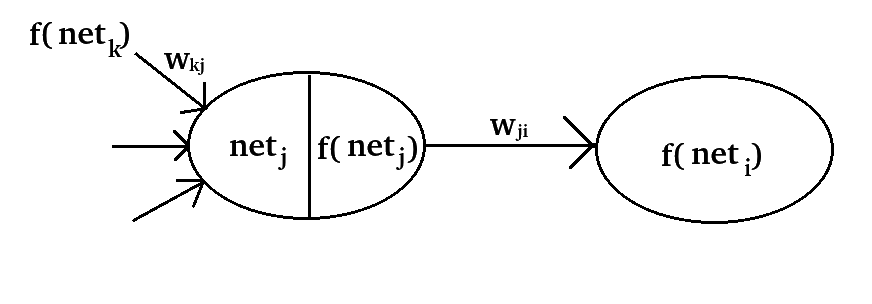
Na *obrázku 1*. môžeme vidieť príklad trojvrstvovej plne prepojenej siete, ktorej vstup je trojrozmerný vektor, výstup je skalár a skrytá vrstva obsahuje 4 neuróny. Vrstvy sú rozdelené horizontálne a neurón z l-tej vrstvy posiela svoj výstup do všetkých neurónov l+1-vej vrstvy. Počas učenia upravujeme váhy jednotlivých hrán tak, aby výsledná sieť správne klasifikovala čo najväčší počet trénovacích príkladov. Ak na vstup dostaneme nesprávnu odpoveď, musíme váhy upraviť takým spôsobom, aby sieť začala odpovedať správne, alebo aby sa k tomu blížila. (gradient descent) Naučená sieť (model) je teda usporiadanie váh. [1] [2] [3]



Obrázok 1 – Ukážka viacvrstvovej neurónovej siete

### Učenie pomocou algoritmu spätnej propagácie

Predstavme si situáciu, že máme nejaké nastavenie váh a na základe správnej alebo nesprávnej odpovede na trénovací príklad chceme príslušne váhy upraviť ( napr. lebo klasifikácia pohlavia neprebehla správne ). Na *obrázku 2.* je zobrazený j-ty neurónu v skrytej vrstve, jeho váhy a výstup. Aktivácia je definovaná ako váhovaná suma aktivácií z predošlých neurónov, ktorú použijeme ako vstupnú hodnotu do funkcie ***f.*** Toto má aj biologicky základ, pretože neurón v mozgu prijíma signály z iných neurónov a priraďuje im nejakú dôležitosť(váhu). Aktivačná funkcia ***f*** môže byť rôzna, pričom ju volíme podľa typu úlohy, ktorú riešime ( sigmoid, hinge, softmax atď. ) [3] [2]



Obrázok 2 - ilustrácie aktivácie neurónu

Formálne :

Kde K je sú neuróny z predošlej vrstvy spojené s j-tym neurónom v súčasnej vrstve a je váha medzi j-tym a k-tym neurónom. V prvej skrytej vrstve používame namiesto aktivácie z predošlej vrstvy prvok z trénovacieho príkladu ( vstup do siete ) , kde ***n*** je dimenzia vstupu. Postupné prechádzanie zo vstupnej vrstvy do výstupného neurónu voláme **dopredné šírenie ( feed-forward).**

Ďalej si musíme zadefinovať chybovú funkciu E , podľa ktorej budeme určovať ako zle sme klasifikovali trénovací príklad a podľa nej sa rozhodneme, či budeme upravovať váhy.

Kde je očakávaná hodnota pre vstup a  je to, čo nám neurón reálne odpovedal. Ak je E > 0, tak nám sieť klasifikovala trénovací príklad nesprávne, čo znamená, že musíme nejakým upraviť váhy, a teda potrebujeme neuróny v predošlých vrstvách informovať o tom, že niečo treba zmeniť. To robíme algoritmom **spätnej propagácie chyby,** ktorý využíva gradientovú optimalizačnú metódu. Pokiaľ máme správne dáta a úloha je riešiteľná, sieť by mala postupne skonvergovať k riešeniu. Váhy budeme upravovať tak, že k nim pripočítame parciálnu deriváciu chyby E podľa váhy nasledovne.

Kde je rýchlosť učenia. Ďalej si musíme odvodiť pomocou reťazového pravidla, pričom je parametrom funkcie a je parametrom funkcie ***f****.*

Parciálna derivácia výrazu je deriváciou funkcie ***f***a tretí výraz je hodnotou funkcie *f* v k-tom neuróne, pretože sa k ostatným členom v sume správame ako ku konštantám.

Problémom je výraz , ktorý budeme derivovať pre neuróny výstupnej vrstvy inak ako pre neuróny skrytých vrstiev. Pre výstupnú vrstvu to bude vyzerať takto

Teraz je potrebné to isté zderivovať pre skryté vrstvy. Opäť použijeme reťazové pravidlo tak, že E si rozpíšeme do sumy. I budú neuróny z ( *l + 1* ) vrstvy, teda neuróny bližšie k výstupnej vrstve. Využili sme fakt, že je vstupom do , čo môžeme vidieť na *obrázku 2.*

Tretí výraz je jednoduchý, podobne sme derivovali už predtým . Zavedieme rekurzívny člen , ktorý nám zabezpečí to, že nebudeme musieť riešiť prvé dve parciálne derivácie

Ak to dáme dokopy máme

Keď sa pozrieme ako je definované , môžeme všimnúť, že sa nám v tejto rovnici skrýva . Dostaneme predpis pre skrytú vrstvu

Pre výstupnú

Zmena váhy bude

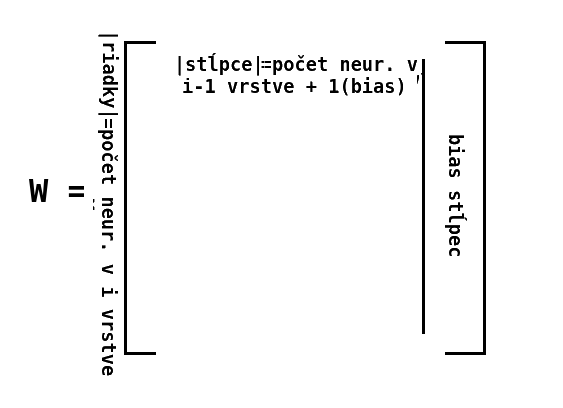
Tento vzorec si vieme dať do maticovej formy nasledovne

pre výstupnú vrstvu

pre skrytú vrstvu

je transponovaná matica , z ktorej je odstránený stĺpec s výchylkami(bias).  je *Hadamardov súčin* ( *Hadamard product* )

bude teraz vektor, ktorého dimenzia zodpovedá počtu neurónov l-tej vrstvy a ) je vektor aktivácií l-tej vrstvy. je matica váh medzi neurónmi *l* a *l+1* vrstvy. Vysvetlenie W môžeme vidieť na obrázku 3



Obrázok 3 - matica váh

### Algoritmus spätnej propagácie



## Konvolučné siete ( CNN )

Aby sme pochopili, prečo používame konvolučné siete, musíme si priblížiť situáciu, v ktorej by sme sa snažili spracovávanie obrázka riešiť pomocou plne prepojenej siete. Pri spracovávaní obrázka by počet neurónov musel zodpovedať počtu pixelov a zároveň by každý jeden neurón musel byť spojený hranou s každým pixelom. Ak by sme napríklad mali bitmapu veľkosti NxN pixelov, počet neurónov len v prvej vrstve by bol  a počet váh by bol n . Už pri čiernobielom obrázku veľkosti 64x64 by sme mali okolo 16 tisíc váh pre jeden neurón. Pridávaním ďalších neurónov a skrytých vrstiev, by počet váh stúpol na také množstvo, že pri dnešných počítačoch prakticky nie je možné realizovať trénovanie. [2] [4] [5]

Plne prepojené siete žiadnym spôsobom neberú do úvahy priestorovú štruktúru obrázkov. Veľmi vzdialené pixely sú spracované s takou istou dôležitosťou ako tie, ktoré sú blízko seba. Využitie konvolúcie a na nich založených neurónových sieťach, rieši problém veľkého počtu parametrov, tak že váhy sa zdieľajú a zároveň je využitá priestorová štruktúra obrázkov.

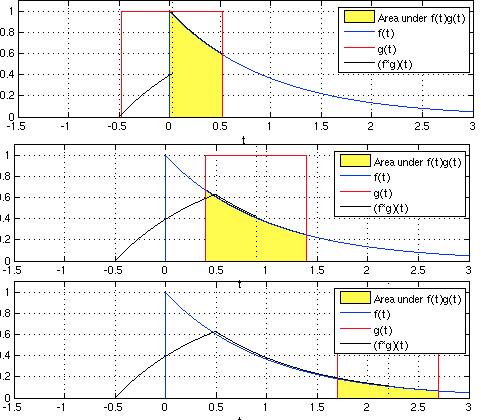
Ďalším problémom obyčajných sietí je to, že dobre nezvládajú použitie viacerých skrytých vrstiev. Očakávali by sme, že použitím hlbokej siete zvyšujeme schopnosť rozpoznávať zložitejšie vzory a zlepšíme aj celkovú presnosť algoritmu, ale experimentálne by sme zistili, že pre ten istý problém dosiahneme horšie výsledky ako s jednou skrytou vrstvou. Je to spôsobené tzv. *problémom miznúceho gradientu* (vanishing gradient problem) . Ak by sme sa pozreli na to akým spôsobom sa menia váhy na jednotlivých vrstvách, zistili by sme, že neuróny v prvých skrytých vrstvách sa takmer vôbec neučia alebo sa učia len veľmi pomaly. Čím viac sme vzdialení od výstupnej vrstvy, tým sa gradient exponenciálne zmenšuje a  zmena váh je menej významná.

V určitých prípadoch môže nastať opačný problém a to taký, že gradient sa bude smerom k vstupnej vrstve exponenciálne zväčšovať. To nazývame *problémom* *explodujúceho gradientu* ( exploding gradient problem ) .

Pri komplexných sieťach v praxi nastáva problém *nestabilného gradientu* (unstable gradient problem), teda rôzne vrstvy sa budú učiť rozdielnymi rýchlosťami. (kombinácia oboch problémov). [5]

### Konvolúcia

Konvolúcia je definovaná ako integrál prekryvu dvoch funkcií a*,* respektíve ju môžeme chápať ako funkciu, ktorá nám určuje, ako sú a„zmiešané”. Na obrázku 4. je zobrazená červenou farbou a funkciamodrou farbou***.*** Funkciu konvolúcie máme zobrazenú sivou a môžeme vidieť, že jej hodnota v čase ***t*** zodpovedá obsahu prekryvu týchto dvoch funkcií. Ako postupne posúvame  ***,*** môžeme vidieť, že sivá krivka( začínajúca v -0.5 na treťom grafe ) rastie dovtedy, kým rastie prekryv. Klesať začne so zmenšovaním žltej plochy.

[](https://upload.wikimedia.org/wikipedia/commons/6/67/Convolution_of_spiky_function_with_box.gif)

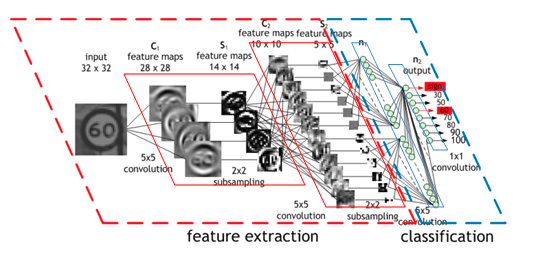
Obrázok 4 – grafické vysvetlenie konvolúcie

Formálne môžeme definovať konvolúciu nasledovne

( je znak matematického operátora konvolúcie )

### Architektúra konvolučnej siete

Na rozdiel od architektúry obyčajnej siete, v tomto type existuje siete viacej druhov vrstiev plniacich špecifické úlohy. Algoritmy dopredného šírenia a spätnej propagácie ostávajú nezmenené. Na *obrázku 5* môžeme vidieť príklad toho ako vyzerá architektúra konvolučnej siete, kde na začiatku je vstupná vrstva nasledovaná konvolučnou, pooling, normalizačnou a nakoniec plne prepojenou. Jednotlivé vrstvy a ich funkcie si v tejto kapitole vysvetlíme.



Obrázok 5 – ukážka architektúry konvolučnej siete

### Vstupná vrstva

Môžeme si ju predstaviť ako mriežku neurónov, ktorých počet bude zodpovedať veľkosti obrázka, respektíve počtu pixelov daného vynásobením šírky s výškou, čiže WxH. Nemusíme sa obmedziť len na dvojrozmernú obdĺžnikovú mriežku a v prípade, že by sme chceli spracovávať RGB obrázok, mali by sme pre každú farbu samostatnú mriežku. Teda WxHx3.

Môže nastať situácia, že máme dobrú architektúru a kvalitné dáta, ale z nejakého dôvodu trénovacia chyba neklesá. Preto je potrebné dáta najprv normalizovať.

#### Normalizácia vycentrovaním do [0,0]

Predpokladáme, že obrázky z množiny X sú už vektory. Potrebujeme si vypočítať priemerný vektor *c* respektíve geometrický stred dát, teda od každého vstupného vektor odpočítame *c***.**  Platí N = |X|

A každý vektor nahradíme novým vektorom

#### Normalizácia škálovaním

Niekedy vycentrovanie do počiatku súradnicovej sústavy nestačí a dáta je potrebné správne naškálovať. Najčastejšie chceme, aby sa hodnoty pohybovali v intervale <0,1>, čo urobíme tak, že vydelíme každý vstupný vektor smerodajnou odchýlkou.

Následne dostaneme nové vektory

V praxi pri použití 8-bitových obrázkov vieme, že pixel má hodnoty od 0 po 255. Preto nám stačí vynásobiť všetky dáta číslom 1/256 = 0.00390625.

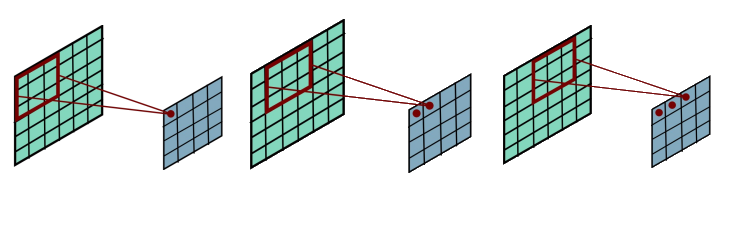
### Konvolučná vrstva

V plne prepojenej siete by sme mali každý pixel zo vstupného vektorizovaného obrázka spojený so všetkými neurónmi prvej skrytej vrstvy. Teda pri vstupne WxH a M neurónoch by sme mali WxHxM váh. Neuróny v konvolučnej vrstve nebudú prepojené s každým pixelom zo vstupu, namiesto toho budú spojené s malým lokalizovaným regiónom vstupného obrázka, ktorý nazývame **receptívne pole ( receptive field ).** [4] [5]

Konvolučná vrstva sa skladá z množiny naučiteľných **konvolučných filtrov ( kernelov )**. Konvolučný filter je malá FxF matica, ktorej veľkosť sa rovná veľkosti receptívneho poľa a jej hodnoty môžeme chápať ako maticu zdieľaných váh medzi vstupnou vrstvou ( receptívnym poľom ) a konvolučnou vrstvou. Tento filter je spoločný pre všetky neuróny konvolučnej vrstvy, teda pre jeden kernel a jeden vstupný obrázok máme len FxF váh. Oproti obyčajným sieťam sa výrazne znižuje počet parametrov, ktoré musíme trénovať a proces učenia sa výrazne zrýchli.

Počas dopredného šírenia filter postupne posúvame v horizontálnom a vertikálnom smere po obrázku, pričom výsledkom ( aktiváciou konvolučnej vrstvy ) bude dvojrozmerná tzv. „*feature mapa*“ , ktorej veľkosť bude ((W - F)/S + 1) x ((H - F)/S + 1), kde S je veľkosť kroku (*stride*, posun).

Na *obrázku 6.* môžeme vidieť ako by takéto posúvanie a aplikácia filtra vyzerala. Zelený obrázok reprezentuje vstup, červenou je zobrazené receptívne pole, respektíve konvolučný filter a modrou výstupná feature mapa. Aktivácia bude ekvivalentná skalárnemu súčinu FxF filtra a pixelov z receptívneho poľa, na ktoré aplikujeme filter.



Obrázok 6 – ilustrácia posuvu konvolučného filtra

Formálnejšie je aktivácia neurónu konvolučnej vrstvy ( z *feature mapy )* zapísaná ako [2]

Kde **i, j** sú súradnice neurónu v rámci *feature mapy,* ***f*** je aktivačná funkcia( nie nutne potrebná ), ***l*** je index vrstvy a ***s*** veľkosť kroku ( stride ). Zoberieme malú FxF maticu **R**reprezentujúcu receptívne pole, ktorého centrum je a urobíme operáciu konvolúcie medzi **R** a maticou filtra **K**. Operáciu konvolúcie dvoch 3x3 matíc definujeme nasledovne.


\begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{bmatrix}
*
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
= 
(1*i)+(2*h)+(3*g)+(4*f)+(5*e)+(6*d)+(7*c)+(8*b)+(9*a)


Ak toto zovšeobecníme, bude to zodpovedať vyššie uvedenému vzťahu pre . Názov konvolučné siete používame kvôli tomu, že nerobíme nič iné ako konvolúciu medzi K a R.

Ďalšou výhodou toho, že sa jedná o jednoduchú maticovú operáciou je, že v dnešnej dobe je možné implementovať trénovanie konvolučných sietí so značným zrýchlením  použitím grafických kariet.

Úlohou filtra je extrahovať zo vstupného obrazu príznaky ( features ), ktoré pomáhajú riešiť daný problém. Napríklad, ak chceme detegovať, či sa na obrázku nachádza ľudská tvár, pravdepodobne budeme používať nejaký filter s dobrou schopnosťou detekcie hrán, ktorý vyprodukuje *feature mapu* s dobre zvýraznenými okrajmi tváre. Príklad takých filtrov a výsledných *feature máp* je možné vidieť na *obrázku 7*.



Obrázok 7 – ukážka konvolučných filtrov a feature máp

Existuje niekoľko známych kernelov schopných riešiť určité typy úloh akými je detekcia tváre, očí, ostrenie alebo rozmazávanie obrazu, ale pri mnohých problémoch, filter nepoznáme. Cieľom konvolučných sietí je pomocou algoritmu spätnej propagácie nájsť tieto konvolučné kernely, čo znamená, že počas trénovania budeme iteratívne upravovať váhy filtra presne tak, ako sme to robili pri plne prepojených sieťach.

#### Parametre konvolučnej vrstvy

Konvolučná vrstva má niekoľko parametrov, podľa ktorých si vieme určiť počet a rozmery *feature máp,*  počet neurónov a ich príslušných váh, veľkosť kroku atď. Zároveň je dôležité vedieť odhadnúť akým spôsobom sa odrazí zmena parametra vo výpočtovej zložitosti siete a vo výslednej chybe.

**Počet filtrov, počet výstupných feature máp ( K )**

Spomínali sme, že vo vstupnej vrstve sa nemusíme obmedziť na dvojrozmerný WxH vstup. To isté platí pre konvolučnú vrstvu, kde si môžeme zvoliť ľubovoľný počet výstupných feature máp. Ich množstvo bude rovnať počtu kernelov. V praxi sa používajú desiatky a väčšinou je toto číslo deliteľné dvoma alebo je mocninou dvojky.

Aj keď neurónom je pixel na *feature mape*, toto nastavenie môžeme chápať ako analógiu počtu neurónov v skrytej vrstve pri plne prepojených sieťach. Ak by sme chceli dosiahnuť vyššiu komplexitu a potenciálne lepší výsledok v obyčajnej sieti, museli by sme zväčšovať počet neurónov v skrytých vrstvách. Tu isté budeme dosiahneme zvyšovaním tohto parametra. [4]

**Rozmery filtra ( F )**

Definuje šírku a výšku filtra. Najčastejšie sa používa štvorcový filter a jeho tretí rozmer by mal zodpovedať počtu kanálov vstupného obrázka. Napríklad pre RGB obrázok by mal filter rozmery FxFx3

**Veľkosť kroku(stride - S)**

Definuje o koľko pixelov sa bude posúvať filter počas dopredného šírenia. Napríklad ak je krok rovný 2, stredy receptívnych polí budú [1,1], [3,1], [5,1] atď. Platí, že čím menší je krok, tým je feature mapa väčšia, čo spôsobí, že máme viac neurónov a trénovanie sa predĺži. Naopak so zväčšovaním veľkosti kroku strácame informácia zo vstupu a dôsledkom môže byť nižšia presnosť algoritmu.

Dôležité pre nastavenie kroku je, aby výrazy a boli celé čísla. Ak toto neplatí, nastavili sme tento parameter nesprávne a nie sme schopní napasovať filter do vstupu. [4]

**Zarovnanie ( zero-padding - P )**

Obaľuje vstupný obrázok ( alebo v hlbších vrstvách feature mapu ) pixelmi s nulovými hodnotami. Umožňuje nám kontrolovať veľkosť výstupných feature máp. Tzv. *zero-padding* použijeme napríklad v prípade, že chceme, aby sa veľkosti vstupu a výstupu rovnali.

**Grupovanie ( G )**

Ak je *G > 1*, obmedzíme konektivitu len na podmnožinu výstupu. Vstupné a výstupné neurónu budú rozdelené do G skupín a  i-ty neurón vo vstupnej vrstve bude spojený s i-tym neurónom vo výstupnej. Opäť týmto zmenšujeme počet parametrov a teda aj komplexitu siete [6]

**Váhový a bias filter ( caffe )**

Určuje ako na začiatku trénovania inicializovať váhy a výchylky kernelov. Napríklad môžeme zvoliť gaussovskú distribúciu

**Veľkosť výstupnej feature mapy**

Ak má vstupná vrstva rozmery . Celkový počet váh bude potom . Plus jedna kvôli výchylkám a *feature mapa* bude mať rozmery :

#### Spätná propagácia chyby

Z predpisu pre dopredné šírenie si chceme odvodiť presne ako v plne prepojenej sieti gradient chyby , pričom poznáme hodnoty E na konvolučnej vrstve. [2]

Použijeme reťazové pravidlo. Keďže máme zdieľané váhy, musíme urobiť sumu cez všetky neuróny z feature mapy, ktoré používajú váhu . Tých je (N-F+1) x (N-F+1) , , čo je počet neurónov (pixelov) vo feature mape.

Výraz a vieme zderivovať ľahko a dostaneme

Opäť použijeme reťazové pravidlo a dostaneme

Parciálna derivácia je len a teda dostávame

Ostáva nám rekurzívne propagovanie chyby do ďalších vrstiev a teda musíme vyriešiť výraz .

### Pooling vrstva ( sub-sampling )

Nasleduje po konvolučnej vrstve a jej úlohou je zredukovať veľkosť *feature mapy* tak, aby sme informáciu z feature mapy zjednodušili, respektíve z nej dostali najdôležitejšiu časť. Tým zároveň redukujeme parametre siete a znižujeme možnosť potenciálneho preučenia.

Podobne ako v konvolučnej vrstve, aj tu zvolíme veľkosť kroku a použitím aktivačnej funkcie zredukujeme malú PxP časť feature mapy na jeden pixel. V prípade, že máme viac feature máp, na každú aplikujeme pooling nezávisle. Ak je vstup mapa veľkosti , potom rozmery zmenšenej feature mapypo aplikácii poolingu budú.

Najčastejšie používaná technika je tzv. max pooling. Z receptívneho poľa sa vždy vyberá pixel s najvyššou hodnotou. Ako aktivačné funkcie ( pooling ) sa zvyknú používať aj iné funkcie, ale v praxi sa ukázalo, že max dosahuje najlepšie výsledky. Výsledný v zmenšenej feature mape pre max pooling bude

#### Parametre pooling vrstvy

**Aktivačná funkcia**

Môžeme si definovať akú funkciu chceme použiť. *Max pooling* môžeme nahradiť napríklad priemerom alebo L2 normou.

**Rozmery filtra ( F )**

Tak ako v konvolučnej vrstve, aj tu vieme definovať veľkosť receptívneho poľa

**Zarovnanie ( zero-padding - P )**

Totožné so zarovnávaním v konvolučnej vrstve

**Veľkosť kroku(stride - S)**

Podobne ako v konvolučnej vrstve aj tu vieme definovať veľkosť kroku. Najčastejšie sa používa hodnota 2 spolu s 2x2 receptívnym poľom, čo znamená, že feature mapa sa zmenší na 25% pôvodnej veľkosti.

#### Spätná propagácia chyby

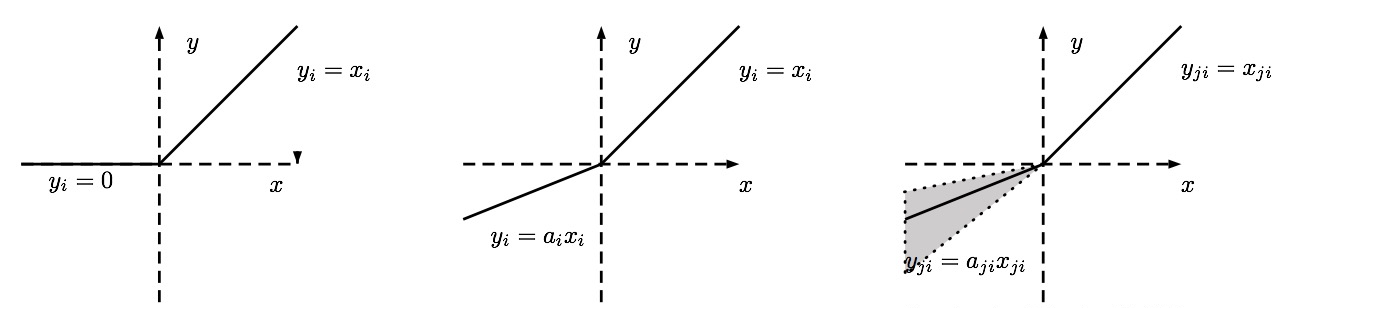
Keďže sa pooling vrstva nič neučí, netreba špeciálne riešiť odvodenie . Chyba sa bez ďalších úprav pošle do ďalšej vrstvy.

### Rectified-Linear Unit ( ReLU )

Jedná sa o vrstvu, ktorá všetky záporné hodnoty zo vstupu vynuluje alebo nastaví na vybranú zápornú hodnotu. V praxi sa ukázalo, že normalizácia pomocou ReLU dáva lepšie výsledky ako logistická funkcia (sigmoida). Dôvod, prečo je to tak, zatiaľ nie je známy a  máme len argumenty založené na experimentoch.[5]

Ďalšou výhodou je výrazné urýchlenie konvergencie gradientovej metódy. Zároveň ReLU nie je taká náročná na výpočet ako logistická funkcia alebo tanh, čo tiež zrýchli trénovanie.

Nevýhodou je, že niektoré neuróny môžu byť počas trénovania deaktivované. V prípade, že by ReLU vrstvou prechádzal priveľký gradient, môže sa stať, že sa váhy neurónu z nejakého dôvodu zmenia na 0 a už nikdy nebudú môcť byť počas trénovania aktivované. Môže nastať situácia, že 40 percent neurónov v sieti bude deaktivovaných, teda nebudú vytvárať žiadnu odpoveď na vstup. Zabrániť tomu môžeme miernym vylepšením a to tak, že namiesto 0 bude mať funkcia malý záporný sklon. ( podpora v caffe ). Ďalšie vylepšenie je, že tento záporný sklon bude čiastočne náhodný. ( *obrázok 8.* ) [7]



Obrázok 8 – obyčajné ReLU, Leaky ReLU, Parametrizované ReLU

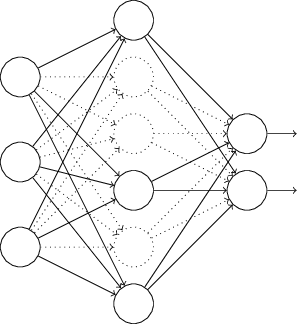
### LRN vrstva ( Local Response Normalization )

LRN vrstva napodobňuje „laterálnu inhibíciu“ normalizovaním oblastí z receptívneho poľa, odpozorovanú z biologických neurónových sietí. Ukázalo sa, že tieto vrstvy nemajú veľký vplyv na výsledky a v moderných sieťach sa postupne prestávajú používať. Caffe však podporuje aj LRN normalizáciu.

Vstup do tejto vrstvy bude normalizovaným nasledovným vzorcom, pričom a  sú nami definované parametre. Veľkosť receptívneho poľa je definovaná *n*. [6]

### Dropout vrstva

Dropout je ďalšia regularizačná technika používaná na vylepšenie trénovacej chyby. Narozdiel od L1 a L2 regularizácie nemodifikujeme chybovú funkciu, ale modifikujeme sieť samotnú. Počas spätnej propagácie s pravdepodobnosťou *p* dočasne vymažeme časť neurónov v skrytých vrstvách, s tým, že vstupné a výstupné vrstvy ostanú nedotknuté. Výsledkom bude sieť, kde niektoré neuróny a ich váhy budú kompletne deaktivované. Príklad na *obrázku 9*.



Obrázok 9 – príklad dropoutu

Dopredné šírenie a spätnú propagáciu potom vykonávame nad touto modifikovanou sieťou a teda meníme len váhy neurónov, ktoré neboli deaktivované. Po úprave váh budú deaktivované neuróny vrátené do siete. Tento proces opakujeme veľakrát s tým, že zakaždým náhodne vyberieme inú podmnožinu neurónov na deaktiváciu.

Keď vyhadzujeme jednotlivé neuróny, môžeme to chápať tak, že trénujeme rôzne siete. Preto si dropout procedúru môžeme predstaviť ako priemerovanie efektov alebo hlasovanie veľkého počtu neurónových sietí, čo sa ukázalo byť efektívne pri predchádzaní preučenia. Dropout môžeme chápať ako spôsob siete, ktorý má zabrániť strate informácie pri prípadnej eliminácii časti neurónov. Biologicky to môžeme prirovnať napríklad k opitiu sa, keď človeku umrie veľké množstvo neurónov, ale sieť je dostatočne dobre naučená rozprávať, čítať alebo chodiť a  pár miliónov mŕtvych mozgových buniek nespôsobí to, že by sme tieto schopnosti stratili. Dropout teda pomáha pri decentralizovaní informácie v neurónovej sieti.

#### Parametre dropout vrstvy

**Dropout koeficient ( dropout ratio )**

Definuje pravdepodobnosť *p* , že neurón bude počas trénovania deaktivovaný.

### Plne prepojená vrstva

Po niekoľkých konvolučných a pooling vrstvách, keď už sú *feature mapy* dostatočne zredukované, sa na výslednú klasifikáciu alebo regresiu použije plne prepojená neurónová sieť. Presne taká, ktorú sme si vysvetlili v predošlej kapitole. Neuróny v tejto vrstve sú spojené s každým pixelom vo feature mapách a prebieha  vysokoúrovňové rozhodovanie. ( finálna klasifikácia, regresia ). Zároveň sa nám tým stráca informácia o priestorovej štruktúre feature mapy a namiesto toho sa do úvahy sa berie globálna informácia z  obrázka. Preto za plne prepojenou vrstvou môže nasledovať len ďalšia plne prepojená alebo výstupná vrstva.

#### Parametre plne prepojenej vrstvy

**Počet výstupov ( caffe )**

Definuje počet neurónov v ďalšej skrytej vrstve alebo vo výstupnej vrstve

**Váhový filter a bias filter( caffe )**

Ten istý parameter s tou istou funkciou ako pri konvolučnej vrstve

### Vrstva chybovej funkcie

V konvolučných sieťach a vo väčšine metód strojového učenia je učenie riadené chybovou funkciou. ( Loss, cost, objective function v angličtine ). Informácia zo vstupného obrázka prejde dopredným šírením do poslednej vrstvy a vyprodukuje nejaký výstup ( predikciu ), ktorý je porovnaný s  očakávanou hodnotou. Na základe rozdielu medzi očakávaným a predikovaným výsledkom sa vypočíta gradient a chyba je šírená pomocou spätnej propagácie celou sieťou.

#### Logistická funkcia

Jedna z najstarších a v minulosti najpoužívanejších aktivačných funkcií v neurónových sieťach. Hodnoty sú v intervale <0,1> a zväčša sa rozhodujeme, či je neurón aktivovaný podľa toho, či f(x) > 0.5. Môžeme ju použiť jednak na regresiu, ale aj binárnu klasifikáciu Veľakrát je ale výhodnejšie použiť inú funkciu. [6]

Nevýhodou logistickej funkcie ( sigmoida ) je, že môže spôsobiť spomínaný problém miznúceho gradientu, preto ju v hlbokých sieťach nie je vhodné používať na normalizáciu, ale namiesto toho ju nahradiť ReLU. Použitie v poslednej vrstve je v poriadku. Druhý problém je, že stredná hodnota nie je 0 a nehodí sa na úlohy, ktoré si vyžadujú hodnoty v intervale <-1,1> (použili by sme tanh )

Výhodou a dôvodom prečo je sigmoida taká populárna je to, že jej derivácia vyzerá veľmi pohodlne

#### Hinge

Vhodná na riešenie klasifikácie ( jedna vs. jedna alebo jedna vs. viac). Inšpirovaná SVM( support vector machine ) .

Predpokladaný výstup označme *t* je a *y* bude predikovaná hodnota sieťou. Potom predpis funkcie je nasledovný

#### Euclid

Používa sa na regresné úlohy, teda ak chceme odhadovať presnú hodnotu. ( napríklad odhadovanie veku ). Počíta L2 normu nasledujúceho výrazu. N je veľkosť trénovacej množiny.

#### Softmax

Vhodná a preferovaná možnosť pre akýkoľvek klasifikačný problém, špeciálne na klasifikáciu viacerých tried. K dimenzionálny vstupný vektor je zredukovaný na vektor, ktorého hodnoty sú v intervale <0,1> a súčet jeho hodnôt je 1. Softmax vracia vektor pravdepodobnostnej distribúcie x, teda výsledkom bude vektor, ktorý nám hovorí, aká je pravdepodobnosť, že daný vstup sa rovná konkrétnej triede .

Predpis funkcie

A zároveň platí

## Caffe

Je opensource framework určený na trénovanie konvolučných hlbokých neurónových sietí. Bol vyvinutý Berkeley univerzitou, konkrétne Berkeley Vision and Learning Center. Keďže zdrojový kód je voľne dostupný, projekt má (v čase písania tejto práce) dosť veľkú a živú komunitu. Caffe používa technológiu **CUDA ( Compute Unified Device Architecture ),** ktorá umožňuje spúšťať programy napísané v  jazykoch C, C++, Fotran atď. priamo na grafickej karte. Paralelizácia výpočtov mnohonásobne skráti dĺžku trénovania, čo robí z hlbokého učenia prakticky vykonateľnú úlohu. Porovnanie výkonu CPU a GPU pri trénovaní môžeme vidieť na *obrázku 10.* Technológia je dostupná len na grafických kartách od firmy NVIDIA, aj keď konkurenti ako AMD majú vlastné riešenie. [6]



Obrázok 10 – porovnanie dĺžky trénovania GPU vs CPU

### Použitie Caffe

Caffe používa na definíciu modelu konvolučnej siete *google protocol buffer,* čo je binárny protokol na výmenu dát zapísaný v textovom súbore a v caffe sú v ňom zapísané jednotlivé konvolučné, pooling, ReLU a ďalšie vrstvy. Týmto súborom sa dá definovať počet konvolučných vrstiev, ako idú za sebou, aktivačné funkcie a ostatné parametre, ktoré sme si vysvetľovali v predošlej kapitole. Podrobnejšie informácie si môže používateľ prečítať v dokumentácii na oficiálnej stránke projektu [6] . Použitie Caffe sa delí na niekoľko fáz, ktoré si priblížime v nasledujúcich podkapitolách.

#### Príprava dát

Caffe spracováva dáta vo forme *blobov(blob v AJ),* čo sú dáta (obrázky) transformované do N-dimenzionálnych polí. Tie sú uložené na disku v key-value databázach ( LevelDB [8], LMDB [9] ) a je na používateľovi, aby originálne obrázky previedol do jednej z nich. Všetky dáta by mali byť normalizované a musia mať tú istú veľkosť akú potom definujeme v trénovacom \*.prototxt súbore.

Druhým krokom je vypočítanie priemerného obrázka ( mean file - viď 1.2.3 ) , ktorý bude použitý na vycentrovanie dát. Výsledkom budú tri súbory : \*.binaryproto, \*.npy, \*.png. Caffe ponúka nástroje, ktoré s tým pomáhajú.

#### Definícia architektúry siete

V tejto fáze treba pripraviť samotnú architektúru siete vytvorením troch súborov : train\_val.prototxt, predict.prototxt, solver.txt. Train\_val.prototxt obsahuje definíciu siete pre trénovanie, môžeme tu tiež definovať cross validačné ( 2.2.2 ) fázy ( TEST, TRAIN ), hovoríme kde má hľadať dáta a ich anotácie, definujeme jednotlivé vrstvy a ich parametre.

Predict.txt je v podstate ten istý súbor ako train\_val.prototxt akurát sú odstránené všetky časti týkajúce sa testovacej fázy a vstupných databáz.

V solver.txt súbore špecifikujeme parametre trénovania ako:

* cesta k súboru train\_val.prototxt
* rýchlosť učenia
* momentum
* maximálny počet iterácii
* ako často robiť „snapshoty“
* spomaľovanie rýchlosti učenia
* typ solvera ( Stochastic Gradient Descent, Adaptive Gradient, Nesterov’s Accelerated Gradient atď. )

#### Trénovanie a výsledky

Jednoducho zavoláme program ( príkaz *./caffe train –solver=<cesta k solver.txt>* ), ktorému povieme, kde sa nachádzajú spomínané \*.prototxt súbory. Výsledkom trénovania je binárny protobuf (\*.caffemodel) súbor obsahujúci natrénovaný model. Ten potom môžeme načítať pomocou API v C++, Matlabe alebo Pythone a používať ho na našu klasifikačnú alebo regresnú úlohu, napríklad v demo aplikácii. Zároveň môžeme pomocou tohto API vizualizovať mnoho zaujímavých informácií ako napríklad priebeh učenia ( ako sa vyvíjala chyba ) , jednotlivé konvolučné filtre, feature mapy atď. Oficiálna stránka obsahuje pomerne dobré návody ako začať Caffe API používať.

## Podobné práce

V skratke si prejdeme podobné práce, ktoré sa zaoberajú problematikou klasifikácie ľudských tvárí.

### Klasifikácie veku a pohlavia použitím konvolučných sietí

Vedecký článok napísaný Gil Levim a Tal Hassnerom z The Open University v Izraeli, v ktorom sa autori zaoberajú použitím konvolučných neurónových sietí na problém klasifikácie pohlavia a veku. V článku tvrdia, že použili sieť s pomerne jednoduchou architektúrou ( len s 3 konvolučnými vrstvami ) kvôli tomu, aby predišli riziku preučenia a zároveň použili tú istú architektúru na obe úlohy. Ich vstupom bol 3 kanálový 227x227 RGB obrázok a ako databázu použili *Adience Benchmark.* Trénovanie prebiehalo na Amazon GPU stroji pomocou Caffe frameworku. Podľa experimentov im vyšlo, že trénovanie po dávkach ( image batches ) im pomohlo dosiahnuť značné zrýchlenie trénovania. Pri úlohe zisťovanie pohlavia sa im podarilo dosiahnuť približne 85% úspešnosť a pri zisťovaní veku bola úspešnosť len 50%. Pri zmýlení sa o jednu vekovú veková triedu, bola už presnosť na úrovni cca. 84%. Autori tvrdia, že aj napriek tomu, že použili jednoduchú architektúru, tak sa im podarilo dosiahnuť podobné výsledky ako v iných prácach, v ktorých boli použité zložitejšie siete. [10]

### Estimácia pohlavia a veku nefiltrovaných tvárí

Autori sa v článku zaoberajú rozpoznávaním atribútov tváre – konkrétne pohlavia a veku – z obrázkov, ktoré neboli zozbierané v laboratórnom prostredí. Použili databázu anotovaných tvarí zozbieraných pomocou telefónov, fotoaparátov alebo zo sociálnych sietí. Poukazujú na to, že väčšina podobných prác používa databázy, kde sú tváre frontalizované, a že obe úlohy boli v tejto práci náročnejšie na implementáciu ako zvyčajne. Ukazujú ako použiť techniku dropout-SVM pri implementácii rozpoznávania pohlavia a veku a zdôvodňujú to tým, že chceli predísť preučeniu.

Autori použili FG-NET, MORPH, Adience Benchmark databázy a ďalšie, na ktorých použitím SVM testovali rôzne techniky, vrátane dropoutu, PCA alebo inak filtrovaných obrázkov. Pri predikcii pohlavia bol najlepší dosiahnutý výsledok na úrovni 88,6 percenta. Pri odhadovaní veku použili rozdelenie do tried a dosiahli 45,1 percentnú úspešnosť v odhadovaní presnej vekovej triedy. V závere argumentujú, že trénovanie použitím dropoutu pri SVM dosiahli lepšie výsledky ako bez neho. [11]

### Odhadovanie pohlavia použitím Adaboostu

Prezentujú metódu založenú na AdaBoost a použitú na identifikovanie pohlavia človeka z čiernobieleho ( grayscale ) obrázka s malým rozlíšením. Trénovali nad FERET databázou s takmer 9000 obrázkami. Autori tvrdia, že ich metóda mala byť implementovaná do systému je schopného spracovať obrázkov. Cieľom bolo navrhnúť extrémne efektívny systém, ktorý bude ľahké implementovať a udržiavať. Tvrdia, že sa im poradilo dosiahnuť 80% úspešnosť pri použití len 10 pixelového obrázka. Pri 50 hovoria, že úspešnosť stúpla na 90% a pri 20x20 obrázku až 93%. Zároveň porovnávajú rýchlosť AdaBoostu oproti SVM klasifikátorom a tvrdia, že ich riešenie je 50-krát rýchlejšie. [12]

# Implementácia

V tejto kapitole si vysvetlíme, akým spôsobom sme pripravili a normalizovali dáta, aké databázy sme použili, a ako sme ich filtrovali. Rozoberieme si, ktoré architektúry sietí sú vhodné pre predikciou pohlavia a veku, ukážeme si spôsob ako ich pomocou Caffe frameworku naimplementovať. Zároveň si priblížime spôsob hľadania najlepšej modelu, finálneho testovania a integrácie najlepšieho modelu do grafickej aplikácie.

## Príprava dát

V dnešnej dobe nie je také dôležité akú metódu strojového učenie na riešenie problému zvolíme, ale skôr ide o to, aké máme dáta. Preto by mala byť trénovacia množina dostatočne veľká a mala by dobre reprezentovať populáciu tak, aby mal finálny model dobrú schopnosť generalizácie. (napríklad trénovanie klasifikátora pohlavia len s deťmi nie je vhodné).

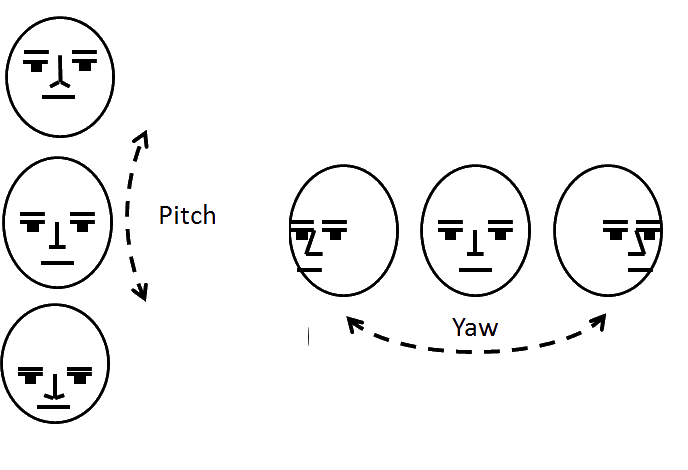
Druhým dôležitým faktorom je správna normalizácia dát. Chceme mať kvalitné nerozmazané obrázky s tvárami ľudí, na ktorých poriadne vidno oči, človek nemá veľmi vytočenú hlavu a v pozadí nie je zbytočný šum. Vyfiltrované tváre potom transformujeme tak, že oči budú na nami určených pozíciách.  Nakoniec vektorizované obrázky uložíme do LMDB databázy.

### Použité databázy, filtrácia obrázkov

V práci sme použili veľké množstvo rôznych databáz. Kombinujeme obrázky tvárí z laboratórneho prostredia s tvárami zo sociálnych sietí alebo filmových webov. Diverzifikácia zdrojov by mala pomôcť pri lepšej generalizácii trénovacieho algoritmu a sieť by mala byť schopná lepšie rozpoznávať neznáme tváre.

Pri prvotnom spracovávaní je dôležité tváre základným spôsobom anotovať. Pre naše dve úlohy, sme hľadali databázy, ktoré majú informácie o pohlaví, veku, pozícii očí alebo o natočení hlavy. Pohlavie a vek sú nutnosťou, ale ak chýbajú ostatné informácie, môžeme použiť treťostranové API schopné ich doplniť.

Všetky zozbierané dáta potom uložíme do spoločného (pre pohlavie a vek oddeleného ) \*.csv súboru, ktorý použijeme na následnú filtráciou nevhodných obrázkov. Vyhodíme všetky, ktoré neobsahujú pohlavie alebo vek. Keďže chceme frontalizované tváre, musíme odfiltrovať aj tie, ktoré sú natočené príliš vertikálne alebo horizontálne.(pitch, yaw). Ilustráciu môžeme vidieť na *obrázku 11.*



Obrázok 11 – filtrácia natočenia tvárí

Jednotlivé databázy filtrujeme rôznymi spôsobmi a v niektorých môžeme vyhodiť obrázky s natočením 25 stupňov, čo bola najčastejšie používaná hodnota v tejto práci, v iných povolíme nižšie alebo tolerujeme vyššie hodnoty.

Treba si dať pozor na osvetlenie. Chceme zabrániť tomu, aby sa sieť naučila rozpoznávať svetlo namiesto pohlavia a veku. Preto v databáze, ktorá bola robená v zlých svetelných podmienkach, môžeme filtrovať príliš svetlé a tmavé obrázky alebo ju v niektorých prípadoch zahodíme celú.

#### Klasifikátor pohlavia

Pre klasifikátor pohlavia sme použili nasledovné databázy. Detaily je možné pozrieť si v priložených odkazoch.

* ARFace [13] – vyrobená v laboratórnych podmienkach. Rôzne výrazy, okuliare, zakryté otvorené ústa
* AT&T – The Databases of Faces [14] – vyrobená laboratórnych podmienkach 10 obrázkov pre každý subjekt , rôzne výrazy tváre ( zavreté, otvorené oči )
* BioID [15] – komerčná databáza. Použitá len demo verzia
* CACD2000 [16] – databáza celebrít pôvodné určená na rozpoznávanie tvárí. Výhodou je, že herci sú fotení v rôznych obdobiach života
* CASIA-WEB [17] – veľká databáza obrázkov z reálneho života zozbieraná z webu
* CAS-PEAL [18] – čínska laboratórna databáza s výrazmi tváre a rôznymi natočeniami hláv subjektov
* FaceScrub [19] – ďalšia databáza celebrít odfotených v reálnych situáciách
* Morph [20] – komerčná databáza, použité len demo
* Multi-Pie [21] – veľká laboratórna databáza zozbieraná v štyroch fázach s 5 mesačnými pauzami

Nasledujúca tabuľka ukazuje pre každú databázu výsledný počet obrázkov a subjektov rozdelený po pohlaviach

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Databáza* | *Ženy subjekty* | *Muži subjekty* | *subjekty* | *ženy obrázky* | *muži obrázky* | *obrázky* |
| *ARface* | 60 | 76 | 136 | 1367 | 1690 | 3057 |
| *att* | 4 | 35 | 39 | 40 | 350 | 390 |
| *BioId* | 7 | 16 | 23 | 545 | 976 | 1521 |
| *CACD2000* | 773 | 836 | 1609 | 48824 | 58332 | 107156 |
| *CASIA-WEB* | 3864 | 5620 | 9484 | 140548 | 169708 | 310256 |
| *CAS-PEAL-R1* | 445 | 595 | 1040 | 4013 | 5960 | 9973 |
| *FaceScrub* | 265 | 265 | 530 | 31920 | 37768 | 69688 |
| *morph* | 117 | 810 | 927 | 136 | 1087 | 1223 |
| *Multi-Pie* | 102 | 235 | 337 | 19422 | 44190 | 63612 |
| Suma | 5637 | 8488 | 14125 | 246815 | 320061 | 566876 |

Tabuľka 1 – počet obrázkov a subjektov s pohlavím

#### Regresor veku

Databáz s údajmi o veku je podstatne menej ako s údajmi o  pohlaví a mnoho z nich nie je dostatočne kvalitných alebo vek neudávajú v presných hodnotách. Často sa objavujú dáta, ktoré uvádzajú vekové rozpätie, preto je potrebné z toho nejakým spôsobom vyrobiť presný vek. Väčšinou sme z tohto rozpätia urobili priemer. Ak napríklad informáciu k tvári obsahovala vek od 15-20, subjektu sme priradili vek 17 rokov. Priemer však nebol pravidlom a rozhodovali sme sa od prípadu k prípadu.

Druhý problém bolo malé zastúpenie starších vekových skupín. Keďže tréning regresnej siete potrebuje mať horné obmedzenie veku a z celkových takmer 5000 subjektov sme mali len málo takých, ktorí mali 60 a viac rokov, rozhodli sme maximum stanoviť na 60.

V tejto sme použili sme nasledovné databázy

* Adience benchmark[11] – databáza zozbieraná zo sociálnych sietí. Vek tu nie je presne vyplnený a zároveň rozpätia nie sú uniformné. Je to najväčšia, ktorú sme v práci použili.
* CAS-PEAL [18] – neobsahuje presný vek. Delená na tri skupiny : mladý , stredný, starý. Keďže sa väčšinou jedná o čínskych študentov a pracovníkov univerzity rozhodli sme sa im prideliť sme im : mladý 🡪 20, stredný 🡪 35, starý 🡪 55
* Morph [20] – obsahuje presný vek
* Multi-Pie [21] – obsahuje dátum narodenia. Keďže databáza bola vyrobená v roku 2000 a presný vek sa dá ľahko dopočítať
* FG-NET [22] – obsahuje presný vek. Jedna z najcitovanejších v podobných prácach. Špeciálne vyrobená pre rozpoznávanie veku. Obsahuje, ale pomerne málo obrázkov.
* FERET [23] – obsahuje presný vek, laboratórna databáza
* GROUPS – obsahuje približný vek, doplnili sme ju v neskorších fázach

|  |  |  |
| --- | --- | --- |
| *Databáza* | *Počet obrázkov* | *Počet subjektov* |
| Adience | 15163 | 2051 |
| CAS-PEAL-R1 | 21172 | 1039 |
| ColorFeret | 5114 | 990 |
| fg-net | 979 | 82 |
| morph | 1223 | 927 |
| Multi-Pie | 88700 | 337 |
| Suma | 132351 | 5426 |

Tabuľka 2 – rozdelenie obrázkov a subjektov s vekom

### Normalizácia obrázkov

Po vyselektovaní tvári je potrebné urobiť ich frontalizáciu, teda všetky obrázky by mali byť transformované tak, aby mali rovnaké rozlíšenie a pozície očí boli na rovnakej pozícii. Normalizácia prebieha v niekoľkých fázach.

1. **Vycentrovanie očí do rovnakých pozícií**

Cieľom je transformovať obrázok tak, aby pôvodné pozície očí a boli transformované na nové, nami definované pozície a . Napríklad, budeme chcieť, aby všetky normalizované tváre mali oči na pozíciách (40,40) (80,40). Maticu afinnej transformácie, ktorú použijeme vo OpenCV funkcii warpAffine, si musíme vypočítať.

1. **Vyrezanie obrázka tak, aby mal zvolené rozlíšenie**

Môžeme to urobiť to v kroku 1 napríklad pomocou OpenCV [24] funkcie warpAffine, kde si rovno vieme definovať veľkosť výstupnej bitmapy.

1. **Prevod do 8-bitového obrázka ( grayscale )**

Opäť môžeme použiť OpenCV na prevod farebného obrázka do 8-bitovej čiernobielej bitmapy

1. **Zmenšenie do požadovanej veľkosti**

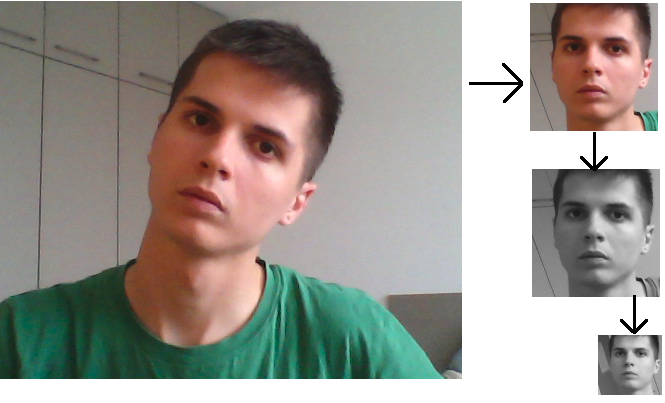
Je dobré v nechať si v prvej fáze obrázky väčšie napr. 128x128 a neskôr pri testovaní iných architektúr zmenšiť rozlíšenie na napr. 64x64.

1. **Môžeme použiť histogramovú ekvalizáciu**

Aj keď sa nám to v tejto práci ukázalo kontraproduktívne, niekedy má zmysel použiť ekvalizáciu na zvýraznenie kontrastu.

1. **Vektorizácia**

Po frontalizácii je potrebné transformovať z bitmapu do vektorizovanej formy, pretože Caffe nepracuje pracuje len s jednorozmernými *blob*-mi.



Obrázok 12 – príklad frontalizácie tváre

### Zväčšenie trénovacej množiny

Jedna z techník na zvyšovanie presnosti algoritmu je umelé zväčšovanie trénovacej množiny. Zároveň tým, že ten istý obrázok mierne pozmeníme a ukážeme ho sieti viackrát, predchádzame preučeniu. Normalizovanú tvár môžeme mierne rotovať, upraviť jas, naškálovať alebo ju horizontálne prevrátiť ( zrkadlový obraz ).

Druhý dôvod pre upravene trénovacej množiny je vybalansovanie populácie. Ak by sme napríklad mali príliš veľa mužov oproti ženám, musíme počet žien dorovnať.

#### Klasifikátor pohlavia

Pre úlohy klasifikácie pohlavia sme mali k dispozícii viac ako pol milióna obrázkov, preto sme sa rozhodli, že trénovaciu množinu nie je potrebné rozširovať.

#### Regresor veku

Zastúpenie jednotlivých vekových skupín bolo v použitých databázach nerovnomerné, konkrétne sme mali malý počet detí a ľudí nad 50 rokov. Preto bolo potrebné trénovacie dáta pre menej početné skupiny rozšíriť. Najprv sme zostrojili histogram, kde sme ľudí rozdelili do vekových skupín : 0-10, 10-20, ... 60-70.

Kde je počet kópií každej vekovej skupiny , pričom sme si nastavili maximálny počet zväčšení na 10 . Tiež chceme mať jednu horizontálne prevrátenú tvár. ( preto 2 ). Na *tabuľke 3* môžeme vidieť histogram rozdelenia vekových skupín pred a po zväčšení.

Tabuľka 3 – histogramy trénovacej množiny vek pred a po zväčšení

#### Prevod do LMDB databázy

Fáza, v ktorej musíme pripraviť potrebné \*.mdb (LDMB databázový súbor) súbory, na ktoré sa budeme odkazovať v trénovacom train\_test.prototxt súbore. Pomocou API schopného pracovať s LMDB( napríklad v pythone ) si pripravíme dva \*.mdb súbory. V jednom budú uložené vektorizované obrázky vo formáte akému Caffe rozumie a v druhom budú ich anotácie (ground truth), teda pohlavie a vek pre každý subjekt.

Anotácie priraďujeme pre pohlavie tak, že všetkým mužom priradíme 0 a ženám 1. Pretože pri regresii používame sigmoid, pre odhadovanie veku sme každú hodnotu vydelili maximálnym vekom (teda 60) tak, aby sme mali hodnoty na intervale <0,1>.

Ak používame normalizáciu centrovaním do (0,0), môžeme použiť nástroj priamo z Caffe frameworku *tools/compute\_image\_mean,* ktorému zadáme cestu k adresárom s \*.mdb súbormi a ten nám vyrobí potrebný \*.binaryproto a \*.npy *mean* ( priemerným obrázok ) súbor.

## Trénovanie

V tejto fáze si vysvetlíme ako pripraviť, ako púšťať samotné trénovanie, čo je cross validácia a ako sme ju použili. Ukážeme si ako nájsť najlepší model alebo ako vizualizovať architektúru siete a výsledné konvolučné filtre.

Kvôli povahe úlohy sme pre klasifikátor pohlavia sme použili ako chybovú funkciu softmax, pre regresor veku sigmoid vrstvu nasledovanú euclid vrstvou.

### Príprava trénovania, architektúry sietí

Potom ako pripravíme dáta vo formáte, ktorému Caffe rozumie, musíme vyrobiť trénovacie súbory ( solver.txt, train\_test.prototxt, predict.prototxt ). Aby sme zabezpečili jednoduché hľadanie najlepšej architektúry, potrebovali sme generovať \*.prototxt súbory z predpripravených šablón. Chceli sme testovať aký vplyv majú na výslednú chybu parametre ako : veľkosť kroku, veľkosť kernelu, počet výstupov konvolučnej vrstvy, dropout, počet iterácií, rýchlosť učenia atď. Ak je všetko pripravené, spustíme nad generovanými \*.prototxt súbormi program *caffe train,* ktorý vykoná trénovanie.

### K-fold cross validácia

Za účelom vylepšenia presnosti algoritmu a zabráneniu preučenia sa pri metódach strojového učenia používa tzv. *cross validácia*. Trénovaciu množinu ( testovaciu máme odloženú ) si rozdelíme na dve časti – estimačnú, na ktorej prebieha samotné trénovanie a validačnú, na ktorej vyhodnocujeme trénovaciu chybu. Upravovanie váh a spätná propagácia teda prebieha len na základe dát z estimačnej množiny.

Pri *k-fold cross validácii* si najprv celú trénovaciu množinu náhodne usporiadame a rozdelíme na *k* častí. V prvom kroku zvolíme časti (*1,2 ... k-1)* za estimačnú a *k* za validačnú množinu. Urobíme to druhýkrát a estimačnou budú teraz časti (1, 2...*k-2, k)*, validačnou *(k-1).* Dostaneme 7 rôznych kombinácii, v ktorých máme rôzne validačné a trochu inú estimačné množiny. (vždy chýba jedna časť) . Trénovanie budeme musieť pustiť na všetkých kombináciách ( foldoch.), čím dostaneme *k* rôznych modelov, ktorých výsledky môžeme spriemerovať a tak lepšie odhadnúť chybu danej architektúry. V práci sme pre obe úlohy použili 4-fold cross validáciu.

Caffe podporuje cross validáciu definovaním TEST ( validácia ) a TRAIN ( estimácia ) fáz. Budeme mať dve vstupné vrstvy, ktoré budú ukazovať na iné \*.mdb súbory. Jedna bude aktivovaná len vo váze TEST(validácia) a ukazovať bude na \*.mdb s validačnou množinou, druhá vo fáze TRAIN a ukazovať bude na \*.mdb s estimačnou množinou. Názov TEST si netreba mýliť s finálnym testovaním a testovacou množinou.

#### Hľadanie najlepšieho modelu

Keďže počas trénovania nemusíme dosiahnuť najnižšiu chybu v poslednej iterácii, potrebovali by sme si ukladať priebežné modely a ich validačné chyby. Caffe to našťastie podporuje a pri definovaní solver.txt súboru si môžeme zvoliť frekvenciu ukladania snapshotov( \*.caffemodel súbory ) , na ktorých budeme hľadať najlepší model.

Počas trénovania si štandardný výstup budeme ukladať do logovacie textového súboru, v ktorom potom hľadáme reťazec – „*Test net output #0: loss = nejaké číslo* “ – na základe ktorého vyberieme snapshot s najnižšou validačnou chybou.

#### Vizualizácia siete, chyby, natrénovaných filtrov

Caffe ponúka niekoľko možností ako výsledky trénovania vizualizovať. Kombináciou Numpy knižnice a Caffe API pre python môžeme načítať natrénovaný \*.caffemodel súbor, podhodiť mu nejaký obrázok a z jednotlivých vrstiev extrahovať konvolučné filtre a feature mapy. Zavolaním nasledovnej funkcie vieme načítať model.

*net* = *caffe.Net(‘predict.txt’,’model.caffemodel’, caffe.TEST)*

Pomocou *net.params* dostaneme potrebné dáta na vizualizáciu.

*net.params[layer][0].data* 🡪filtre

*net.blobs[layer].data[0,:]* 🡪feature mapy

Tieto jednorozmerné polia môžeme ďalej pretransformovať na bitmapy a uložiť. Podrobný návod je možné nájsť v návodoch na oficiálnej stránke Caffe.

Niekedy je vhodné vidieť architektúru siete na obrázku, čo vieme docieliť pomocou funkcie *caffe.draw.draw\_net\_to\_file .*

Vizualizácia estimačnej a validačnej chyby je zložitejšia, pretože Caffe na to priamo neponúka nástroje. Ak si však zapneme debugovacie výstupy, môžeme tieto údaje vyčítať z logovacieho súboru a  zostaviť graf vývoja chýb.

## Testovanie

Ako sme spomínali, pôvodné dáta si rozdelíme na testovaciu a trénovaciu množinu. Nad trénovacou, ktorú sme delili na validačnú a estimačnú časť, prebieha trénovanie zatiaľ, čo testovaciu sme máme odloženú. Po skončení trénovania a nájdení najlepšieho modelu podľa validačnej chyby, nám ostáva otestovať schopnosť generalizácie, čo urobíme práve testovaním nad testovacou množinou. Dáta v nej si musíme normalizovať presne tak, ako sme to urobili v 2.1.2.

Zo všetkých sietí, ktoré sme natrénovali hľadáme model s najnižšou testovacou chybou, ktorý potom môžeme použiť v GUI aplikácii. Keďže sme trénovali K-fold cross validáciou, najlepší model vyberieme podľa najnižšej priemernej testovacej chyby, ktorú počítame zo všetkých *k* častí.

### Klasifikátor pohlavia

Keďže sa jednalo o klasifikačnú úlohu vyhodnocovali sme testovaciu chyby, vypočítali sme confustion maticu, FPR, FNR, TPR a TNR.

#### Testovacia chyba ( accuracy )

Testovacie chyba je základné vyhodnotenie presnosti nášho algoritmu. Prejdeme cez všetky testovacie dáta a  určíme koľko percent z nich nám sieť odhadla správne. Podobne ako pri vizualizácii, aj tu môžeme použiť funkciu *caffe.Net* na načítanie siete a následne *caffe.Net.forward* na vypočítanie predikcie. To nám vráti dvojrozmerné pole ( odpoveď na softmax s 2 neurónmi ), nad ktorým zavoláme *argmax*, čím dostaneme predikované pohlavie.

#### Confusion matica

Confusion matica (známa aj ako *matica chyby )* je špeciálny typ dvojdimenzionálnej kontingenčnej tabuľky, ktorá sa používa v strojovom učení na podrobnú vizualizáciu chyby pri klasifikácii. Stĺpce matice reprezentujú inštancie predikcií triedy, riadky reprezentujú očakávané hodnoty. Pre klasifikáciu pohlavia, teda dvoch tried, bude vyzerať 2x2 matica ***C*** nasledovne.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Predikované hodnoty** | |
| **Trieda 1 (muž)** | **Trieda 2 (žena)** |
| **Skutočné hodnoty** | **Trieda 1 (muž)** | **Počet správne klasifikovaných mužov**  **(true positive)** | **Počet mužov klasifikovaných ako ženy (false negative)** | **Počet mužov**  **(condition positive)** |
| **Trieda 2 (žena)** | **Počet žien klasifikovaných ako muži**  **(false positive)** | **Počet správne klasifikovaných žien(true negative)** | **Počet žien**  **(condition negative)** |
|  | | **Počet predikovaných mužov** | **Počet predikovaných žien** |

Tabuľka 4 – confusion matica

Maticu môžeme znormalizovať tak, že každú hodnotu C vydelíme súčtom príslušného riadka.

Výslednú chybu dostanem spočítaním všetkých prvkov mimo diagonály a naopak na presnosť ( accuracy ) nám stačí spočítať prvky na diagonále

#### TPR,TNR ,FPR,FNR

Pri klasifikačnej úlohe nás zaujímajú nasledujúce údaje : senzitivita, špecificita**,** falošná negativita, falošná pozitivita. Podľa nich môžeme určiť, či je klasifikačný algoritmus je dobre vybalansovaný a chceme mať, čo najnižšie FNR, FPR, a aby ich hodnoty boli podobné.

Tieto údaje môžeme jednoducho zobrať z normalizovanej confusion matice. Podľa tabuľky 4 sme si ako pozitívnu triedu stanovili muža, preto sa budeme pozerať na nasledujúce údaje z tohto pohľadu.

**Senzitivita (TPR - True positive rate) –** pravdepodobnosť, že pozitívny vstup bol správne klasifikované ako pozitívny ( muž bol klasifikovaný ako muž )

**Špecificita (TPR - True negative rate) –** pravdepodobnosť, že negatívny vstup bol správne klasifikovaný ako negatívny ( žena bola klasifikovaná ako žena )

**Falošná negativita (FNR – False negative rate) –** pravdepodobnosť, že pozitívny vstup bol nesprávne klasifikovaný ako negatívny ( muž bol klasifikovaný ako žena )

**Falošná pozitivita (FPR – False positive rate) –** pravdepodobnosť, že negatívny vstup bol nesprávne klasifikovaný ako pozitívny (žena bola klasifikovaná ako muž )

### Regresor veku

Keďže sme pri rozpoznávaní veku použili regresiu, nemohli sme využiť bežné štatistické metóde ako *confusion* *maticu* alebo jednoduché meranie celkovej chyby. Preto sme pre natrénovaný model zisťovali aké percento testovacích dát sa nám podarilo správne predikovať s určitou tolerovali chyby. Napríklad, pri tolerancii 5 rokov považujeme za úspešnú predikciu prípad, keď sieť predikovala 17 ročnému subjektu 15 rokov. V práci sme merali tolerancie : 2,5,10,15,20,25,30,35,40 rokov.

Obrázok 13 – príklad vyhodnotenia testovacej chyby veku

Kvôli nerovnomernému zastúpeniu vekových kategórií nie je absolútne vyhodnotenie testovacej chyby dostačujúce na nájdenie modelu s najlepšou schopnosťou generalizácie. Môže sa stať, že algoritmus dobre odhaduje mladých ľudí, ale pri nedostačujúcom množstvo starších ľudí v testovacej množine a vysokej chybe pri odhadovaní ich veku, by sa veľmi absolútna chyba nezmenila. Preto potrebujeme merať chybu vekových kategórií a v práci sme sa rozhodli testovať intervaly : 0-10, 10-20, 20-30, 30-40, 50-60. Z jednotlivých chýb vekových kategórií potom urobíme priemer, čím dostaneme vybalansovanú testovaciu chybu. Príklad môžeme vidieť v nasledujúcej tabuľke.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tolerancia** | **0-10** | **10-20** | **20-30** | **30-40** | **30-50** | **50-60** | **Priemerná presnosť cez všetky k foldy** | | **Balansovaná presnosť cez všetky k foldy** |
| 2 | 36.63 | 33.796 | 32.602 | 17.014 | 15.838 | 14.885 | | 28.307 | 25.127 |
| 5 | 79.387 | 71.945 | 66.294 | 44.94 | 38.351 | 33.627 | | 60.595 | 55.757 |
| 10 | 89.276 | 94.708 | 89.156 | 86.256 | 80.955 | 61.705 | | 87.708 | 83.676 |
| 15 | 93.872 | 98.12 | 96.046 | 97.314 | 96.859 | 81.867 | | 95.862 | 94.013 |
| 20 | 98.05 | 99.081 | 98.803 | 99.563 | 98.822 | 92.084 | | 98.605 | 97.734 |
| 25 | 99.582 | 99.653 | 99.765 | 99.979 | 99.411 | 96.143 | | 99.563 | 99.089 |
| 30 | 99.721 | 99.877 | 100 | 99.979 | 99.738 | 97.835 | | 99.836 | 99.525 |
| 35 | 99.721 | 100 | 100 | 100 | 99.869 | 99.526 | | 99.96 | 99.853 |
| 40 | 99.721 | 100 | 100 | 100 | 100 | 99.797 | | 99.982 | 99.92 |

Tabuľka 5 – príklad vybalansovanej chyby podľa vekových kategórií

## Integrácia natrénovaných modelov do aplikácie

Súčasťou našej práce bolo pripravenie demo aplikácie, do ktorej sme integrovali natrénované modely pre rozpoznávanie veku a pohlavia. Program, ktorý sme implementovali v C++ pomocou OpenCV a Caffe frameworku, sníma obraz z web kamery, hľadá ľudskú tvár a v reálnom čase robí klasifikáciu pohlavia a odhadovanie veku pomocou natrénovaných modelov.

### Nájdenie a frontalizácia tváre pomocou OpenCV

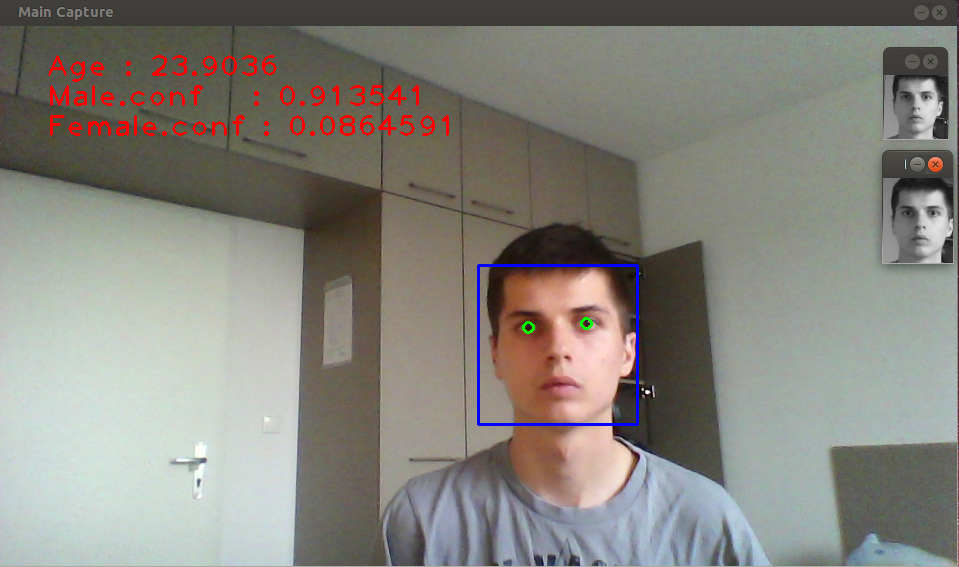
Podobne ako pri trénovaní a testovaní, aj tu sme potrebovali obrázok získaný web kamerou najprv normalizovať. OpenCV ponúka detektor ľudských tvári implementovaný v triede *CascadeClassifier*. Do konštruktora zadáme cestu k natrénovanému modelu ( \*.xml) detektora tvári , konkrétne *haarcascade\_frontalface\_alt.xml.* Zároveň môžeme pomocou tej iste triedy použiť aj detektor očí (*haarcascade\_eye.xml ),* ktorý je však pomalý a nepresný. Preto sme sa rozhodli na tento účel použiť [25].

Po nájdení tváre a očí urobíme tú istú normalizáciu tváre ako v 2.1.2. Ak by sme siete veku a pohlavia trénovali s rôznymi veľkosťami vstupných obrázkov, museli by sme urobiť dve normalizácie. Jednu pre vek, druhú pre pohlavie.

### Implementácia klasifikátora, regresora pomocou Caffe API pre C++

Podobne ako pre python, Caffe ponúka API s tými istými triedami a funkcionalitou pre C++. Súbory modelu siete ( \*.caffemodel, predict.prototxt, images\_mean.binaryproto) načítavame pomocou C++ API tak, ako pri testovaní. Na predikciu použijeme funkciu caffe::Net::Forward.

Na obrázku 14. môžeme vidieť screenshot z našej GUI aplikácie. Modrý štvorec predstavuje detegovanú tvár. V ľavom hornom rohu vidíme zobrazenie odhadovaného veku a pravdepodobnosti, že snímaná osoba je muž alebo žena. V pravom rohu je obrázok normalizovanej tváre ( z modrého štvorca ) , ktorú používame ako argument funkcie caffe::NET::Forward.



Obrázok 14 – screenshot z GUI aplikácie

# Experiment a výsledky

V tejto kapitole si ukážeme, akým spôsobom sme hľadali najlepšie architektúry sietí pre rozpoznávanie veku a pohlavia. Pozrieme sa na to, ktoré metódy fungovali najlepšie, podrobne si odprezentujeme výsledky a vysvetlíme si ako by sme potenciálne mohli rozpoznávanie vylepšiť.

### Klasifikátor pohlavia

V článku [7] sa môžeme dočítať, že miesto toho, aby sme navrhovali architektúru konvolučnej siete od nuly, mali by sme zobrať nejakú existujúcu, ktorá bola použitá na podobný problém a tú iteratívne vylepšovať.

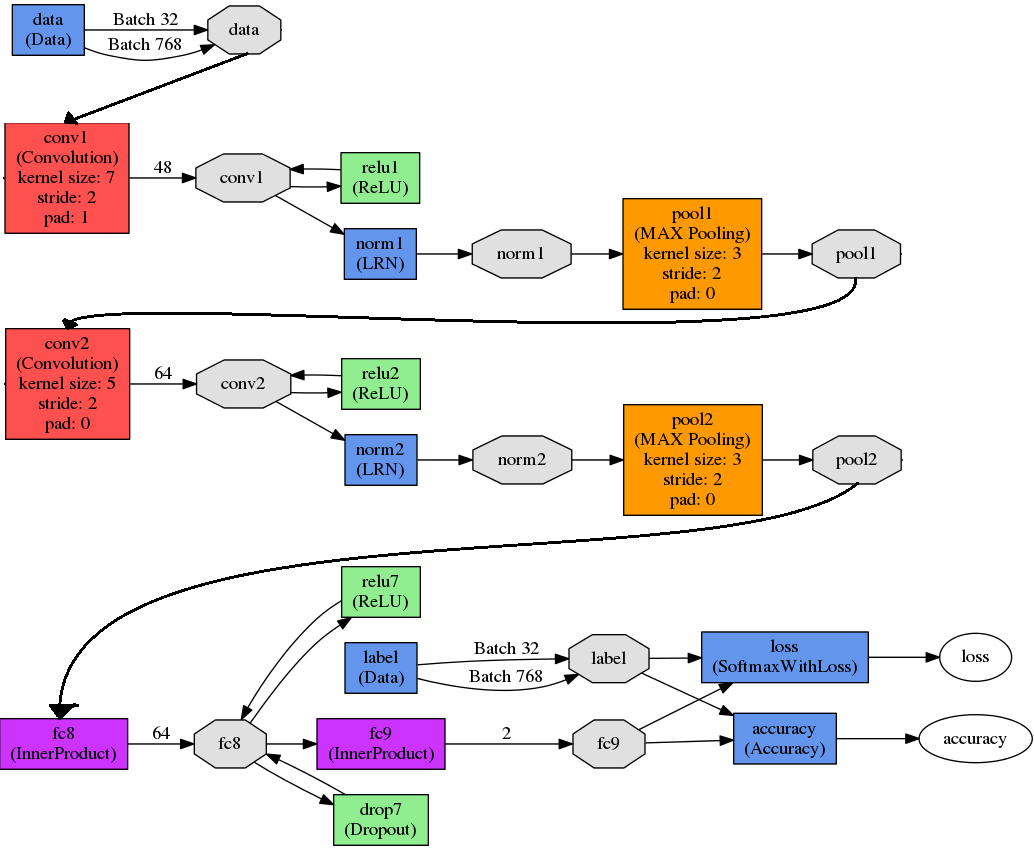
Základnú kostru siete pre rozpoznávanie pohlavia sme sa snažili napodobniť podľa [10], ale bola pre nás príliš komplexná, pretože autori použili RGB obrázky s rozlíšením 227x277 a veľké množstvo neurónov, čo zvyšovalo výpočtovú zložitosť. My sme trénovali sieť na 8-bitových čiernobielych obrázkoch, a aby sme zabránili preučeniu, museli sme komplexitu siete znížiť, teda ubrať počet neurónov. Zároveň sme použili omnoho menšie obrázky (70x85) ako v [10] a zistili sme, že rozlíšenie nemá veľký vplyv na výslednú chybu. Osvedčilo sa nám používať obrázky veľkosti 70x85, keďže chyba nebola oveľa vyššia ako pri 140x170. ( rozdiel bol okolo 0.5% )

V práci sme testovali dva základné typy sietí :

1. s dvoma konvolučnými a dvoma plne prepojenými vrstvami
2. s troma konvolučnými a troma plne prepojenými.

Na *obrázku 15* je zobrazená východisková architektúra konvolučnej siete pre úlohu rozpoznávania pohlavia. Na vstupe je vektorizovaný obrázok veľkosti 70x85. S každou konvolučnou vrstvou sa viaže následná normalizácia pomocou ReLU a LRN vrstiev a zredukovanie veľkosti feature mapy pomocou pooling vrstvy. Pokračujeme tým, že na druhú konvolučnú pripojíme plne prepojenú vrstvu a experimentálne sme zistili, že sieť dosahuje dobré výsledky, ak počet neurónov v nej je ekvivalentný počtu *feature máp* v poslednej konvolučnej vrstve.

Keďže sa jedná o klasifikačnú úlohu, po plne prepojených vrstvách použijeme ako aktivačnú funkciu *softmax* nasledovanú *accuracy*.



Obrázok 15 – východisková architektúra siete pre rozpoznávanie pohlavia

#### Architektúra s najnižšou testovacou chybou

Pri hľadaní siete s najnižšou testovacou chybou sme skúšali :

* Meniť počty filtrov v konvolučných vrstvách
* Pridať dropout aj konvolučným vrstvám
* Rôzne hodnoty dropout koeficientu
* Symetrickú architektúru ( počty neurónov na všetkých vrstvách sú rovnaké )
* Rôzne hodnoty stride, grupy alebo *zero-paddingu* na konvolučných vrstvách
* Pridanie tretej konvolučnej vrstvy

Keďže trénovanie trvá pomerne dlhú dobu a najlepšiu architektúru sme hľadali pomocou k-fold cross validácie, nastavili sme maximálny počet iterácii na malú hodnotu – 100 000. Po natrénovaní všetkých sietí sme tréning východiskového modelu zopakovali s 500 000 iteráciami.

Výsledky pre dve konvolučné vrstvy môžeme vidieť v *tabuľke 6*, v ktorej (n, k, s, p, g) pri konvolučných vrstvách znamená (viď. 1.2.4.1) :

1. počet výstupných feature máp
2. veľkosť kernelu
3. veľkosť kroku ( stride )
4. zero-padding
5. grupovanie

Pre plne prepojené vrstvy v tabuľke udávame len počet ich neurónov, pre dropout vrstvu jej koeficient a v posledných dvoch stĺpcoch priemernú rýchlosť a presnosť vypočítanú cez všetky *k* modely *k-fold cross validácie.* ( v našom prípade 4-fold )

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Popis architektúry** | **Conv1  (n, k, s, p, g )** | **Drop1** | **Conv2 (n, k, s, p, g )** | **Drop2** | **FC3** | **Drop3** | **Priemerná rýchlosť** | **Priemerná presnosť** |
| Východiskový model | 48,7,2,1,1 |  | 64,5,2,0,2 |  | 64 | 0.2 | 1.09041 | 0.97075 |
| **Group+stride conv2 = 1** | **48,7,2,1,1** |  | **64,5,1,0,1** |  | **64** | **0.2** | **1.24398** | **0.97475** |
| Viac neur. na plne prep. | 48,7,2,1,1 |  | 64,5,2,0,2 |  | 96 | 0.2 | 1.09636 | 0.97012 |
| **Stride na conv2 = 1** | **48,7,2,1,1** |  | **64,5,1,0,2** |  | **64** | **0.2** | **1.37975** | **0.97458** |
| Jednoduchšia sieť | 32,7,2,1,1 |  | 48,5,2,0,2 |  | 64 | 0.2 | 0.98374 | 0.96883 |
| Vyšší dropout na plne prepojenej | 48,7,2,1,1 |  | 64,5,2,0,2 |  | 64 | 0.5 | 1.09464 | 0.96903 |
| Padding conv1 = 0 | 48,7,2,0,1 |  | 64,5,2,0,2 |  | 64 | 0.2 | 1.05135 | 0.97088 |
| Symetrická architektúra | 64,7,2,1,1 |  | 64,5,2,0,2 |  | 64 | 0.2 | 1.46319 | 0.97137 |
| Dropout na conv1, conv2 | 48,7,2,1,1 | 0.1 | 64,5,2,0,2 | 0.2 | 64 | 0.2 | 1.08958 | 0.97080 |
| Dropout na conv1, conv2, vyšší na fc3 | 48,7,2,1,1 | 0.1 | 64,5,2,0,2 | 0.2 | 64 | 0.5 | 1.09653 | 0.96893 |

Tabuľka 6 – výsledky sietí s dvoma konvolučnými vrstvami pre klasifikátor pohlavia

Z výsledkov môžeme dedukovať tieto fakty

* Nastavenie veľkosti kroku ( stride ) na 1 v druhej konvolučnej vrstve znižuje chybu, ale zvyšuje výpočtovú náročnosť, pretože sa zväčšuje feature mapa. Zníženie grupovanie na tej iste vrstve mierne zníži výpočtovú náročnosť a chybu.
* Zvýšenie počtu neurónov na plne prepojených vrstvách chybu nezlepšuje
* Vyšší dropout na plne prepojene vrstve mierne zvyšuje chybu
* Symetrická architektúra je pomalá a menej presná ako východisková
* Dropout na konvolučných vrstvách chybu neznižuje
* Zero-padding nemá takmer žiadny účinok

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Popis architektúry** | **Conv1  (n, k, s, p, g )** | **Conv2 (n, k, s, p, g )** | **Conv3 (n, k, s, p, g )** | **FC4** | **Drop4** | **FC5** | **Drop5** | **Priemerná rýchlosť** | **Priemerná presnosť** |
| Jednoduchšia trojvrstvová sieť | 32,7,2,0,1 | 48,5,1,0,2 | 64,3,1,0,2 | 64 | 0.2 | 64 | 0.3 | 4.36668 | 0.97383 |
| Trojvrstvová sieť | 48,7,2,0,1 | 64,5,1,0,2 | 64,3,1,0,2 | 64 | 0.2 | 64 | 0.3 | 4.74966 | 0.97498 |

Tabuľka 7 – výsledky sietí s troma konvolučnými vrstvami pre klasifikátor pohlavia

V *tabuľke 7* môžeme vidieť výsledky potom ako sme pridali jednu konvolučnú vrstvu a jednu plne prepojenú. Keďže sme opäť použili v druhej konvolučnej vrstve stride veľkosti 1 vidíme, že chyba nám takmer vôbec neklesla , ale priemerná rýchlosť predikcie sa takmer štvornásobne zhoršila.

#### Podrobná analýza výsledkov

Tréning východiskového modelu ( prvý v tabuľke 6 ) sme si zopakovali s 500 000 iteráciami, aby sme zistili ako sa zlepší výsledná chyba. V nasledujúcej tabuľke máme uvedené výsledky pre všetky 4 foldy z *k-cross validácie.* Vidíme, že päťkrát dlhšie trénovanie malo len minimálny dopad na finálny výsledok.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fold** | **Presnosť** | **Balansovaná presnosť** | **FPR** | **FNR** | **Rýchlosť** |
| fold 0 | 97.430% | 97.280% | 1.964% | 3.476% | 1.0564555 |
| fold 1 | 97.410% | 97.267% | 2.026% | 3.440% | 1.0584501 |
| fold 2 | 97.610% | 97.507% | 1.957% | 3.030% | 1.0490883 |
| fold 3 | 97.560% | 97.400% | 1.795% | 3.404% | 1.0479117 |
| **priemer** | **97.500%** | **97.364%** | **1.935%** | **3.337%** | **1.0529764** |

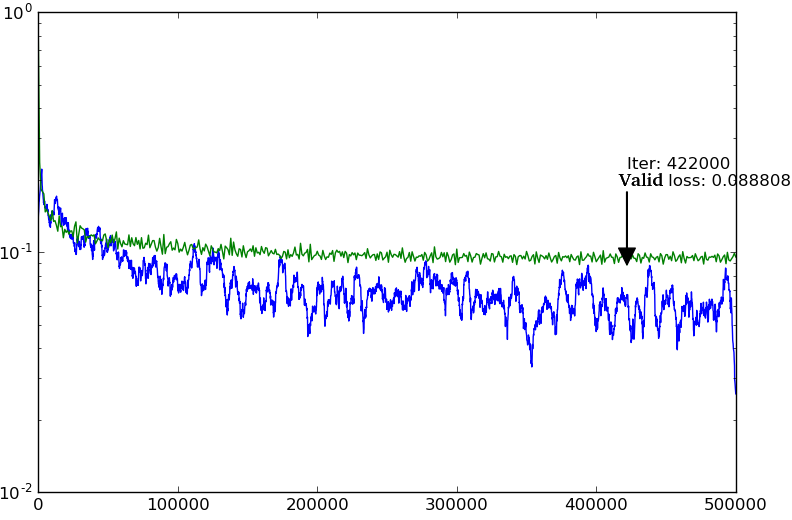
Tabuľka 8 – vypočítané priemerné hodnoty chýb pre východiskový model

Hodnoty FPR, FNR ale aj absolútny a balansovanú presnosť vieme dostať z *confusion matice.* Príklad confusion matice pre fold 0 je na *tabuľke 9*. Vo vnútorných bunkách sú naľavo sú uvedené absolútne hodnoty, napravo normalizované. Môžeme si všimnúť, že hodnoty FPR a FNR môžeme dostať rovno z tabuľky.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Predikované hodnoty** | | | |
| **muži** | | **ženy** | |
| Skutočné  hodnoty | **muži** | 41285 | 0.980362 | 827 | 0.019638 | **42112** | **1** |
| **ženy** | 974 | 0.034757 | 27049 | 0.965243 | **28023** | **1** |
|  | | **42259** | **0.602538** | **27876** | **0.397462** |

Tabuľka 9 – confusion matica pre fold 0 pre východiskového modelu

Na *obrázku 16* môžeme vidieť vývoj estimačnej(modrá) a validačnej(zelená) chyby počas trénovania s 500 000 iteráciami. Prudký pokles chyby nastal počas prvých 100 000 iterácií, preto sme tento počet vybrali pre hľadanie najlepšieho modelu.



Obrázok 16 – vývoj estimačnej a validačnej chyby pre klasifikátor pohlavia

*Obrázok 17* obsahuje subjekty, ktoré boli označené v testovacej množine ako muži, ale sieť ich vyhodnotila ako ženy. ( falošná negativita). V prvom rade sú ľudia, ktorých sieť vyhodnotila s viac ako 95 % istotou, že sa jedná ženy. Zistili sme, že tieto obrázky boli zlým spôsobom anotované v trénovacej databáze a naša sieť v podstate správne zistila, že sa nejedná o mužov. Dolný rad obsahuje subjekty, kde si sieť nebola istá a ide sa o nekvalitné obrázky alebo o ľudí, ktorí majú ženské črty.

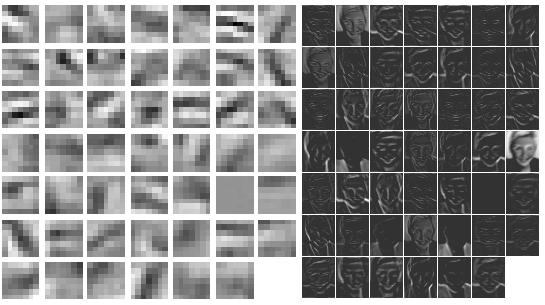


Obrázok 17 – falošné negatíva, anotovaní muži klasifikovaní ako ženy

*Obrázok 18* obsahuje falošné pozitíva, teda anotované ženy boli „nesprávne“ klasifikované ako muži. Podobne ako na *obrázku 17*, v prvom rade ľudia, pri ktorých si sieť bola takmer istá, že sa jedná o mužov, ale subjekty mali zle vyplnené pohlavie. ( Mel Gibson bol napríklad označený za Khloe Kardashian. ) Tiež si môžeme všimnúť, že sa tam nachádzajú obrázky s dvoma tvárami, teda sieť pravdepodobne klasifikuje nesprávnu osobu alebo oboch. Druhý rad znova obsahuje nekvalitné obrázky alebo ženy, ktoré skutočne majú mužské črty.



Obrázok 18 – falošné pozitíva, anotované ženy klasifikované ako muži



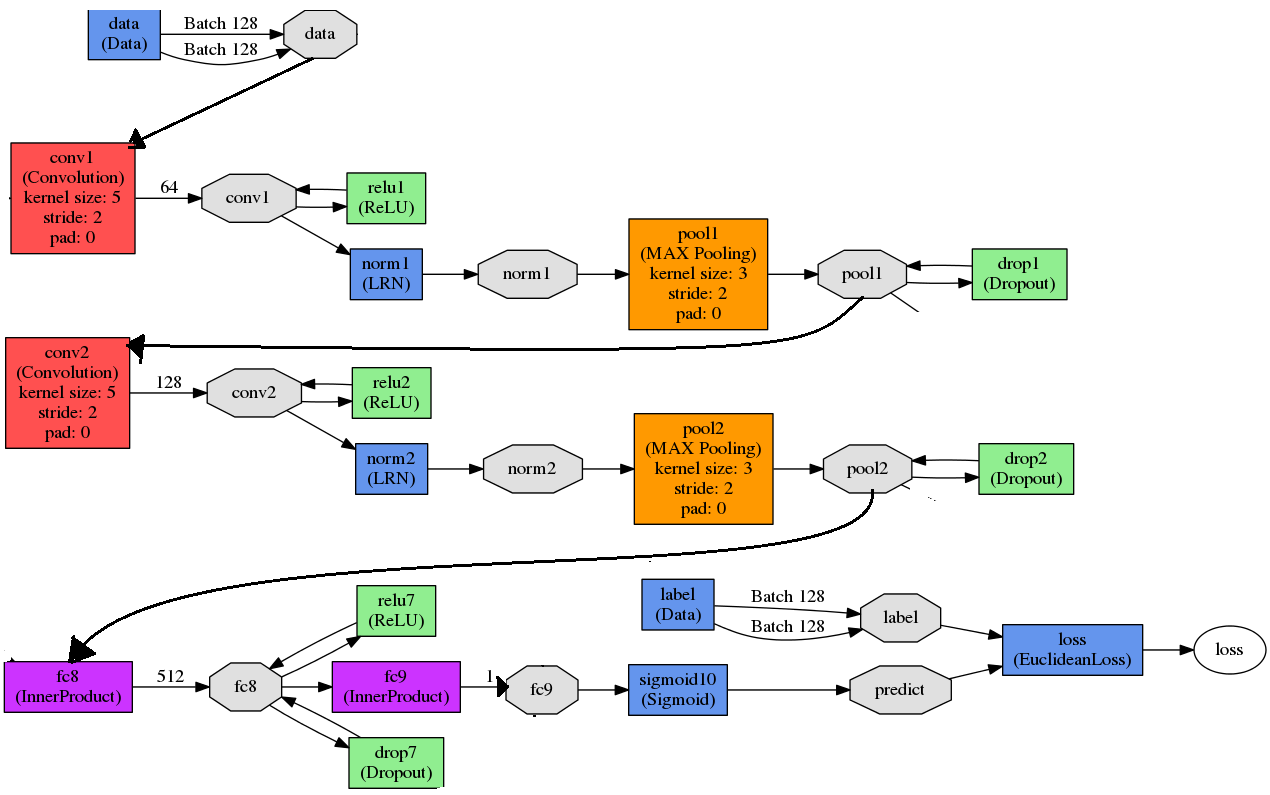
Obrázok 19 – vizualizácia konvolučných filtrov a feature máp 1. konvolučnej vrstvy pre klasifikátor pohlavia

#### Návrh na zlepšenie

Potenciálne by sme mohli vylepšiť trénovaciu množinu a zároveň výslednú chybu tak, že s prvou natrénovanou sieťou by sme vyhodili všetky falošné pozitíva a falošné negatíva, pri ktorých pravdepodobnosť predikcie pohlavia presahuje nejakú vysokú hodnotu ( napríklad. 85% ). Týmto spôsobom by sme z trénovacej množiny vyfiltrovali dáta, ktoré boli zle anotované a sieť pretrénovali nad vyčistenými dátami.

### Regresor veku

Pri rozpoznávaní veku sme sa snažili vylepšiť architektúru siete, ktorú sme používali na klasifikáciu pohlavia. Spočiatku sme úlohu riešili tak, že sme ľudí rozdelili do vekových skupín po 10 rokov a snažili sme sa použiť takmer totožnou architektúru siete ako pre pohlavie, ale validačná dokonca ani estimačná chyba počas tréningu vôbec neklesala. ( dokonca stúpala ) Preto sme sa skúsili na problém pozrieť ako na regresiu a rozhodli sme sa natrénovať sieť schopnú odhadovať presný vek, teda *softmax* sme nahradili euklidovskou chybovou funkciou.

Východisková architektúra v tomto prípade bude obsahovať dropout na konvolučných vrstvách, veľkosti kernelov a krokov budú rovnaké, a pretože pre regresiu potrebujeme vyššiu komplexitu siete , zvýšili sme počet neurónov v plne prepojenej vrstve.

Obrázok 20 – východisková architektúra siete pre rozpoznávanie veku

#### Architektúra s najnižšou testovacou chybou

Podobne ako pri rozpoznávaní pohlavia aj tu sme púšťali kratšie trénovania ( 100 000 iterácií ) na determinovanie najlepšej architektúry. Treba dodať, že výsledky v *tabuľke 10* sú natrénované bez GROUPS databázy, ktorú sme neskôr doplnili a preto sa budú chyby v ďalších meraniach mierne líšiť ( chyba bude vyššia ) . Skúšali sme nasledovné :

* Vplyv dropoutu na konvolučné vrstvy
* Zmeniť veľkosť kernelu
* Meniť počty neurónov a feature máp
* Meniť hodnotu dropoutu
* Nepoužívať dropout na konvolučných vrstvách

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | **Tolerancie v rokoch** | | | | | |
| **Popis** | **conv1** | **drop1** | **conv12** | **drop2** | **fc3** | **drop3** | **2** | **5** | **10** | **15** | **20** | **25** |
| Východisková sieť | 64,5,2,0,1 | 0.1 | 128,5,2,0,1 | 0.2 | 512 | 0.5 | 29.03 | 61.48 | 88.29 | 96.09 | 98.68 | 99.64 |
| Bez dropoutu na conv | 64,5,2,0,1 |  | 128,5,2,0,1 |  | 512 | 0.5 | 28.22 | 61.19 | 88.38 | 96.20 | 98.66 | 99.57 |
| **Menej neurónov na plne prepojenej** | **64,5,2,0,1** |  | **128,5,2,0,1** |  | **256** | **0.5** | **32.91** | **63.46** | **88.51** | **96.21** | **98.50** | **99.46** |
| Viac feature máp | 128,5,2,0,1 | 0.1 | 256,5,2,0,1 | 0.2 | 512 | 0.5 | 28.91 | 62.71 | 89.55 | 96.75 | 98.98 | 99.72 |
| Jednoduchšia sieť | 48,5,2,0,1 | 0.1 | 96,5,2,0,1 | 0.2 | 384 | 0.5 | 29.46 | 62.23 | 88.31 | 96.15 | 98.77 | 99.71 |
| Bez dropoutu na conv + menší na plne prepojenej | 64,5,2,0,1 |  | 128,5,2,0,1 |  | 512 | 0.2 | 29.26 | 62.30 | 88.69 | 96.23 | 98.61 | 99.53 |
| 128 neurónov na FC3 | 64,5,2,0,1 |  | 128,5,2,0,1 |  |  | 0.5 | 27.35 | 60.68 | 88.56 | 95.83 | 98.24 | 99.41 |
| 128 na FC3, 96 na conv2 | 64,5,2,0,1 |  | 96,5,2,0,1 |  | 128 | 0.5 | 30.54 | 63.81 | 88.77 | 95.92 | 98.41 | 99.48 |

Tabuľka 10 – výsledky sietí pre rozpoznávanie veku

Trénovaciu množinu sme doplnili o GROUPS databázu obsahujúcu takmer 28 tisíc obrázkov a *tabuľka 11* ukazuje výsledky po natrénovaní na upravených dátach. Tú sme tentokrát doplnili o balansovanú presnosť cez vekové kategórie (viď. 2.3.2). Vidíme, že chyba mierne stúpla ( presnosť klesla )

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | **Absolútna presnosť** | | | **Balansovaná presnosť** | | | |
| **Popis** | **conv1** | **drop1** | **conv12** | **drop2** | **fc3** | **drop3** | **2** | **5** | **10** | **2** | **5** | **10** | **15** |
| Menej neurónov na FC3 | 64,5,2,0,1 |  | 128,5,2,0,1 |  | 256 | 0.2 | 30.78 | 60.95 | 86.58 | 25.36 | 52.16 | 78.51 | 90.53 |
| Východiskový model | 64,5,2,0,1 | 0.1 | 128,5,2,0,1 | 0.2 | 512 | 0.5 | 27.38 | 59.54 | 86.83 | 24.51 | 54.76 | 82.44 | 93.51 |
| 7x7 kernel na conv1 | 64,7,2,0,1 | 0.1 | 128,5,2,0,1 | 0.2 | 512 | 0.5 | 28.10 | 61.11 | 87.42 | 24.10 | 54.98 | 81.22 | 93.18 |

Tabuľka 11 – výsledky sietí pre rozpoznávanie veku po pridaní GROUPS databázy

Z výsledkov môžeme dedukovať nasledovné

* V oboch prípadoch nám najlepšie vychádza architektúra s 256 neurónmi na plne prepojenej vrstve
* Podobné rozdiely v chybách jednotlivých architektúr môžeme pozorovať pri obmene dát
* Zväčšenie kernelu na prvej konvolučnej vrstve zvyšuje absolútnu presnosť
* Máme dosť zásadný rozdiel medzi absolútnou a balansovanou chybou. To je spôsobené nerovnomerným zastúpením vekových skupín v testovacej množine
* Prílišným zjednodušovaním siete začne sa zvyšovať chyba

#### Podrobná analýza výsledkov

Podobne ako pri trénovaní klasifikátora pohlavia, aj tu sme východiskový model pretrénovali s 500 tisíc iteráciami, aby sme videli, aký to bude mať dôsledok na finálnu chybu. Na *obrázku 21* si môžeme všimnúť, že nemá zmysel trénovať dlho, pretože zhruba okolo 40-tej tisícej iterácie nastane zlom, po ktorom ostáva validačná aj estimačná chyba stabilizovaná.



Obrázok 21 – vývoj estimačnej a validačnej chyby pre regresor veku

V tabuľke 12 máme podrobné výsledky pre jednotlivé vekové kategórie. Vidíme, že sieť má pomerne „*vysokú*“ úspešnosť pri určovaní veku detí a ľudí od 20-30 rokov, ale s čím je človek starší, tým sa schopnosť siete odhadnúť správny vek zhoršuje. To môže byť spôsobené dvoma faktormi.

1. V trénovacej množine sme mali málo ľudí nad 50 rokov a aj tí mali väčšinou vek zadaný pomerne veľkých v intervaloch ( GROUPS mala napríklad ľudí od 37-65 rokov )
2. Odhadovanie veku u ľudí nad 30 rokov je ťažká úloha aj pre človeka. Niektorí vyzerajú staršie, iní mladšie.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tolerancia** | **0-10** | **10-20** | **20-30** | **30-40** | **30-50** | **50-60** | **Priemerná presnosť** | **Balansovaná presnosť** |
| **2** | 37.604 | 32.652 | 31.971 | 18.565 | 16.198 | 13.532 | 28.025 | 25.087 |
| **5** | 77.786 | 70.566 | 65.236 | 47.376 | 38.253 | 31.225 | 60.058 | 55.073 |
| **10** | 89.067 | 94.023 | 88.811 | 86.256 | 80.825 | 58.525 | 87.225 | 82.918 |
| **15** | 93.245 | 97.834 | 95.871 | 97.209 | 97.186 | 79.567 | 95.582 | 93.485 |
| **20** | 97.423 | 99.019 | 98.703 | 99.584 | 98.953 | 91.644 | 98.514 | 97.554 |
| **25** | 99.095 | 99.653 | 99.761 | 99.99 | 99.346 | 96.11 | 99.545 | 98.992 |
| **30** | 99.652 | 99.908 | 100 | 99.99 | 99.738 | 97.903 | 99.845 | 99.532 |
| **35** | 99.721 | 99.99 | 100 | 100 | 99.935 | 99.628 | 99.967 | 99.879 |
| **40** | 99.791 | 100 | 100 | 100 | 100 | 99.831 | 99.985 | 99.937 |

Tabuľka 12 – podrobné výsledky pre vekové kategórie regresora veku

Na *obrázku 22* máme sadu subjektov , kde sieť odhadla takmer presný vek ( tolerancia jedného roka ). Môžeme si všimnúť, že sa jedná o mladých ľudí a kvalitné obrázky s  pohľadom na kameru.

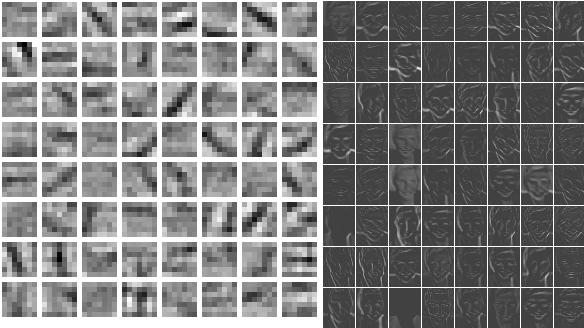


Obrázok 22 – subjekty s najlepšie odhadnutým vekom

Na *obrázku 23* sú ľudia, kde rozdiel medzi skutočným a predikovaným vekom bol väčší ako 15 rokov. Podobne ako pri klasifikácii pohlavia , aj tu sa nachádzajú obrázky, ktoré sú nekvalitné, zle anotované, s viacerými osobami, vytočenými hlavami alebo so zlým osvetlením



Obrázok 23 – subjekty s najhoršie odhadnutým vekom



Obrázok 24 – vizualizácia konvolučných filtrov a feature máp 1. konvolučnej vrstvy pre regresor veku

#### Návrh na zlepšenie

Pomocou prvej, nami natrénovanej siete, by sme trénovaciu množinu filtrovali tak, že by sme vyhodili všetky obrázky, kde sieť urobila veľkú chybu. Nad filtrovanými dátami by sme tréning zopakovali a tento proces by sme mohli iteratívne opakovať. Rozpoznávanie veku je však ťažký problém a väčšina dostupných databáz neposkytuje anotáciu s presným vekom, preto by bol potrebovali dáta rozšíriť o ďalšie databázy.

# Záver

V diplomovej práci sa nám podarilo nájsť vhodné architektúry hlbokých konvolučných sietí a úspešne sme zvládli úlohy rozpoznávania veku a pohlavia. V prácach podobného typu sa väčšinou používali veľké RGB obrázky, ale my sme ukázali, že porovnateľné výsledky vieme dosiahnuť s 8-bitovými obrázkami s rozlíšením 70x85 pixelov.

Našou výhodou bolo pomerne veľké množstvo dát, ktoré sme mali k dispozícii a ukázali sme, ako ich pripraviť, ako sa dajú filtrovať a efektívne spracovať pre potreby trénovania konvolučných sietí.

Úspešne sa nám podarila implementácia oboch úloh pomocou Caffe frameworku a dokázali sme, že s rýchlymi grafickými kartami je trénovanie hlbokých konvolučných sietí realizovateľné a pomerne rýchle.

Neurónovú sieť pre klasifikáciu pohlavia sme natrénovali na takmer 15 tisíc subjektoch a  500 tisíc obrázkoch a podarilo sa nám dosiahnuť zhruba 97% úspešnosť predikcie na testovacích dátach. Stačilo použiť sieť s dvoma konvolučnými vrstvami, pričom rozšírenie o tretiu, nemalo veľký zmysel, pretože výrazne stúpla výpočtová zložitosť a chyba klesla len minimálne. Všetky siete, ktoré sme skúšali dosahovali viac ako 93 % úspešnosť, čo znamená, že nezávisí tak na samotnej architektúre ako na dátach.

Ukázali sme, že pri hľadaní najlepšej architektúry siete, nie je potrebné robiť príliš dlhé tréningy, pretože validačná chyba v istom momente prudko klesne a ostáva stabilizovaná. Osvedčili sa nám trénovania, ktoré trvali 100 tisíc iterácií.

Odhadovanie veku je pomerne náročná úloha aj pre ľudí a kvôli limitovanému množstvu dát a  ich zlej kvalite sa nám podarilo dosiahnuť 30% úspešnosť pri tolerancii chyby 2 rokov. Pri tolerancii 5 rokov sme mali 60 %, pri 10 už 88 %. Ukázali sme, že rozpoznávanie veku je možné riešiť pomocou regresie.

Pre budúce práce sme pre klasifikátor pohlavia a regresor veku navrhli potenciálne vylepšenie, kde by sme pomocou nami natrénovaných sietí z  dát filtrovali zle anotované alebo nekvalitné obrázky, tréning zopakovali a tým zlepšili výslednú testovaciu chybu.

# Zoznam použitej literatúry

1. **V. Kvasnička, L. Beňušková, J. Pospíchal, I. Farkaš, P. Tiňo, a A. Kráľ.** *Úvod do teórie neurónových sietí.* Bratislava : IRIS, 1997. ISBN 80-88778-30-1.

2. **Andrew Gibiansky.** *Convolutional Neural Networks*. [Online] 2014. http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/.

3. **Makin, J.G.** *Backpropagation*. [Online] 2015. https://inst.eecs.berkeley.edu/~cs182/sp06/notes/backprop.pdf.

4. **karpathy@cs.stanford.edu.** *Convolutional Neural Networks (CNNs / ConvNets)*. [Online] 2014. http://cs231n.github.io/convolutional-networks/.

5. **Michael A. Nielsen.** Neural Networks and Deep Learning. [Online] Determination Press, 2015. http://neuralnetworksanddeeplearning.com/.

6. **Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick.** Caffe: Convolutional Architecture for Fast Feature Embedding. [Online] 2014. http://caffe.berkeleyvision.org. 1408.5093.

7. **Xiu-Shen Wei.** Tricks in Deep Neural Networks. [Online] 2016. http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html.

8. **Jeffrey Dean, Sanjay Ghemawat.** LevelDB. [Online] 2016. http://leveldb.org.

9. **Howard Chu.** Lightning Memory-Mapped Database. [Online] 2016. http://symas.com/mdb/.

10. **Gil Levi and Tal Hassner.** Age and Gender Classification using Convolutional Neural Networks, IEEE Workshop on Analysis and Modeling of Faces and Gestures (AMFG), at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, June 2015. [Online] http://www.openu.ac.il/home/hassner/projects/cnn\_agegender/.

11. **Eran Eidinger, Roee Enbar, Tal Hassner.** Age and Gender Estimation of Unfiltered Faces, Transactions on Information Forensics and Security (IEEE-TIFS), special issue on Facial Biometrics in the Wild, Volume 9, Issue 12, pages 2170 - 2179, Dec. 2014. [Online] http://www.openu.ac.il/home/hassner/Adience/EidingerEnbarHassner\_tifs.pdf.

12. ***SHUMEET BALUJA AND HENRY A. ROWLEY.*** *2007, "Boosting Sex Identification Performance." International Journal of Computer Vision, s. 111-119.*

*13.* ***Aleix M Martinez.*** *The AR Face Database. CVC Technical Report #24, June 1998. [Online] http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html.*

*14.* ***AT&T Laboratories Cambridge.*** *The Database of Faces. [Online] 2015. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.*

*15.* ***info@bioid.com.*** *The BioID Face Database. [Online] 2015. https://www.bioid.com/About/BioID-Face-Database.*

*16.* ***Bor-Chun Chen, Chu-Song Chen, Winston H. Hsu.*** *Cross-Age Reference Coding for Age-Invariant Face Recognition and Retrieval (CACD) ECCV 2014. [Online] http://vintage.winklerbros.net/facescrub.html.*

*17.* ***Dong Yi, Zhen Lei, Shengcai Liao and Stan Z. Li,.*** *“Learning Face Representation from Scratch”. arXiv preprint arXiv:1411.7923. [Online] 2014. http://www.cbsr.ia.ac.cn/english/CASIA-WebFace-Database.html.*

*18.* ***Wen Gao, Bo Cao, Shiguang Shan, Delong Zhou, Xiaohua Zhang, Debin Zhao.*** *CAS-PEAL Face Database. [Online] 2015. http://www.jdl.ac.cn/peal/.*

*19.* ***H.-W. Ng, S. Winkler.*** *Paris : s.n., 2014. A data-driven approach to cleaning large face datasets. (ICIP), Proc. IEEE International Conference on Image Processing.*

*20.* ***Karl Ricanek Jr and Tamirat Tesafaye.*** *“MORPH: A Longitudinal Image Database of Normal Adult Age-Progression,” IEEE 7th International Conference on Automatic Face and Gesture Recognition, Southampton, UK, April 2006, pp 341-345. [Online] https://ebill.uncw.edu/C20231\_ustores/web/product\_detail.jsp?PRODUCTID=10.*

*21.* ***Gross, R., Matthews, I., Cohn, J. F., Kanade, T., & Baker, S.*** *2008. Multi-PIE. Proceedings of the Eighth IEEE International Conference on Automatic Face and Gesture Recognition.*

*22.* ***G. Panis, A. Lanitis , N. Tsapatsoulis and T.F.Cootes.*** *“An Overview of Research on Facial Aging using the FG-NET Aging Database”. IET Biometrics. [Online] 2015. https://www.researchgate.net/publication/276026141\_An\_Overview\_of\_Research\_on\_Facial\_Aging\_using\_the\_FG-NET\_Aging\_Database.*

*23.* ***The National Institute of Standards and Technology (NIST) .*** *The Color FERET Database. [Online] 2015. http://www.nist.gov/itl/iad/ig/colorferet.cfm.*

*24.* ***G. Bradski.*** *"The OpenCV Library." Dr. Dobb’s Journal of Software Tools Key: citeulike:2236121, 2000. [Online] http://opencv.org/.*

*25.* ***Tristan Hume.*** *Simple, accurate eye center tracking in OpenCV. [Online] 2015. http://thume.ca/projects/2012/11/04/simple-accurate-eye-center-tracking-in-opencv/.*

*26.* ***A. Gallagher, T. Chen.*** *Understanding Groups of Images of People, IEEE Conference on Computer Vision and Pattern Recognition, 2009. [Online] http://chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html.*