Temporal Modeling Matters: A Novel Temporal Emotional Modeling Approach for Speech Emotion Recognition (TIM-Net)

Description

Platform

I have trained and tested the code on an *OSX-ARM* Architecture in a Python 3.8 *miniforge*-based environment.

Datasets

- TIM-Net was tested on 6 datasets: CASIA, SAVEE, EMODB, RAVDESS, IEMOCAP, and EMOVO.
- Raw Dataset and pre-processed MFCCs are available here.
- Raw IEMOCAP dataset is not provided in the drive link due to its large size (24 GB)
- Raw CASIA dataset wasn't publicly available, hence isn't provided. The author has provided the MFCCs for the same, which were used as is.
- Most datasets were very small and the model easily overfitted on them. Due to the low number of samples per emotion in the datasets, the author hasn't created a separate testing data split.

Code

This Github Repository contains my **modified** code for the paper. The README.md has instructions to run the code.

Implementation and Improvement Details

- **Environment Setup**: The provided package requirements weren't getting imported directly as a new environment due to internal conflicts. Hence, I fixed the package versions for the environment and have provided a *requirements.txt* file.
- **Dataset Organisation**: The provided pre-processing code was incomplete. I have added code to sort the dataset into specific folders based on emotions.
- **Training Efficiency Improvements**: The training script ran very slow on my device. Hence, I have added custom model checkpointing constraints to minimize overfitting and increase training speed. We can now customise how often model is checked for improvements and when to save checkpoints.
- **Hyper-parameter Tuning**: The author's Github repository included updated results, after the paper publication. These addressed some overfitting concerns causing the accuracies to drop. Furthermore, the new results corresponding to 2 datasets (EMOVO, IEMOCAP) were not provided. I have carried out the training for all the cases and have reported the results in Table 1.
- **Training Progress Visualisation**: I have implemented a new feature to visualise the training progress. This includes plots for accuracy and loss plots averaged over the number of splits as illustrated in Table 2.
- **t-SNE Visualisation**: I have created a script to visualise the t-SNE plots of the weights of the penultimate layer. The study at the end of the paper provides results only for 1 dataset, I have added the corresponding results for all 6 datasets in Table 2.

Results

Following validation accuracies are based on the hyperparameters chosen to optimise local training on my laptop. The WAR was calculated separately based on class distribution weightages in the dataset.

Table 1: UAR / WAR of the TIM-Net on the 6 datasets

Data	My Accuracy	Github Repo Accuracy	Paper Accuracy
SAVEE*	0.7415 / 0.7667	0.7726 / 0.7936	0.8607 / 0.8771
EMODB	0.8377 / 0.8523	0.8919 / 0.9028	0.9517 / 0.9570
RAVDESS	0.8412 / 0.8451	0.9004 / 0.9007	0.9193 / 0.9208
CASIA	0.8553 / 0.8608	0.9108 / 0.9108	0.9467 / 0.9467
EMOVO	0.8046 / 0.8044	-	0.9200 / 0.9200
IEMOCAP**	0.4125 / 0.4395	-	0.7250 / 0.7165

Note:

* The major focus of hyperparameter tuning was on SAVEE as it offered a balance between number of samples and classes during exprimentation.

** IEMOCAP was very large causing 1 epoch to run for a long time (> 1.5 min) on my local machine. With 10 splits, it would take around 1 week for training and hence reduced parameters were used for training. The accuracy can be further improved by running more epochs (> 500) on larger number of split-folds (> 8) in a decent GPU environment. I had attempted to train on a NVIDIA GPU on kaggle and the lab GPU however setting up older version of Python and Tesnorflow libraries resulted in some issue which couldn't be solved in a limited time frame.

Table 2: Visualised Results of the proposed TIM-Net on 6 datasets