Vysoké učení technické v Brně
Fakulta informačních technologií

Strojové učení a rozpoznávání

Projektová dokumentace

Klasifikace obrázků obličejů a hlasových nahrávek

Bc. Tomáš Rajsigl ‹xrajsi01›
Bc. Petr Bartoš ‹xbarto0g›

5. května 2025

1 Použité technologie a knihovny

Projekt je implementován v jazyce **Python 3.10.12**. Pro trénování a vyhodnocování neuronové sítě pro klasifikaci obličejů je využita knihovna `PyTorch`. Načítání a transformace obrazových dat zajišťují knihovny `torchvision` a `Pillow`. Pro vizualizaci výsledků a dodatečné numerické operace je použit `matplotlib` a `numpy`.

2 Struktura projektu a spuštění

Zdrojové kódy implementovaných systémů se nachází ve složce `src/`, přičemž provádění všech příkazů se předpokládá právě z této složky.

```
/ . . . . . .
├── src/ . . . . . Zdrojové kódy implementovaných systémů
│   ├── train/ . . . . . Adresář obsahující trénovací data
│   ├── dev/ . . . . . Adresář obsahující testovací data
│   ├── face_classifier.py . . . . . Klasifikátor člověka dle obrázku obličejů
│   └── audio_classifier.py . . . . . Klasifikátor řečníka dle hlasové nahrávky
├── README.md . . . . . Stručný návod ke spuštění projektu
├── requirements.txt . . . . . Seznam potřebných knihoven
├── dokumentace.pdf . . . . . Technická zpráva popisující jednotlivá řešení
├── image_CNN . . . . . Soubor s výsledky klasifikace obrázků
└── audio_GMM . . . . . Soubor s výsledky klasifikace nahrávek
```

Spuštění trénovacích skriptů

Kód lze spustit sekvencí následujících kroků:

1. Instalace externích knihoven: `pip install requirements.txt`
2. Spuštění hlavního skriptu: `python3 <script> <cmd>`
3. Výsledné grafy a ukázky predikcí osob z obličejů jsou poté k nalezení v:
   - `loss_accuracy_per_fold.pdf`
   - `sample_predictions.pdf`

   Obdobně tak grafy z predikce osob na základě hlasové nahrávky:
   - `ll_accuracy_per_fold.pdf`

kde `<script>` může být buď `face_classifier.py` nebo `audio_classifier.py`. Pro spuštění trénování je pak název skriptu následován `--train` v případě detekce obličejů nebo `train` v případě detekce hlasů.

Spuštění klasifikace skriptů

V případě klasifikace je postup prakticky stejný a mění se pouze přepínače pro daný skript. Pokud chceme spustit skript pro detekci obličejů, lze využít `--eval --model_path <path>` a v případě detekce hlasů `classify --eval_dir <path> --model_dir <path>`.

3 Rozpoznávání obličejů

Pro klasifikaci obrázků jsme zvolili relativně jednoduchou konvoluční neuronovou síť se třemi reziduálními bloky. Architektura vypadá následovně:

- **Vstupní vrstva:** Konvoluce 7×7 → *BatchNorm* → *ReLU* → *MaxPool*
- **Reziduální bloky:**
  - každý blok obsahuje dvě 3×3 konvoluce s BatchNorm a reziduálním spojem,
  - bloky inkrementálně zvyšují počet kanálů (32 → 64 → 128 → 256) a v druhém a třetím bloku se vstup podvzorkovává.
- **Výstupní vrstva:** AdaptiveAvgPool → plně propojená vrstva na 31 tříd.

Tato jednoduchá síť inspirovaná architekturou ResNet poskytuje dostatečnou kapacitu pro zachycení klíčových rysů obličejů. I přesto, že se jedná o mělkou síť s malým počtem parametrů, reziduální spojení umožňují jak lepší průtok informací, tak zachování nízkoúrovňových rysů jako například tvary očí či nosů a zrychlují v našem případě konvergenci.

3.1 Popis přístupu k trénování

Data ve složkách `train` a `dev` jsou pojmenována tak, aby kódovala jak identifikátor osoby, tak číslo nahrávacího sezení (např. `f401_01_f12_i0_0.png`). Za účelem lepšího využití poskytnutých dat při trénování využíváme metodu *jackknifing*, při níž data rozdělujeme právě podle nahrávacích sezení.

Technika Jackknifing

V praxi takto model učíme na čtyřech samostatných tzv. *foldech*, přičemž jeden fold představuje vždy konkrétní nahrávací sezení vyčleněné pro validaci a zbývající tři slouží k učení modelu. Fold je tedy jednou „skupinou“ dat, na které testujeme schopnost generalizace. Tento přístup se osvědčil zejména díky tomu, že model trénovaný na různých kombinacích nahrávacích sezení lépe zachycuje variabilitu vstupů a výrazně se pozitivně projevil na celkové přesnosti systému. Po dokončení každého fold-u ukládáme váhy modelu a pro finální klasifikaci volíme výhradně tu sadu vah, která dosáhla nejlepší validační přesnosti.

Mechanismus Early Stopping

Pro zamezení přetrénování využíváme techniku *early stopping*. Po každé trénovací epoše vyhodnocujeme validační přesnost a v případě, že se tato metrika po třiceti (tato hodnota byla zvolena zcela empiricky) po sobě jdoucích epochách nezvýší, trénink se ukončí a uchovají se váhy z předchozího nejlepšího bodu.

Současně při detekci stagnace validační přesnosti dynamicky snižujeme učící rychlost určenou parametrem *learning rate* na polovinu, čímž dosahujeme hladšího a jemnějšího doladění modelu.

Augmentace dat

Počet trénovacích dat je sám o sobě pro komplexitu využívané neuronové sítě poměrně malý, a proto jsme pro zvýšení rozmanitosti trénovacích vzorů zařadili následující transformace:

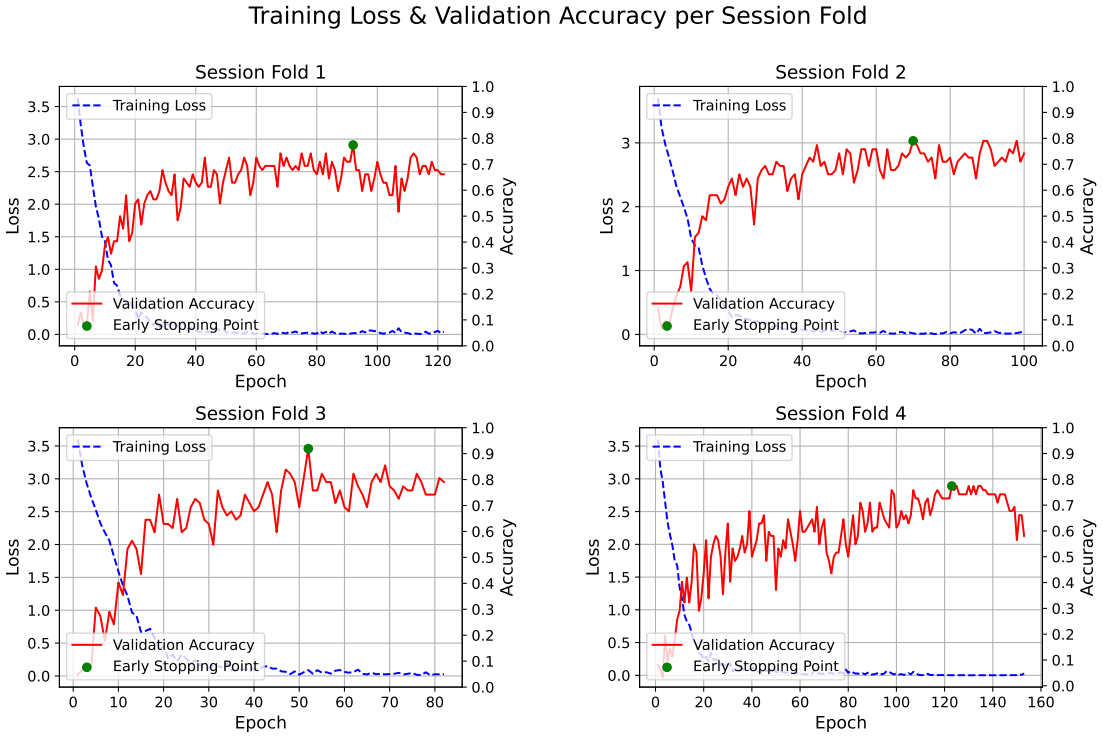
- náhodné horizontální **převrácení** a drobné **rotace**,

- variace **jasu**, **kontrastu** a **sytosti**,
- **zaostření** či naopak **rozostření** nebo jemné **zašumění** a
- standardní **normalizace** kanálů pro stabilnější učení.

Augmentace dat se na závěr jevila jako naprosto klíčová pro dosažení generalizace modelu a významně přispěla k celkové přesnosti systému.

3.2 Vyhodnocování

Systém je vyhodnocován na základě přesnosti predikcí na validační množině v každém fold-u. Průběh trénování z pohledu trénovací chyby a validační přesnosti je znázorněn na obrázku 3.1.



Obrázek 3.1: Vizualizace trénování na sadách dat dle jednotlivých nahrávacích sezení s využitím techniky *jackknifing*. Při trénování byla využita standardní chybová funkce *Cross Entropy* a optimalizační algoritmus *Adam*.

Jednou z možných cest k dalšímu zlepšení by bylo nasazení hlubší architektury, například přidání dalších bloků či zvýšení počtu výstupních kanálů jednotlivých filtrů. Avšak vzhledem k omezenému množství dat hrozí, že takto zvětšený model si pouze zapamatuje trénovací vzory a ztratí schopnost generalizace a proto jsme se tímto směrem nevydali.

Hlavním limitem je tedy množství dostupných trénovacích dat, a proto by bylo spíše vhodné nadále zkoumat další techniky augmentace. Nakonec lze dále ladit hyperparametry jako jsou *learning rate*, *batch size* a *weight decay*.

4 Rozpoznávání řeči

Pro úlohu rozpoznávání mluvčích jsme zvolili přístup založený na Gaussových směsových modelech (GMM). Každý mluvčí je reprezentován pomocí několika Gaussových rozdělení, které modelují jeho akustické rysy. Tento přístup se osvědčil zejména pro jednodušší a středně velké úlohy rozpoznávání řeči, a zvláště tam, kde je relativně málo trénovacích dat na třídu, tj. osobu.

4.1 Trénování

I během trénování zvukového modelu byla opět využita výše zmiňovaná technika jackknifing. Jelikož se princip použití nemění, zaměřme se na detaily specifické pro trénování modelů pracujících se zvukem.

4.1.1 Předzpracování dat

Zvukové nahrávky byly nejprve převedeny do spektrálních reprezentací. Použili jsme extrakci MFCC příznaků (*Mel-Frequency Cepstral Coefficients*), které lépe vystihují lidské vnímání zvuku. Parametry extrakce byly následující:

- 25 ms dlouhé rámce,
- 10 ms posun,
- 13 koeficientů.

Rozhodli jsme se vynechat 0. koeficient, který odpovídá celkové energii signálu, která bývá silně ovlivněna proměnlivými podmínkami, jako je vzdálenost mikrofonu nebo šumové pozadí, a proto může negativně ovlivnit extrakci rysů.

Pro zajištění lepšího rozlišení mezi různými mluvčími a stabilnějšího trénování jsme po extrakci MFCC aplikovali techniku *Cepstral Mean and Variance Normalization (CMVN)*. Tato technika minimalizuje zkreslení způsobené šumem tím, že lineárně transformuje kepstrální koeficienty tak, aby měly stejné střední hodnoty a rozptyly napříč segmenty.

4.1.2 Trénování GMM

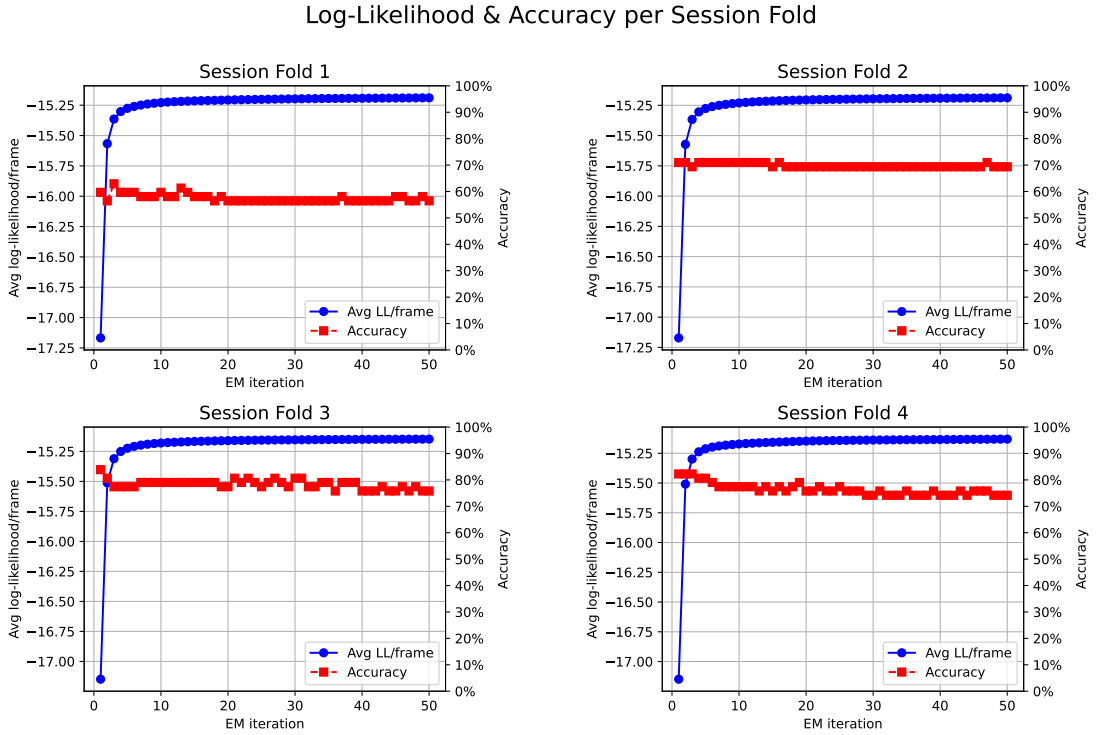
Pro každého mluvčího jsme trénovali samostatný GMM pomocí algoritmu Expectation-Maximization (EM). Pro inicializaci parametrů Gaussova směsového modelu jsme místo náhodného výběru využili klasický algoritmus *k-means*. Důvodem je vyšší stabilita a rychlejší konvergence EM algoritmu. Parametry pro trénování byly následující

- počet Gaussových komponent: 16
- 100 EM iterací

Zvolili jsme 16 komponent kvůli omezenému množství trénovacích dat. Vyšší počet komponent by vedl k přeučení (overfitting).

4.2 Vyhodnocení

Vyhodnocení systému probíhalo na základě přesnosti klasifikace na validačních množinách nahrávek v jednotlivých *fold*ech. Klasifikace probíhá pomocí max log-likelihood – každá testovací nahrávka je přiřazena k tomu mluvčímu, jehož GMM model jí přiřadí nejvyšší pravděpodobnost. Průběh trénování z hlediska nárůstu trénovacího a validačního log-likelihood je znázorněn na Grafu 4.1. Viditelné je počáteční rychlé zlepšení následované stabilizací.



Obrázek 4.1: Graf zobrazující vývoj průměrné logaritmické pravděpodobnosti (modré kruhy, levá osa y) a klasifikační přesnosti (červené čtverce, pravá osa y) v průběhu 16 iterací algoritmu EM (osa x) pro každý fold (1-4). Logaritmická pravděpodobnost v prvních iteracích strmě roste a poté se vyrovná, zatímco přesnost nemá čistě rostoucí tendenci a ustálí se mezi 60 % a 80 % v závislosti na foldu.

Současný systém vykazuje poměrně stabilní chování při trénování a dosažitelná přesnost se pohybuje v rozmezí 60–80 % v závislosti na validačním foldu. Přesto existuje několik oblastí pro potenciální zlepšení:

- **Augmentace dat:** Vytváření syntetických variant původních nahrávek pomocí technik jako přidání šumu, změna rychlosti nebo tónu řeči by mohlo zvýšit variabilitu trénovacích dat a zlepšit robustnost modelu.
- **Nekonstantní volba komponent:** Fixní volba 16 komponent se nezdá být ideální. Adaptivní volba počtu komponent dle složitosti dat jednotlivých mluvčích (např. pomocí *Bayesian information criterion* (BIC)) by mohla zlepšit modelování individuálních charakteristik.
- **Vylepšení předzpracování:** Zvážení alternativních příznaků (např. *perceptual linear prediction* (PLP)) nebo pokročilejších normalizačních technik (např. *feature warping*).
- **Nahrazení GMM modernějšími metodami:** Dobrými kandidáty se zdají deep learning přístupy (např. *i-vector*, *x-vector* s následnou klasifikací), které překonávají GMM v úlohách rozpoznávání mluvčích.