SUPA3 REPORT - [B20AI013]

A report for the Speech Understanding Programming Assignment 3, by Ishaan Shrivastava [B20AI013]

Ishaan Shrivastava

1. **GitHub Repository** - https://github.com/shrivastava95/supa3
2. **Assignment Report + WandB logs** - This file itself
3. **Gradio link** - https://github.com/shrivastava95/supa3/tree/main/gradio_demo

Task 1 - Loading SSL_W2V pretrained model

First, I have followed the instructions in the SSL_Anti-spoofing repository (link - https://github.com/TakHemlata/SSL_Anti-spoofing/) and created a conda environment in python 3.7 and installed the various dependencies.

After this, I have downloaded the dataset and model checkpoint files involved and set them up in various places.

Task 2 - Custom Dataset Results

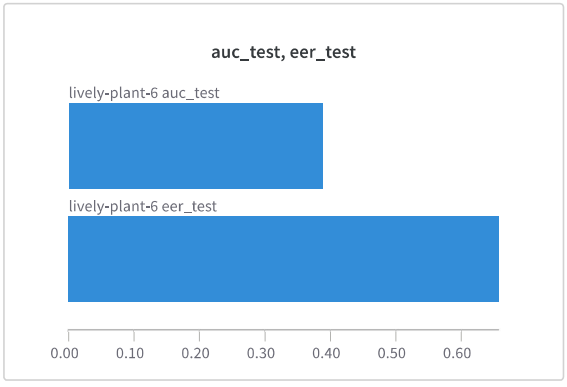
I have downloaded the custom dataset uploaded by Rishabh. There were roughly 300 files inside it, with a 120-180 split of samples.

Using the loaded SSL-W2V model, I obtained the following results:

AUC (pretrained model - custom dataset) - 0.3889

EER (pretrained model - custom dataset) - 0.6580

Note that I am using this model as the testing set for my experiments.

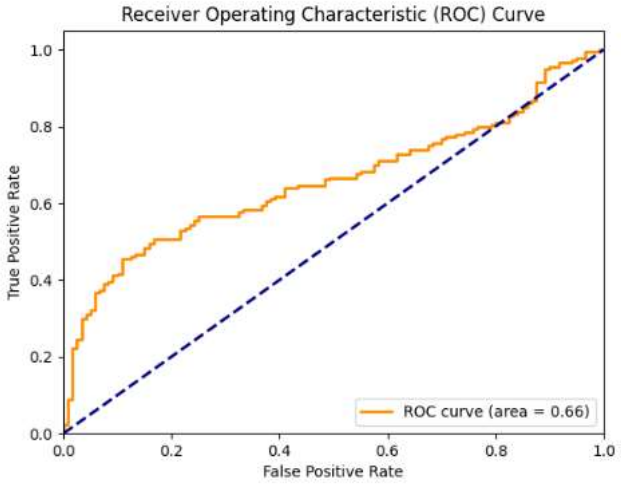


Import panel Add panel

Task 3 - Analyse the performance of the model

As seen in the matplotlib plot below, I have plotted the ROC AUC curve for the pretrained model on the custom dataset.

The curve is skewed towards the TPR (sensitivity) axis. This implies that the model is able to achieve a good recall without needing to take too many negatives.



Task 4 - Finetune the model on FOR dataset

I downloaded the dataset from https://www.eecs.yorku.ca/~bil/Datasets/for-2sec.tar.gz

The dataset contains *"Fake"* and *"Real"* audio files where the *"Fake"* audios are generated with the objective of spoofing a voice model.

Let us finetune our SSL_W2V model on this dataset and see how the performance changes.

As we can see below, the blue lines correspond to the `eval` set and the yellow lines to the `train` set. The dashed lines measure AUC while the smooth lines measure EER.

The training is done on the train split of the for-dataset, and the eval is done on the validation split of the for-dataset.

Config:

1. learning rate - 3e-4
2. num epochs - 2
3. batch size - 32
4. criterion - binary cross entropy classification loss

5. optimizer - adam

Results on FOR Eval dataset:

EPOCH 0 - before training

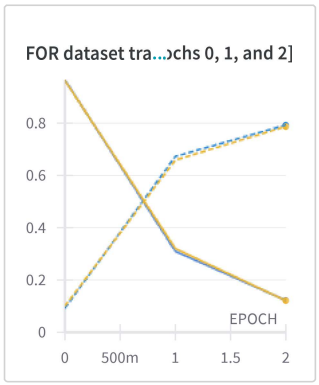
AUC - 0.092

EER - 0.9651

EPOCH 2 - finetuned

AUC - 0.7923

EER - 0.1220



Import panel Add panel

Task 5 - Evaluating finetuned model on Custom Dataset

I have also used the Custom Dataset provided by Rishabh as a testing dataset across the training.

- cyan - Custom Dataset [test set]
- yellow - FOR - validation [eval set]
- magenta - FOR - training [train set]

I have also used the Custom Dataset provided by Rishabh as a testing dataset across the training.

The test scores on Custom Dataset across epochs 0 and the last epoch 2 are as follows:

Results on Custom Dataset:

EPOCH 0 - before training

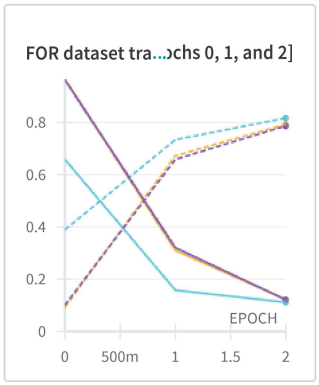
AUC - 0.3889

EER - 0.6580

EPOCH 2 - finetuned

AUC - 0.7863

EER - 0.1211



Import panel Add panel

Task 6 - Comment on change in performance

It can be seen that the FOR-dataset initially had worse performances on its eval set before training, compared to the results obtained on the custom dataset given. This could be due to the more robust nature of construction of the FOR-dataset which makes it a better candidate for training and evaluation of deepfake detection datasets. Additionally, the FOR dataset is larger which means that the evaluations run on it will be more reliable and the training will benefit from scale.

After training, it can be seen that the EER and AUC of the trained model on both Custom and FOR datasets are basically quite similar to each other and FOR eval scores have caught up. This is because of the similarity between the training and eval distributions in the FOR dataset compared to that of the custom dataset.