Projekt SUR 2025

Implementace se skládá ze souboru classes.py, kde jsou definice architektur konvolučních sítí, datasetů a klasifikátoru. Funkční část programu je pak v souboru proj.ipynb.

Popis řešení:

K řešení byly použity 2 konvoluční sítě zvlášť pro obrázky a zvuk (třídy FaceCNN a VoiceCNN). Obrázek vchází do konvoluční vrsty tradičně jako hodnoty RGB ve 3 kanálech, zatím co audio se nejdříve převádí na Mel-spektrogram. Tyto 2 konvoluční sítě se následně spojí do 1 vstupu pro klasifikační síť. Vzhledem k malému množství dat trénovací sady, ale existenci obrázku a audia pro každé sezení, jsem se rozhodl trénovací dataset rozšířit dvěma způsoby.

Prvním rozšířením je kombinační výběr. Jelikož pro každou osobu bylo v trénovací sadě 6 obrázků a 6 nahrávek zvuku, místo pouze 6 dvojic ze stejných sezení je vytvořeno všech 36 kombinací zvuků a obrázků. Sama o sobě tato metoda moc efektivní není, protože se stále jedná o duplikáty a jejich unikátnost se projeví až po jejich složení v klasifikační síti. NN ovšem vyžaduje fixní velikost vstupu a nahrávky jsou různě dlouhé. Oba tyto problémy tedy řeším tím, že se z každé nahrávky vždy vykousne pouze náhodný úsek fixní velikosti. Díky tomu každá z oněch zmíněných 36 kombinací má svou vlastní unikátní zvukovou složku.

Druhým typem rozšíření je augmentace obrazových dat. Tedy náhodné otočení, přiblížení a změna jasu a kontrastu každého obrázku. Tím lze výrazně rozšířit trénovací sadu o nová kvalitní data (pokud tedy kvalitní byla i ta původní).

Postup reprodukce výsledků:

Projekt očekává extrahované složky train, dev a eval, tak jak byly poskytnuty v archívu ze zadání, na stejném místě jako proj.ipynb. K vytvoření modelu a evaluaci nových dat následně stačí pouze otevřít Python Notebook a spustit všechny buňky v jejich pořadí.

Výsledný model je uložen do souboru best_model.pth a výsledky evaluace se vypisují přímo v notebooku.

Konkrétní model, ze kterého pochází dodané výsledky, byl vytvořen za použití následujících parametrů:

augmented_dataset -> num_augs=10
train_model -> epochs=20, lr=1e-4

Nutno však podotknout, že model trpí přeučením a vlivem náhodného výběru audia a náhodnými augmentacemi obrazu v době běhu programu, je i se stejnými parametry každý nově trénovaný model jinak úspěšný.

Verze použitéhoho softwaru:

Python	3.11.9
torch	2.7.0+cu118
torchaudio	2.7.0+cu118
torchvision	0.22.0+cu118

Vyhodnocování a techniky

Začal jsem s kombinovaným modelem pro obrázky i audio, 2krát 3vrstvá CNN + klasifikační NN s 1 skrytou vrstvou s 512 neurony a dropout 30%. Model byl přeučený a na validační sadě dosahoval úspěšnosti kolem 43%. Zkusil jsem snížit přeučení přidáním augmentace dat, což sice nepomohlo, ale úspěšnost vzrostla na zhruba 80% při parametrech 5 augmentací na prvek za 5 epoch. Následně jsem našel nejúspěšnější model s 85,48% pro parametry 10 augmentací za 20 epoch. Abych zamezil přeučení, zkusil jsem dále odstranit vrstvy z CNN, změnit dropout, zmenšit výslednou NN na pouze 100 neuronů, ještě zvětšit trénovací sadu přidáním obrázků s gausovským šumem, přidat L2 redukci, ... nepomohlo nic a model vždy skončil s >99% na trénovací sadě, akorát úspěšnost na sadě validační s každou další metodou klesala.

Co jsem dále zkusil a nevyšlo to, bylo podle doporučení ze „zadání“ zahodit první 2 sekundy audia kvůli počátečnímu tichu a ruchu v nahrávce. To bylo neúspěšné pravděpodobně kvůli tomu, že se prakticky jednalo o vestavěnou augmentaci, taková nekvalitní data, která model při učení, podobně jako třeba dropout, přiměla se nefixovat na něco konkrétního a posunula ho jiným směrem

Při dalším vývoji by to asi chtělo větší zásah do architektury sítě, zejména obrázkové části, která se velmi snadno přeučí. V tomto ohledu nemám žádné zkušenosti, takže jsem sice zkoušel nějaké techniky, které přeučení mají potlačovat, ale zřejmě to chtělo ještě razantnější změny, protože jsem ničeho nedosáhl.