VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Projektová dokumentácia k predmetu SUR

Detektor osoby

Dominik Boboš (xbobos00)
Tomáš Kender (xkende01)

25. apríla 2020

1 Úvod

Zadaním projektu je natrénovať detektor jednej osoby z obrázku tváre či hlasovej nahrávky. Detektor je implementovaný v programovacom jazyku **`Python 3.8`**.

2 Spustenie a používanie detektoru

2.1 Inštalácia knižníc

Na spustenie všetkých súčastí programu je potrebné mať nainštalované všetky potrebné knižnice a súčasti, ktoré sú obsiahnuté v súbore `requirements.txt`.

Okrem toho je potrebné mať nainštalované aj linuxové knižnice **`sox`** a **`libsndfile1-dev`**.

Inštaláciu pythonových knižníc je možné uskutočniť pomocou `pip`:

```
$ pip install -r requirements.txt
```

Prípadne:

```
$ pip3 install -r requirements.txt
```

Poznámka: Program sa dá spustiť aj priamo cez priložený dockerfile, ktorý nainštaluje linuxoé i pythonové balíky do virtuálneho prostredia a tým nerozhodí našu existujúcu konfiguráciu na počítači.

2.2 Dátová štruktúra

Program očakáva nasledujúcu dátovú štruktúru.

```
-- |-- data              // dáta určené k trénovaniu a testovaniu detektoru
       |-- target_dev
       |-- non_target_dev
       |-- target_train
       |-- non_target_train
   |-- eval              // dáta určené ku klasifikácií
   |-- src               // zdrojové kódy
-requirements.txt
-images_CNN.txt          // vyhodnotené dáta z eval modelom CNN
-images_SGD.txt          // vyhodnotené dáta z eval modelom SGD
-voice_RandomForest.txt  // vyhodnotené dáta z eval modelom RandomForest
```

2.3 Spustenie jednotlivých súčastí detektoru

Spustenie detektoru je možné pomocou príkazu.

```
$ python src/detector.py --system=face/voice
```

Prípadne:

```
$ python3 src/detector.py --system=face/voice
```

Po spustení sa natrénujú modely trénovacími dátami, modely sa uložia do priečinku `src` a následne sa uskutoční klasifikácia dát v `eval` a uložia sa výsledky do .txt súboru. Funkcie detektoru je možné upravovať pomocou prepínačou na príkazovom riadku.

```
--system    // [povinný argument] výber systému, ktorý sa má spustit'
               (system=face | system=voice)
--verbose   // informácie pre ladenie
--plot      // vykreslí informácie získané počas trénovania
--evalonly  // vyhodnotia sa len dáta v /eval (potrebné mat' uložené modely)
--trainonly // trénovanie modelu bez vyhodnotenia
--cnn       // použitie CNN modelu pre spracovanie obrázkov
```

3 Spracovanie obrázkov

Pred samotným trénovaním modelov je dôležité získať, čo najviac trénovacích a testovacích dát. Toto je docielené tým, že na trénovanie sa používajú všetky *.png* súbory v priečinku `data`. Obrázky sa načítajú pomocou knižnice *Pillow*. Target dátam sa nastaví label 1, ostatným label 0. Pomocou *ImageDataGenerator* z knižnice *Keras* sa vytvoria ďalšie trénovacie dáta a to vďaka modifikácií ako posun, rotacia, skrivenie, prevrátenie. Takto získané dáta smerujú modelom na trénovanie.

3.1 Model CNN

CNN je model konvolučných neuronových sietí, bol vybraný na základe zvedavosti, či CNN dokáže na akceptovateľnej úrovni rozpoznať tváre.

Tento model získava features obrázkov z rôznych vrstiev, známe aj ako *feature maps*, pomocou *pooling-u* postupne matice „zmenšujeme“, respektíve podvzorkujeme. Nakoniec použijeme funkciu *flatten* čím vytvoríme 1D pole. Ako optimalizátor je použitý **SGD** - *Stochastic-Gradient Descent*, a aby sme zabránili pretrénovaniu, dáta sa predkladajú v náhodnom poradí a zo všetkých dát sa náhodne vybere 20%, ktoré sa budú používať ako testovacie dáta.

Trénovanie modelu potom prebieha v desiatkách *Epochs* a samotné trénovanie trvá pomerne dlho. Model sa po trénovaní uloží `src/modelCNN.h5`.

3.2 Lineárny model SVM s SGD

Ako druhý model bol zvolený **SGDclassifier** použitý z knižnice *sklearn.linear_model*. Používa lineárny klasifikátor **SVM** - *Support Vector Machines* s trénovaním dát SGD. Model bol vybraný hlavne ako o dosť rýchlejšia alternatíva k CNN modelu vzhľadom na trénovanie modelu a celková rýchlosť v porovnaní s ostatnými klasifikátormi a pritom so stále použiteľnou úspešnosťou.

Nevýhodou sú vlastnosti modelov z knižnice *scikit*, kde je obtiažne získať soft score pre vyhodnotenie jedného obrázku, pri viacerých modeloch (*KNeighborsClassifier, GaussianProcessClassifier, GaussianProcessClassifier s modifikáciou RBF, RandomForestrClassifier*) sme dokázali vždy získať len celkovú presnosť modelu. Zo všetkých uvedených modelov bol SGD najrýchlejší, aj preto bol ponechaný v implementácií. Po natrénovaní sa model uloži do `src/modelSGD.pkl`

4 Spracovanie zvuku

4.1 Random Forest s MFCC

Z nahrávky sa najprv odstráni šum, následne sa oddelí ticho. Pre extrahovanie features bola zvolená knižnica *librosa*. Z nej boli využité MFCC a piptrack (čiže pitch hlasu) pre charakterizáciu nahrávky. MFCC featúry sa následne ešte ďalej škálovali a prehnali funkciou zscore z knižnice *scipy*.

Čo sa týka použitého klasifikátora, tu padla voľba na **Random Forrest Classifier** zo *sklearn.ensemble*. Jeho hlavnou črtou je delegovanie úloh na jednotlivé rozhodovacie stromy. Každý rozhodovací strom spraví predikciu a najrozšírenejší výsledok sa zobere ako výsledok. Základným predpokladom ale je to, že jednotlivé

stromy sú navzájom od seba nezávislé (čiže používajú pri vyhodnocovaní rôzne vlastnosti), a teda aj keď niektoré stromy dojdú k zlému výsledku, prevažná väčšina dojde k správnemu a túto chybu kvantitatívne vyváži.

Z toho dôvodu platí, ze čím vyšší máme počet stromov, tým vyššiu šancu máme na to, že prečíslime chybujúce stromy. Zároveň to ale pochopiteľne zvyšuje zaťaženie procesora. Druhou najvýznamnejšou konfigurovateľnou premennou je maximálna hĺbka stromu. Ladením a testovaním som nakoniec skončil pri lese s veľkosťou 96 stromov a zhodnou hĺbkou.

Pôvodne sa featúry skladali len z MFCC, delty a double delty. Tie mi však spoľahlivo nedokázali rozlišovať osoby a výsledky boli pomerne náhodné. Presnosť sa dosiahla až zmenou evaluovaných dát na MFCC+pitch hlasu. Práve pitch sa ukázal byť ako kľúčový faktor.

5 Záver

Pre detekciu tváre sa používajú celkovo dva modely. Presnosť CNN modelu je vyššia, čo je pravdepodobne spôsobené aj dlhším trénovaním dát. Presnosť by sa dala zvyšiť zmenou určitých nastavení ako napríklad pridanie ďalšieho filtru alebo použitím vhodnejšieho optimalizátoru. Pri SGD modele je najväčším problémom nemožnosť nastaviť získanie score pre vyhodnotený obrázok, avšak pre jeho rýchlosť a prijateľnú presnosť sme sa rozhodli ho ponechať v projekte. Z hladiska zvýšenia presnosti by bolo najvhodnejšie tento model vynechať, nakoľko lineárny klasifikátor nemusí byť najvhodnejšia voľba pre naše dáta.

Pre detekciu hlasovej nahrávky by mohlo byť zaujímavé porovnanie s GMM modelom, ktoré sa javilo byť rýchlejšie na natrénovanie, avšak to sa mi nepodarilo spojazdniť a musel som ostať pri Random Foreste. Systém vracia pre zhodu skóre 0.14-17 a pre nezhodu zhruba 0.08-0.12 až 0.13. Toto by šlo určite vylepšiť chytrejším výpočtom celkového skóre z výstupu predikcie modelu, ale pre naše potreby vracalo postačujúce výsledky aj existujúce riešenie, ktoré priemeruje čiastkové skóre. Ďalším možným vylepšením by mohlo byť pridanie delty a double delty piptracku do evaluácie na úkor MFCC. Na otestovanie tohto scenára už ale čas neostal.

Projekt nám dal praktický pohľad na teóriu preberanú na prednáškach a dal nám možnosť vyskúšať si metódy v praxi. Mohli sme vidieť viac či menej funkčné modely a vybrali sme spomedzi nich tie, ktoré nám prišli najvhodnejšie pre naše dáta. V budúcnosti by bolo možné zvyšiť kvalitu a presnosť detektorov vďaka lepšiemu nastaveniu parametrov, na ktoré je napríklad neurónová sieť veľmi citlivá a vedeli by sme sa vďaka nadobudnutým skúsenostiam lepšie zamerať na vhodnejšie modely.