# 1 Úvod

Tento dokument je dokumentáciou riešenia projektu pre predmet SUR na FIT VUT. Cieľom projektu bolo natrénovať detektor jednej osoby z obrázku tváre a hlasovej nahrávky. Pre túto úlohu boli na dodaných trénovacích vzoroch natrénované dva detektory, konkrétne *Gaussian Mixture Model* pre detekciu osoby z nahrávky, a *Support Vector Machine* pre detekciu z obrázka.

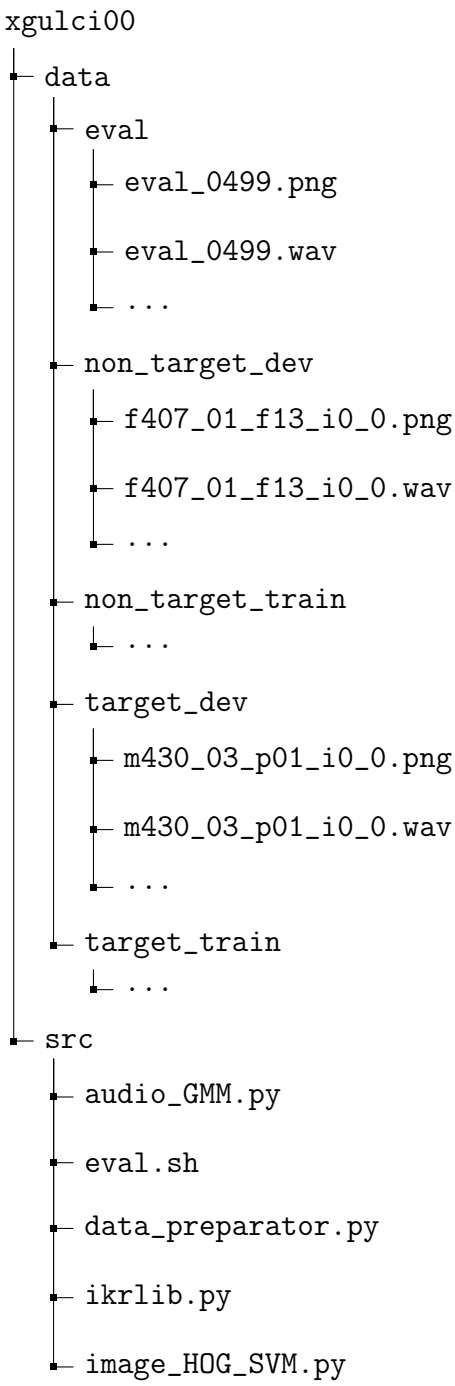
## 1.1 Spustenie a použité nástroje

Projekt bol implementovaný v jazyku `Python 3.9`. Pre implementáciu niektorých funkcií bola použitá knižnica `ikrlib.py`. Keďže táto knižnica bola implementovaná v staršej verzii pythonu, bolo nutné v nej vykonať niekoľko drobných zmien pre zaistenie kompatibility s akutálnym riešením. Väčšina z nich sa však týka použitia novších verzií knižníc, či úpravy spôsobu delenia dvoch čísel, ktoré oproti pythonu 3 vždy vrátili celočíselný výsledok.
Odovzdaný archív obsahuje nasledujúce súbory:

- `src/audio_GMM.py` - implementácia detektora z hlasovej nahrávky,
- `src/data_preparator.py` - funkcie pre generovanie požadovaného množstva augmentovaných dát, a funkcie pre samotné augmentácie[1],
- `src/eval.sh` - skript použitý pre vyhodnotenie behu a ladenie hyperparametrov detektora z hlasovej nahrávky,
- `src/ikrlib.py` - knižnica `ikrlib` s drobnými úpravami pre kompatibilitu riešenia,
- `src/image_HOG_SVM.py` - implementácia detektora z fotky tváre,
- `src/requirements.txt` - súbor so zoznamom knižníc potrebných pre spustenie programu,
- súbory s výsledkami evaluácie a túto dokumentáciu.

Implementácia predpokladá štruktúru uloženia súborov popísanú na Obrázku 1.1, konkrétne cesty k dátam však možno zmeniť podľa potreby vo funkcii `parse_args()` príslušného programu, či zmenou hodnoty konštanty `_EVAL_DATA_PATH` pre ostré, testovacie dáta. Pri trénovaní je nutné ponechať formát pomenovania súborov uvedený v zadaní (napr. `f401_01_f21_i0_0.png`), implementácia pracuje s identifkátormi osoby v názve, ktorý je pri detekovanej target osobe *m430*. Spustenie evaluácie ostrých dát je u oboch detektorov možné volaním funkcie `_evaluate_test_data()`.

[1] Toto platí pre obrazové, aj zvukové dáta. Dáta je tu tiež možné vygenerovať a uložiť na disk. Samotné implementácie detektorov však augmentácie vytvárajú za behu programu a dáta spracovávajú bez ich ukladania.




Obrázok 1.1: Predpokladaná hierarchia súborov

# 2 Detekcia osoby z nahrávky

Pre implementáciu detektora osoby z nahrávky bol zvolený prístup využívajúci Gaussian Mixture Model, pretože umožňuje modelovať komplexné rozloženia v príznakoch akými sú napr. MFCC koeficienty. Pri prvotnom riešení boli na postavenie GM modelu volené rôzne prístupy. Pôvodne bola každej osobe v konkrétnom sedení priradená vlastná trieda, následne bola trieda priradená každej osobe nezávisle na sedení. Väčší počet Gaussovských komponent na triedu však na takto postavenom probléme nefungoval dostatočne dobre, keďže zastúpenie trénovacích dát v konkrétnej triede sa nieslo v rade jednotiek. Prístup, ktorý bol nakoniec zvolený pre natrénovanie detektoru funguje na spôsob binárnej klasifikácie. Osoby, ktoré nechceme detekovať sú pri trénovaní priradené do jednej, *non-target* triedy, a osoba ktorú detekovať chceme, je priradená do *target* triedy. Takéto postavenie problému však na dostupných dátach so sebou nesie predpoklad, že model si bude istejší na *non-target* dátach, keďže ich je násobne viac. Tento predpoklad potvrdili aj pôvodné výsledky, ktoré v sebe niesli vysoký počet *false negative* zaklasifikovaných vzorov. Vysporiadanie sa s týmto problémom popisuje Sekcia 2.1. Z každej nahrávky v trénovacej množine boli najskôr extrahované MFCC príznaky, na ktorých bol ďalej trénovaný model. Pri postavení modelu je pre každú triedu použitý rovnaký počet Gaussovských komponent. Na začiatku sú jednotlivé parametre vypočítané nasledovne. Vrámci každej triedy je váha každej Gaussovskej komponenty rovnaká. Body v triede sú náhodne priradené jednotlivým komponentám, a je z nich vypočítaná pôvodná počiatočná stredná hodnota a kovariančná matica. V jednotlivých iteráciách trénovania sa body prehadzujú medzi Gaussovskými komponentami a prepočítavajú sa ich parametre, kým trénovanie nedosiahne konvergenciu, alebo zadaný počet iterácií. Pri klasifikácii každého bodu sa potom počíta rozdiel logaritmov pravdepodobností, že bod patrí do jednej triedy oproti druhej. Tento rozdiel, tzv. *soft score* slúži na rozhodnutie o klasifikácii bodu. Toto skóre je porovnané so zadanou rozhodovacou hranicou, čím vyvodíme finálne rozhodnutie z množiny $\{0, 1\}$. Teda:

$$\text{GMM}(x) = \begin{cases} 1, & \text{if } \log \mathcal{L}(x \in \text{target}) - \log \mathcal{L}(x \in \text{non-target}) > -500 \\ 0, & \text{inak,} \end{cases}$$

pre klasifikované dato $x$, a rozhodovaciu hranicu **-500**.

## 2.1 Augmentácia dát

Pri načítaní trénovacích dát bolo z každej nahrávky odstránených prvých 1.5 sekundy. Táto časť nahrávky obsahuje artefakt z nahrávacieho zariadenia, ktorý nenesie žiadnu hodnotu o identite rečníka. Keďže poskytnutá dátová sada bola relatívne malá, a obsahovala nerovnomerné zastúpenie dát pre target a non-target triedy, tento dataset bol obohatený o augmentované dáta.

Krok augmentácie dát umožňuje umelo zväčšiť trénovaciu množinu, a zároveň vyrovnať počet dát v oboch triedach. Pri snahe pracovať s nevyváženým datasetom je vysoká pravdepodobnosť, že zvolená technika bude ignorovať minoritnú triedu, a vykazovať na nej nedostatočne dobré výsledky. V prípade tejto úlohy je však minoritná trieda mimoriadne dôležitá, keďže jej správne "naučenie sa" modelom je cieľom projektu. Obyčajná duplikácia dát by do trénovania nepriniesla žiadnu novú informačnú hodnotu. Môžme teda z dostupných dát vysyntetizovať nové pomocou vhodných augmentácií. Pred samotným spustením augmentácie zistíme váhy, alebo množstvo novo-vygenerovaných vzorov pre každú triedu. Na základe dostupných dát a predpripravených augmentácií, bolo vygenerované nasledujúce množstvo dát, pričom vo finálnom trénovacom datasete po augmenátcii, boli zachované aj pôvodné dáta:

Table 2.1: Počet trénovacích vzorov pred a po augmentovaní zvukovej dátovej sady.

| | Number of Samples | |
|---|---|---|
| | before augmentation | after augmentation |
| target | 20 | 152 |
| non-target | 132 | 151 |

Po určení žiadaného počtu vygenerovaných vzorov pre danú triedu na hlasové nahrávky aplikujeme augmentácie, ktorými sú pridanie šumu, reverbácia, zmena výšky tónu či natiahnutie času pre zmenu dĺžky nahrávky. Parametre každej z augmentácií sú generované stochasticky. To, či sa na konkrétnu nahrávku aplikuje konkrétna augmentácia je tiež určené náhodne. Každá z nahrávok je teda obohatená o náhodne namiešanú množinu augmentácií. Nehrávky pre augmentáciu nie sú vyberané náhodne, ale cyklicky sa iteruje cez zoznam dostupných vzorov, kým nevygenerujeme počet požadovnaných nových vzorov. Pokiaľ pretečieme na koniec zoznamu, iterujeme znova od začiatku.

## 2.2 Ladenie hyperparametrov

Pri ladení parametrov prebiehali experimenty s nastavovaním rôznych počtov gaussovských komponent na jednu triedu, a s počtom iterácií trénovania. Keďže model bol trénovaný na augmentovaných dátach, v ktorých majú obe triedy vyvážené zastúpenie, možno sa opierať o metriku *accuracy*. Na základe niekoľkých experimentov bola teda za finálnu konfiguráciu modelu zvolená verzia s **8 komponentami na triedu**, a **64 iteráciami trénovania**. Priebeh vývinu hodnôt na základe ladenia parametrov možno sledovať v Tabuľke 2.2 a v Tabuľke 2.3. Po získaní fixných hodnôt bola ďalej vyhodnocovaná vhodná hodnota prahu pre rozhodnutie, či ide o detekovanú osobu, alebo nie. Priebeh vplyvu hodnoty prahu na výsledky možno vidieť

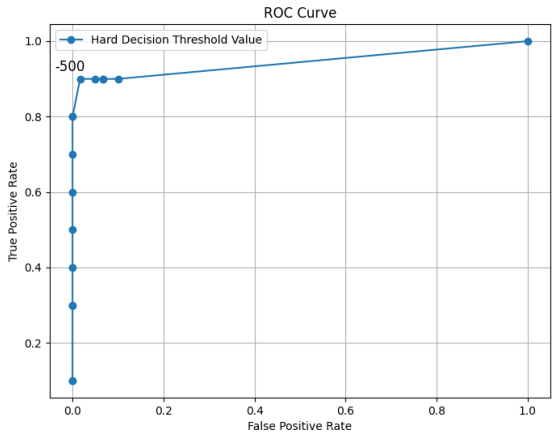
Table 2.2: Porovnanie výsledkov pre rôzny počet Gaussovských komponent na jednu triedu a fixný počet iterácií.

| _GAUSSIANS_PER_CLASS | _TRAIN_ITERATIONS | TP/TN/FP/FN | Accuracy [%] | Precision [%] | Recall [%] | F1 Score [%] |
|---|---|---|---|---|---|---|
| 1 | 128 | 10/48/12/0 | 82.86 | 45.45 | 100 | 62.5 |
| 2 | 128 | 8/58/2/2 | 94.29 | 80 | 80 | 80 |
| 3 | 128 | 7/59/1/3 | 94.29 | 87.50 | 70 | 77.78 |
| 5 | 128 | 7/10/0/3 | 85 | 100 | 70 | 82.35 |
| **8** | **128** | **8/60/0/2** | **97.14** | **100** | **80** | **88.89** |
| 9 | 128 | 9/58/2/1 | 95.71 | 81.82 | 90 | 85.62 |
| 16 | 128 | 9/58/2/1 | 95.71 | 81.82 | 90 | 85.62 |
| 20 | 128 | 10/57/3/0 | 95.71 | 76.92 | 100 | 86.96 |

Table 2.3: Porovnanie výsledkov pre fixný počet Gaussovských komponent na jednu triedu a premenný počet iterácií.

| _GAUSSIANS_PER_CLASS | _TRAIN_ITERATIONS | TP/TN/FP/FN | Accuracy [%] | Precision [%] | Recall [%] | F1 Score [%] |
|---|---|---|---|---|---|---|
| 8 | 16 | 8/60/0/2 | 97.14 | 100 | 80 | 88.89 |
| 8 | 32 | 8/60/0/2 | 97.14 | 100 | 80 | 88.89 |
| **8** | **64** | **9/60/0/1** | **98.57** | **100** | **90** | **94.74** |
| 8 | 128 | 8/60/0/2 | 97.14 | 100 | 80 | 88.89 |
| 8 | 200 | 9/59/1/1 | 97.14 | 90 | 90 | 90 |

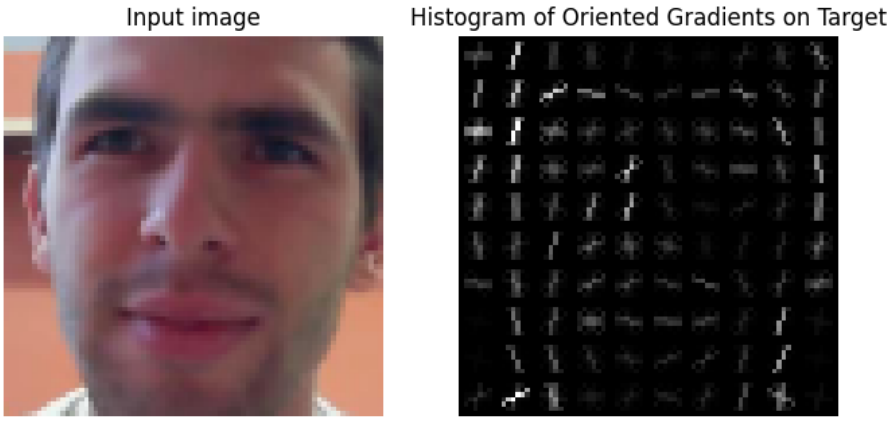
na Obrázku 2.1. Za hodnotu prahu pre vyhodnotenie na ostrých dátach bola zvolená hodnota **-500**, ktorá na krivke dominuje všetky ostatné hodnoty. Je zrejmé, že pri evaluácii na rôznom počte komponent a iterácii je nutné model odznova natrénovať. Keďže model je vždy trénovaný na dátach, na ktoré sú stochasticky aplikované augmentácie, v implementácii bol generátoru náhodných čísel nastavený seed (`numpy.random.seed(42)`), ktorý zaisťuje reprodukovateľnosť a porovnateľnosť výsledkov.




Obrázok 2.1: Vývin kvality detekcie pre rôzne hodnoty rozhodovacieho prahu.

# 3 Detekcia osoby z obrázka

Pre implementáciu detektora osoby z obrázka bol použitý klasifikátor Support Vector Machine. K problému bolo opäť pristúpené ako k binárnej klasifikácii, systém sa nesnaží detekovať konkrétnu osobu na vstupe, ale len fakt, či ide o cieľovú osobu, alebo nie. Pred samotnou klasifikáciou bolo nutné z trénovacích obrázkov extrahovať príznaky pre klasifikáciu. V tomto prípade bol zvolený deskriptor *Histogram of Oriented Gradients* (HOG). Jeho vlastnosti a typické použitie ponúkli dobrý predpoklad pre jeho schopnosť z obrázka extrahovať príznaky s vysokou informačnou hodnotou. Táto metóda pracuje s gradientom (kombinácia magnitúdy a uhla) celého obrázka, aj jednotlivých pixelov. Gradienty sú určitým spôsobom rozdelené do blokov, pre ktoré sú následne vypočítané histogramy. Po určitom počte krokov sú ďalej histogramy konkatenované do vektorov príznakov. Po aplikácii špecifických normalizácii je tak možné obdržať príznaky vhodné pre detekciu. Výpočet týchto príznakov bol vykonaný pomocou funkcie z knižnice `scikit-image`. Príklad HOG-u pre konkrétny trénovací vzor možno vidieť na Obrázku 3.1




Obrázok 3.1: HOG pre detekovanú osobu.

Pri tomto použití SVM nebolo prístupené k ladeniu žiadnych parametrov. Použitá funkcia z knižnice `sklearn` pre zadané trénovacie dáta natrénuje klasifikátor, pričom umožňuje špecifikovať použitý typ kernelu, váhy jednotlivých tried, či ručné nastavenie parametrov modelu. Následne je do tohto klasifikátora možné poslať validačné dáta, pre ktoré vráti nielen príslušnosti do konkrétnych tried, ale aj pravdepodobnosť tejto príslušnosti. Nie je tak nutné voliť vhodný prah, či iné parametre pre rozhodnutie. Pri trénovaní modelu bolo experimentované s rôznymi typmi kernelov, obmeny v ich type však neprispeli k zlepšeniu v presnosti detekcie.

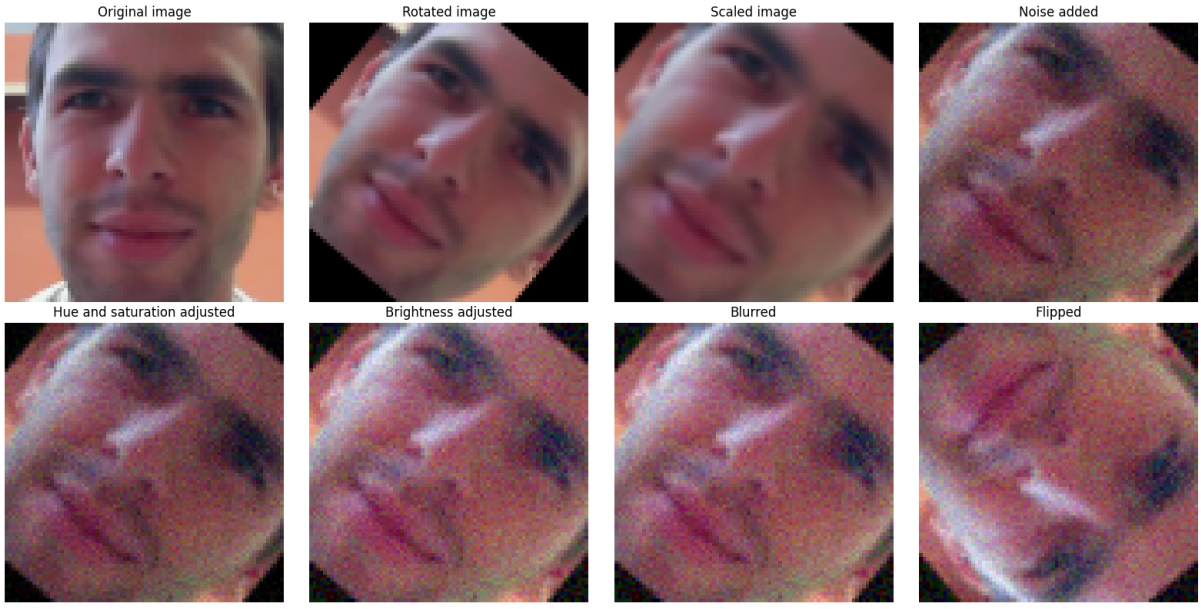
## 3.1 Augmentácia dát

Keďže takáto implementácia detektora nenecháva moc priestoru pre ladenie hyperparametrov, snaha vylepšiť model bola sústredená na vytvorenie robustnej trénovacej dátovej sady. Podobne ako pri zvukových dátach, aj nad týmito dátami boli vykonané augmentácie pre jej umelé zväčšenie a vybalansovanie počtu zastúpenia oboch tried v trénovácích dátach. Keďže počet zvolených augmentácií aj ich intenzita je v tejto implementácií pri obrazových dátach vyššia ako pri zvukových, množstvo vygenerovaných dát je mnohonásobne väčšie. Cieľom bolo tak zabezpečiť rovnomerné rozloženie predstavených artefaktov po dátovej sade.

Table 3.1: Počet trénovacích vzorov pred a po augmentovaní obrazovej dátovej sady.

| | Number of Samples | |
|---|---|---|
| | before augmentation | after augmentation |
| target | 20 | 300 |
| non-target | 132 | 300 |

Do dát boli zanesené nielen intenzitné transformácie ktoré upravujú obrázku jas či saturáciu, ale aj priestorové transformácie, ktoré dáta orotujú, scalujú, či do nich zanášajú šum. Aplikácia konkrétnej transformácie, rovnako ako hodnoty parametrov pre jednotlivé transformácie boli opäť generované náhodne. Príklad aplikovaných augmentácií možno vidieť na Obrázku 3.2.




Obrázok 3.2: Augmentácia trénovacej dátovej sady.

# 4 Záver

Na základe vykonaných evaluácií možno zhrnúť, že zvolené detektory fungujú relatívne dobre aj pri malom počte dostupných dát. Zvolený prístup s augmentáciami dát pre ich umelé namnoženie docielil väčšiu robustnosť modelu a tiež jeho schopnosť rozlišovať medzi triedami bez závažnejšieho biasu voči jednej z nich.

Medzi známe obmedzenia akutálneho riešenia pri detektore rečníka patrí napríklad ignorácia konvergencie riešenia pri trénovaní, keďže model sa natrénuje na relatívne malom počte iterácií. Žiadalo by sa kontrolovať, o koľko sa medzi jednotlivými iteráciami zmenia váhy, či iné parametre Gaussovskej komponenty, a v bode kedy táto zmena podtečie určitú tolerovanú konštantu, trénovanie ukončiť. Ďalej by bolo vhodné experimentovať s cross-validáciou a využiť tak dataset obmedzenej veľkosti efektívnejšie pre ešte lepšie ladenie parametrov. Aktuálne riešenie tiež ponúka priestor na zlepšenie v smere štruktúrovania samotnej implementácie, a experimentovanie s inými modelmi, ktoré umožňujú väčšiu flexibilitu pri ladení hyperparametrov. Na detekciu osoby z obrázka by napríklad tiež mohol byť použitý GMM.