Projektová dokumentace

Strojové učení a rozpoznávání 2019/2020

Vladimír Dušek David Ling Richard Mička Tomáš Strych

9. května 2020

1 Rozpoznávání obrazu – Vlastní neuronová síť

Pro klasifikátor hledané osoby na základě příznaků extrahovaných z obrázků byla sestavena a natrénována vlastní neuronová síť. Byl využit programovací jazyk Python s frameworkem `Tensorflow` a nadstavbou `Keras`.

1.1 Příprava dat

Nejprve bylo nutné připravit data pro trénování. Poměr trénovacích a validačních dat byl ponechán původní, tak jak byl rozdělen v zadání. Pro zpracování dat byl napsán skript `compile_data.py`. Ten nejprve za využití knihovny `OpenCV` načte obrázky do reprezentace `NumPy` *array*. Tím jsou získány příznaky (*features*) pro trénování. Každému obrázku je také přiřazena příslušná třída (*label*). Výsledkem tohoto procesu je dvojice (*feature*, *label*) pro každý trénovací či validační vzorek. Nakonec jsou pomocí knihovny `Pickle` uloženy příznaky a označení do binárních souborů pro trénovací i validační data.[1]

Později se ukázalo, že výsledky trénování nejsou vůbec uspokojící. Bylo vyzkoušeno spoustu nastavení parametrů sítě (více v sekci 1.2), avšak se nedařilo dosáhnout žádného výrazného zlepšení. Z toho bylo vyvozeno, že je k dispozici příliš málo trénovacích dat a síť se na nich není schopná nic rozumného naučit. Typický průběh trénování byl takový, že po prvních několika málo epochách již na validačních datech nedocházelo k žádnému zlepšení.

Z toho důvodu byla vyzkoušena technika augmentace dat. Pomocí jednoduchého Python nástroje imgaug (dostupný na `github.com/aleju/imgaug`) bylo základními úpravami dosaženo vytvoření nových dat. Byly použity úpravy jako zrcadlové otočení, přidání rozmazání, přidání šumu či mírné natočení obrazu. Na takto augmentovaných datech poté probíhalo další trénování a bylo dosaženo značně lepších výsledků.

1.2 Trénování

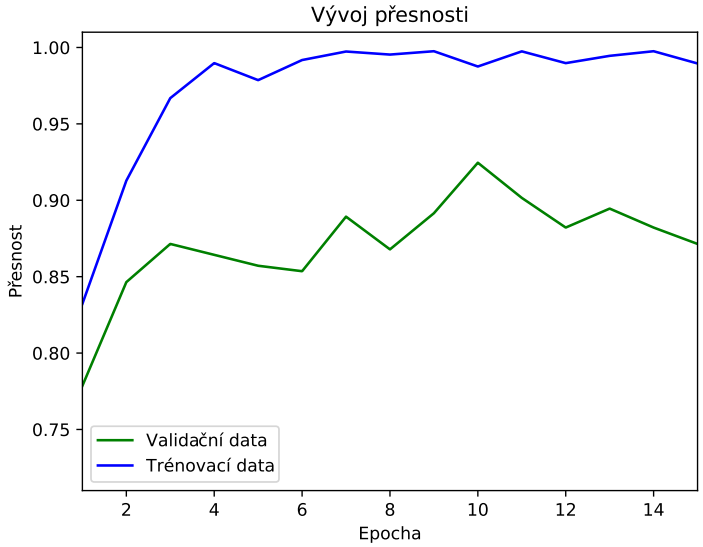
Pro trénování modelů byl napsán skript `train_models.py`. Nejprve jsou pomocí knihovny `Pickle` načteny již připravená trénovací data. Následně jsou příznaky normalizovány vydělením maximem. Tím je dosaženo toho, že každá hodnota reprezentující pixel je přeškálována do intervalu $(0, 1)$. Dalším krokem je, vzhledem k nerovnoměrnému rozložení trénovacích dat pro třídu *target* a *nontarget*, spočítat jejich poměr. Při trénování sítě je tento poměr zohledňován, a díky tomu síť uvažuje, že je každá třída stejně pravděpodobná. Dále je při trénování využit plugin `TensorBoard`, který ukládá logy z trénování (chyby a přesnosti na trénovacích/validačních datech).

Byly vyzkoušeny různé konfigurace neuronových sítí a celkem bylo natrénováno 108 modelů. Konkrétně počet konvolučních vrstev 1, 2 a 3. Velikost konvolučních jader 32, 64 a 128 neuronů. Konvoluční vrstvy následované *dropout* vrstvou. Počet plně propojených vrstev

[1] Je vhodné doplnit, že tento přístup není úplně vhodný, zejména pro velké datasety. Všechny obrázky jsou načítány do paměti, to je pochopitelně velmi limitující. Konkrétně by nebylo možné použít dataset větší, než je velikost operační paměti. V našem případě to vzhledem k malému datasetu není problém, avšak jinak je běžnou praxí používat Python generátory a data rovnou poskytovat trénovacímu modulu.

0, 1 a 2. Počet neuronů v plně propojené vrstvě 256 a 512. Všechny vrstvy obsahují aktivační funkci ReLU (*Rectified Linear Unit*). Pouze výstupní plně propojená vrstva o jednom neuronu má aktivační funkci *sigmoid*, aby skóre pro jednotlivé třídy bylo naškálováno do intervalu $(0, 1)$. Tím je získána pravděpodobnost náležitosti konkrétním třídám.

Na augmentovaných datech byly postupně všechny modely trénovány 15 epoch. Jako nejúspěšnější se ukázal model s jednou konvoluční vrstvou o velikosti 128 neuronů, následovanou *max pooling* vrstvou, *dropout* vrstvou s pravděpodobností $0,35$ a dvěma plně propojenými vrstvami s 512 neurony. Konkrétně po 10. epoše bylo dosáhnuto nejvyšší přesnosti na validačních datech 92 % (viz obrázek 1).



Obrázek 1: Vývoj přesnosti v průběhu testování.

1.3 Vyhodnocení

Pro vyhodnocení evaluačních dat byl napsán skript `eval.py`. Ten načte příslušný natrénovaný model a pomocí knihovny `os` na něm spustí detekci pro všechna evaluační data. Generuje výstupní soubor `out.txt`, do kterého zapisuje požadované informace, tedy jméno vzorku, skóre a výslednou třídu.

2 Rozpoznávanie obrazu - predtrénovaná sieť

Našim cieľom bolo dosiahnuť čo najlepších výsledkov za čo najmenej času. Preto sme sa rozhodli použiť predtrénované modely. Zvolili sme použitie modelu Resnet50. Vybrané boli dva modely, z ktorých jeden bol predtrénovaný na datasete VGGFace2[2], ktorý pozostáva z približne troch miliónov obrazov tváre a samotný model bol natrénovaný na klasifikáciu približne osem tisíc tried obrazov, a druhý na bol predtrénovaný na datasete ImageNet[3], ktorý pozostáva zo 14 miliónov obrázkov s približne 20 tisícami kategórií na klasifikáciu. Avšak v prípade ImageNet to nie sú exkluzívne obrazy ľudských tvárí. Ako prebiehalo trénovanie modelu ResNet50[4] predtrénovanom na datasete VGGFace2 je popísané v nasledujúcej podkapitole. Následne bude popísaná sieť ResNet50 s datasetom ImageNet, ktorý obsahuje rôzne kategórie obrázkov. Obidve predtrénované siete boli trénované pomocou nástroja `Google Colab` a využívajú knižnicu `PyTorch`.

[2]VGGFace2 - `http://www.robots.ox.ac.uk/~vgg/data/vgg_face2/`
[3]ImageNet - `http://image-net.org/`
[4]ResNet50 - `https://arxiv.org/abs/1512.03385`

2.1 ResNet50 s datasetom VGGFace2

Táto sieť pripadá ako vhodná voľba za účelom jednoduchej a pritom vysoko spoľahlivej klasifikácie tvárí, pretože ide o veľmi úspešnú architektúru hlbokej konvolučnej siete na klasifikáciu obrázkov ResNet, ktorá je špeciálne trénovaná na sade obrazov tvárí. Na základe toho sa dá predpokladať dobrá prenositeľnosť naučených váh aj na dátovú sadu dostupnú v rámci tohto projektu. Táto predtrénovaná sieť je trénovaná na klasifikáciu plibližne 8000 rôznych tried klasifikácie – ľudských tvárí.

Bežný vstup tejto siete má rozmery 3x224x224 (kanály, výška, šírka), čo nesedí s rozmermi dátovej sady projektu. Preto boli pred použitím tohto modelu potrebné kroky, ktorými bolo umožnené jeho požitie. Išlo o zmenu kroku (stride) prvej konvolučnej vrstvy pôvodného modelu z 2 na 1. Týmto krokom sa nezmenia predtrénované váhy konvolučných filtrov prvej vrstvy, pretože rozmery filtra sa nemenia, avšak má to za následok zmenu rozmerov výstupu tejto vrstvy. Zmenšením kroku filtra sa šírka a výška výstupu zdvojnásobí. To zaručí, že pri vložení ako vstupu obrázku s polovičnými rozmermi šírky a výšky, budú rozmery výstupu tejto vrstvy rovnakých rozmerov, ako pri pôvodnom kroku filtra v prvej vrstve a pôvodných rozmeroch obrazu. Týmto sa zmenšili rozmery vstupu modelu na veľkosť 3x112x112. Keďže rozmery obrazov boli 3x80x80, tento krok samotný nepostačuje a tak bolo potrebné všetky obrazy pri nahratí zväčšiť na rozmery 3x112x112. Zväčšovanie bolo realizované knižnicou `PIL` pomocou bilineárneho filtra počas nahrávania datovej sady. Posledným potrebným krokom bolo odčítanie stredných hodnôt jednotlivých kanálov trénovacej sady. Tieto parametre sú súčasťou predtrénovaného modelu.

Na samotnú klasifikáciu bol použitý výstup predposlednej vrstvy realizujúcej `average pooling` s výstupným vektorom o rozmere 2048 features, na ktorý bola pripojená a dotrénovaná lineárna vrstva s 2 výstupmi a funkciou `Softmax` medzi nimi. V pôvodnom predtrénovanom modeli bola posledná vrstva realizovaná obdobne na počet tried 8631 pomocou 1x1 konvolúcie (ktorá je ekvivalentná s lineárnou vrstvou). Počas trénovania boli upravované váhy len nami dodanej vrstvy a váhy predchádzajúcich vrstiev boli ponechané na pôvodných hodnotách z predtrénovaného modelu.

V rámci pokusu o použitie predtrénovaných modelov na sade VGGFace2 bol taktiež realizovaný pokus o použitie menšieho modelu Squzee-Excitation ResNet50 'SE-ResNet-50-128D' natrénovaného do embedujúceho 128-dimenzionálneho feature priestoru. Tento model však nebolo triviálne natrénovať na nám dostupnej trénovacej sade a vyžadoval by si väčšiu prácu, či už v nastavení a ladení parametrov pripojených výstupných vrstiev, alebo nastavovaní trénovacích parametrov a úprave datasetu. Z dôvodu jednoduchosti aplikácie pôvodného modelu ResNet 50 a jeho vynikajúcej schopnosti klasifikácie však neboli realizované ďalšie pokusy s použitím tohto menšieho modelu.

2.2 ResNet50 s datasetom ImageNet

Ideou bolo otestovať, či sa model naučený na rozpoznávaní rôznych kategórií obrázkov je schopný naučiť správne rozpoznávať tváre.

V tomto prípade bola poskytnutým obrázkom zmenená veľkosť na 3x250x250. Potom bol vystrihnutý stred obrázkov na rozmer 3x224x224. Po normalizácii dát tak mala sieť pripravené dáta k trénovaniu.

Cieľom siete je aby rozoznávala, azda sa na obrázku nachádza osoba, ktorú hľadamé alebo nie. Preto bola posledná vrstva pozmenená na lineárnu s 2 výstupmi. Okrem toho boli zmrazené váhy v celej sieti okrem poslednej vrstvy. Táto posledná vrstva má pred trénovaním náhodne generované váhy. Bola použitá CrossEntropyLoss funkcia.

Takto upravená sieť bola natrénovaná a vyhodnotená. Aby sme pri vyhodnocovaní získali pravdepodobnosti tak výstupi siete boli prepočítané cez softmax. Jednoduchým python skriptovaním bolo na evaluačnej sade vyhodnotené, koľko obrázkov bolo špecifikovaných správne a koľko nie. Pri nesprávne špecifikovaných obrázkoch pre jednotlivé kategórie sú zobrazené názvy obrázkov, čo umožňuje jednoduchšiu investigáciu toho prečo sa modelu nedarilo tieto obrázky správne rozoznať.

3 Rozpoznávání zvuku

3.1 Popis vybraných klasifikačních metod

Jako klasifikátor pro zvukové nahrávky byl použit gaussovský klasifikátor založený na směsici gaussovských rozložení (anglicky Gaussian Mixture Model, zkratkou GMM).
Tento algoritmu byl vybrán z důvodu toho, že dobře generalizuje i na neviděná data a jednoduché implementace. Algoritmus se snaží odhadnout co nejlépe parametry n gaussovských rozložení pro trénovací data a poté nově přichozí data klasifikovat do jedné z c tříd.
Pro trénink klasifikátoru byla vybrána metoda Expectation maximization, která pracuje tak, že se snaží nalézt takové parametry (střední hodnota a kovarianční matice) GMM, aby hodnota funkce L vracející věrohodnost dat, byla maximální.

3.2 Popis implementace a volba parametrů

Implementace byla provedena v jazyce Python za využití funkcí z knihovny `ikrlib`.
Všechna trénovací data se před spuštěním vložila do složky `data/train/číslo_třídy`, takže data pro hledanou osobu byly ve složce `data/train/1` a ostatní ve složce `data/train/2`. Data pro klasifikaci poté všechna do složky `data/eval`. Apriorní prvděpodobnost byla spočítána pro jednotlivé třídy na základě počtu trénovacích dat v obou třídách.
Po poslechnutí několika nahrávek bylo rozhodnuto, že káždá nahrávka bude oříznuta tak, aby neobsahovala počáteční a koncové ticho, které klasifikaci jednotlivých osob neprospěje. Z každé nahrávky tedy bylo odstraněno prvních 100 a posledních 300 vzorků, což výrazně zvýšilo úspěšnost klasifikátoru.
Bylo třeba rozhodnout, jaké parametry zvolit pro trénování klasifikátoru. Jedná se se o:

- Počet gaussovských rozložení
- Počet iterací trénování pomocí funkce `train_gmm`

Po testování s různými parametry vyšly jako nejlepší hodnoty 8 pro počet gaussovských rozložení a 110 pro počet iterací trénování.
Po natrénování byla pro všechna evaluační data spočtena pravděpodobnost pro obě třídy pomocí funkce `logpdf_gmm` a vybraná ta s vyšší pravděpodobností.

4 Výsledky

Nejlépších výsledků dosáhla předtrénovaná síť ResNet50 na datasetu VGGFace, která dosáhla dokonce nulové chybovosti. Také rozpoznávání z řečových nahrávek skončilo výsledkem s chybou v nízkých jednotkách procent. Kompletní výsledky lze vidět v tabulce níže.

Výsledky projektu				
Název klasifikátoru	Error	Cdet	minCdet	EER
cnn	11.31	12.97	12.89	15.02
gmm_speech	2.53	1.39	0.65	1.49
resnet50_finetuned_result	4.91	17.73	17.40	21.86
resnet50_vggface2_pretrained	0.00	0.00	0.00	0.00