VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

SUR – Strojové učení a rozpoznávání

Klasifikace obličejů a řeči

Martin Kostelník (xkoste12)
Ivo Meixner (xmeixn00)
Adam Gajda (xgajda07)

8. května 2020

1 Klasifikace obličejů

Klasifikátor obličejů je implenetován jako konvoluční neuronová síť. Systém byl vytvořen za pomoci knihovny Keras v jazyce Python.

1.1 Průběh implementace

Původní plán byl vytvořit klasifikátor obličejů pomocí knihovny TFLearn. Úspěšně jsem model vytvořil a otestoval. Bohužel jsem ale nemohl najít optimální konfiguraci neuronové sítě a model nefungoval úplně správně. Většinou označil úplně všechny data jako non-target a výsledná přesnost byla tedy 0.8571, což je ale poměr obrázků non-targetu ku všem datům. Tento model tedy není funkční a ve finálním odevzdání nebyl použit. Z kódu ovšem odstraněn nebyl, byl pouze zakomentován a všechny jeho části jsou označeny komentáři.

Rozhodli jsme se tedy použít knihovnu Keras. Vytvoření a otestování jednoduchého modelu bylo velice podobné modelu TFLearn. Nejdůležitější faktor pro použití této knihovny byl však nástroj Keras Tuner, který nám umožnil najít funkční (a v rámci možností nejoptimálnější) model. Nejprve jsme se seznámili s tím, jak Tuner funguje a vyzkoušeli jsme si najít několik málo modelů a z nich vybrat ten nejlepší. Jakmile jsme měli s Tunerem nějakou zkušenost, rozšířili jsme hodnoty hyper parametrů aby bylo vyzkoušeno více modelů a jednotlivé modely vyzkoušeny vícekrát. Hledání nejlepšího modelu nakonec trvalo asi šest hodin. Po dokončení ladění jsme měli k dispozici model, který byl nakonec využit k finální klasifikaci a generování výsledků. Samotný model je popsaný v souboru model_shape.

1.2 Reprodukce výsledků

Provádění příkazů se předpokládá ze složky `facial_recognition/`

1. Při implementaci byl použit Python 3.6.9
2. Potřebné balíky lze nainstalovat příkazem `pip install -r requirements.txt`
3. Je třeba nastavit konstanty v souboru `main.py`
   - TRAIN_DIR – Název adresáře se všemi trénovacími daty
   - VALIDATION_DIR – Název adresáře se všemi validačními daty
   - TEST_DIR – Název adresáře s testovacími daty
   - TARGET_LABEL – Prefix názvu obrázků obsahujících target
4. Příprava dat – Při prvním použití je třeba spustit funkce `create_data()` pro vytvoření trénovacích a validačních dat a funkci `process_test_data()` pro vytvoření testovacích dat. Tyto funkce vytvoří data v přípustném formátu pro model. Data jsou poté uloženy v souborech `testing_data.npy` a `validation_data.npy`. Při opakovaném použití není třeba data znovu vytvářet a stačí je načíst.
5. Při nepřítomnosti použitého modelu je dále třeba odkomentovat příslušné řádky (LINE 187-201). Nicméně hledání nejlepšího modelu může trvat i několik hodin. Další možností je model ručně vytvořit podle popisu v souboru `model_shape`, ale to může být problematické. Pro použití totožného modelu jako jsme použili my, je náš model dostupný v GitHub repozitáři na adrese: https://github.com/natiiix/vut-fit-sur/blob/master/facial_recognition/best_model.h5. Tento model lze jednoduše načíst provedením funkce `load_model()`. Z důvodu velikosti jsme jej nemohli přiložit do odevzdaného archivu.
6. Pokud je přítomen model a jsou nachystány testovací data, stačí provést funkci `recognize()` a výsledky budou vypsány na standardní výstup.

2 Klasifikace zvuku metodou K-NN

Metoda K-NN (K-Nearest Neighbors) je poměrně jednoduchá a přitom v určitých případech dokáže dosáhnout dostatečně dobrých výsledků. Díky tomu bylo možné implementovat celý klasifikátor na 60 řádcích v Pythonu. Byly použity knihovny NumPy a SciPy, které obsahují všechny běžně používané matematické funkce.

2.1 Popis funkce

Vstupem programu je seznam složek s trénovacími daty. Pro každou složku je nutné uvést, do které kategorie patří. Dále je potřeba předat cestu ke složce obsahující data, která mají být vyhodnocena. Pro každý WAV soubor nalezený ve vstupních složkách se provedou tyto úkony:

1. Načtení zvukových dat z disku.
2. Analýza dat pomocí FFT (Fast Fourier Transform) pro získání informace o dominantních frekvencích.
3. Vypočítání průměrné hodnoty FFT pro každou frekvenci, jelikož pracovat se všemi vzorky z každého audio souboru najednou není u K-NN z časových důvodů praktické. Běžný aritmetický průměr se ukázal jako nejpřesnější, bez výpočtu odmocniny nebo logaritmu z hodnot (aplikace těchto funkcí naopak zhoršila přesnost výsledků).
4. Uložení informací o frekvencích, společně s kategorií daného souboru, do seznamu vstupních dat.

Toto se provede i pro soubory, které mají být vyhodnoceny, s tím rozdílem, že kategorie není známa. Následně se začnou procházet a pro každý z nich se vyhledá K N-dimenzionálních bodů z seznamu vzniklém při zpracování vstupních dát. Z nalezených blízkých bodů se zjistí nejčastější kategorie, což je výsledná předpověď kategorie vyhodnocovaných dat.

2.2 Spouštění

Implementace se nachází ve složce `audio_knn/`. Provádění všech příkazů se předpokládá právě z této složky. Složka obsahuje soubor `known_wavs.pickle`, ve kterém jsou uloženy předem zpracované trénovací audio soubory. Dále se zde nachází adresář `data` obsahující všechna audio data, se kterými se má pracovat. Pokud je soubor `known_wavs.pickle` přítomen, stačí, aby se zde nacházela složka `eval` se soubory, které mají být vyhodnoceny. Pro vygenerování tohoto souboru je nutné přidat složky s trénovacími daty (`target_train`, `non_target_train`, `target_dev`, `non_target_dev`). Pro pokračování musí být nainstalován interpret Pythonu s verzí alespoň 3.6 (testováno na 3.7.7, ale jakákoliv 3.6+ verze by měla stačit). Předpokládá se, že tento interpret bude dostupný příkazem `python`. Pokud tomu tak není, je možné jej v příkazech nahradit cestou k příslušnému interpretu (např. `/usr/bin/python3.6`). Před spuštěním je nutné nainstalovat zmíněné závislosti. To se dělá příkazem [`python -m pip install --upgrade -r requirements.txt`]. Je možné použít virtuální prostředí (neboli virtualenv), ale na to není možné napsat univerzální návod, protože jejich používání závisí na platformě. Samotné spouštění lze provést příkazem [`python .`], případně [`python ./__main__.py`]. Výsledky jsou vypsány na standardní výstup. Pro uložení do souboru je nutné výstup přesměrovat (např. [`python . > results.txt`]).

3 Klasifikace zvuku metodou GMM

Metoda GMM (Gaussian Mixture Model)
Program input - Složky obsahující data trénovací a evaluační.
Program output - Soubor obsahující jméno vyhodnoceného data, výsledné skóre a binární vyhodnocení zda-li se jedná o náš cíl či nikoli.

3.1 Funkcionalita

Extrakce příznaků ze zvukových nahrávek s příponou `wav` je provedena za pomoci funkce z knihovny ikrlib která nám vytvoří za pomocí segmentace, Fourieroveho spektra, banky filtru, logaritmizace a MFCC (Mel Frequency Cepstral Coeficients) slovník obsahující dvojici názvu souboru a matice hodnot. Následně budou matice sloučeny do jednoho velkého seznamu. Po nastavení proměnných obou tříd target, non_target (apriorní pravděpodobnosti třídy, počet komponent GMM třídy, střední hodnoty, kovarianční matici, váhy komponent) se provede N počet iterací samotného trénovaní, kde se s maximální věrohodností určí nové hodnoty již dříve zmíněných proměnných, které by měly zapříčinit přesnější odhad gaussovek. Ted už jen zbývá vyhodnotit evaluační data s již natrénovaným GM modelem a vypsat výsledky do souboru.

3.2 Reprodukce výsledků

1. Při implementaci byl použi Python 3.8.2
2. Silně doporučujeme použít operační systém Windows (alternativně nutnost modifikovat řádek 13-14)
3. Potřebné balíky lze nainstalovat příkazem `pip install -r requirements.txt`
    - target_train – Adresář se všemi cílovými trénovacími daty s příponou `wav`
    - non_target_train – Adresář se všemi ne-cílovými trénovacími daty s příponou `wav`
    - eval – Adresář se všemi evaluačními daty s příponou `wav`
4. Mít připravená data cílů, ne-cílů a data která budou použita na následnou evaluaci úspěšnosti celého systému.
5. Mít nastavené apriorní pravděpodobnosti (P_t, P_nt), počet komponent gaussovy směsice (M_t, M_nt) a počet iterací samotného trénování (N)