FEDERATED LEARNING FOR EMOTION RECOGNITION

Ivan Chau — ic2504@columbia.edu

Daniel Garces — dg3008@columbia.edu

ABSTRACT

Emotion recognition in speech tends to be a very difficult task due to variations in pitch and intonation across different individuals when expressing emotions verbally. Current AI systems try tackling this issue through supervised approaches, but the lack of high-resource data for emotion recognition makes it hard for these approaches to generalize well to real-life data. To tackle this issue, a distributed emotion recognition system that is constantly being improved by user interaction could be considered. This systems will lead to more robust emotion recognition models that address the issue of the variations in speech patterns across users by combining multiple models from the clients to generate a combined centralized model that contains features from all of them, while providing a stream of new data that is directly digested locally on each device and later sent to the centralized server as part of the resulting model (addressing the issue of data scarcity for emotion recognition). In this paper, we want to explore this possibility by implementing a federated learning system for emotion recognition in speech that will adapt to a specific edge devices' user while also contributing to the construction of a shared global model that will provide a more accurate starting point for new users (as it contains the combined information of all the users utilizing the system). To do that, we used the Tensorflow implementation of federation to create a custom federated learning system that process speech in different clients and then aggregates the models into a combined model being run in a centralized server. We hope that by combining fine-tuned models from different users we can generate a model with better predictive accuracy that is more robust towards intonation and pitch variations across users. Our results illustrate that this kind of federated systems could be helpful for generating more robust models with higher accuracy on the task of speech emotion recognition.

Index Terms— Emotion recognition, Deep Learning, Federated Learning

1. INTRODUCTION

The expansion of deep learning and the development of more efficient neural network architectures have brought renewed interest to AI powered systems that could perform previously intractable tasks, including speech generation, speech recognition, and language understanding. The exploration of these fields has revealed the need for better human-computer interfaces that integrate such systems to boost human performance in other disciplines, like medicine [1]. Central to the development of these improved human-computer interfaces is affective computing, which refers to the study of systems that can model, interpret and simulate human affects (feelings or emotions). Emotions play a very important role in our daily lives as humans, as they permeate our thoughts, words, and even our actions. Considering this, emotion can be expressed (and therefore detected) through facial expressions, gestures, speech, and the semantic content of conversations and interactions with other humans and objects in our surroundings. Even though perceiving and identifying emotions might not be such a difficult task for a human, it remains a challenge for machines [2]. Recently, there have been multiple efforts to generate an emotion recognition model that efficiently and accurately classifies emotion using speech, text, and even images[3]. However, these models tend to suffer from low accuracy due to variations across individuals on the ways they express emotions through gestures (facial recognition), intonation and pitch (speech recognition), and word choice (text recognition). Particularly in speech, there are very few labeled data-sets for emotion recognition, so supervised approaches tend to not generalize well to real-life data. A solution to this problem could be a system that is constantly improved by user interaction, so that the resulting model is more robust towards variations in speech patterns, improving the overall accuracy on the classification task.

The emergence of edge devices and the development of faster and more reliable processors for edge devices have allowed these systems to integrate AI algorithms into their frameworks and software to perform predictive and classification tasks locally on-device. This new development brought forward the possibility of utilizing distributed models across edge devices to decrease training time and generate combined models that merge multiple features from the edge device's trained models. In this paper, we want to explore this possibility by implementing a federated learning system for emotion recognition in speech that will adapt to a specific edge devices' user while also contributing to the construction of a shared global model that will provide a more accurate starting point for new users (as it contains the combined information of all the users utilizing the system). The users will be simulated for simplicity, but as part of a future work the system will be expanded to allow real-life users to input data, self-classify it, and see the performance of the model change in real-time.

2. BACKGROUND

To help the reader understand the proposed solution, we will briefly cover the following concepts:

2.1. Mel Scale

Mel scale is a scale that relates the perceived frequency of a tone to the actual measured frequency. It scales the frequency in order to match more closely what the human ear can hear (humans are better at identifying small changes in speech at lower frequencies). This scale has been derived from sets of experiments on human subjects. The basic formula relating standard frequency (in Hz) to Mel's frequency is shown below in Figure 1

$$\mathrm{Mel}(f) = 2595 \log\left(1 + \frac{f}{700}\right)$$

Fig. 1. Formula for Mel Scale using normal frequency (in Hz)

2.2. Mel Frequency Cepstral Coefficients

Any sound generated by humans is determined by the shape of their vocal tract (including tongue, teeth, etc). If this shape can be determined correctly, any sound produced can be accurately represented. The envelope of the time power spectrum of the speech signal is representative of the vocal tract and MFCC (which is nothing but the coefficients that make up the Mel-frequency cepstrum) accurately represents this envelope. The normal procedure for extracting MFCCs is depicted below in Figure 2. In our project, the filter bank is composed of 13 linear filters and 27 logarithmic filters. As part of our audio preprocessing step, we split each audio file in short-term windows (0.2 seconds) and we extract the MFCCs for each window. These features (represented as vectors of doubles) are later stacked together to get the representation of the entire signal.

2.3. LSTM: Long-Short Term Memory

The Long-Short Term Memory architecture is intended to serve as a recurrent neural network. The diagram shows above one particular LSTM cell. This cell contains input, output, and forget gates, which allow the cell to maintain and

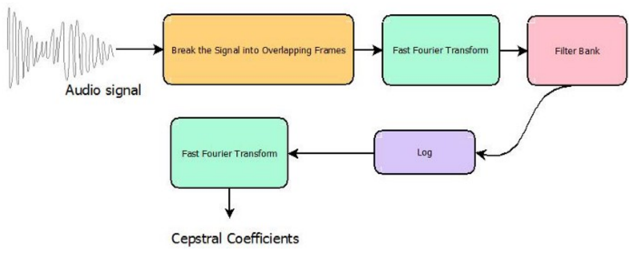


Fig. 2. Process for Extracting MFCCs

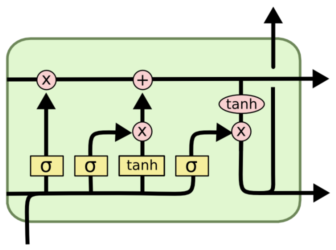


Fig. 3. LSTM Diagram

update a "cell state." The main application of these recurrent neural architectures for audio sequences is so previous information about utterances can persist and aid in making more educated predictions. In our project, we use an LSTM architecture in one of our models.

2.4. Federated Learning

The goal of Federated Learning is to move training onto edge devices, using their gradient updates to build a centralized model. This allows for many benefits, which include data privacy, scalability, and even less cost. However, Federated Learning is distributed computing on an extreme scale, and of course there are many problems – mainly, how to achieve the performance and accuracy of a centralized model. This project attempts to find optimal parameters so that we can converge, or even outperform the centralized models. In practice, setting up a real-life FL experiment could be costly, but simulations can emulate what an environment would look like with some computing constraints – this is the approach our project takes.

3. PREVIOUS WORK

A considerable research has been done in the field of Emotion Recognition in Speech. The following are some of the most recent research projects that had a direct influence in the development and implementation of our idea and experimental setup:

3.1. Ananthram et. al

In this paper, the authors attempted a multi-modal approach to Emotion Recognition using embeddings from both Speech and Text as input features. First, they trained a Time delayed neural network(TDNN) on the speech utterances of the VoxCeleb corpus[1]. Then the TDNN is fine-tuned on the speech utterances of the Crema-D corpus[2]. The text-transcripts of the speech utterances are used to fine-tune a pre-trained English BERT model. The embeddings from the 6th, 7th and 8th layer of TDNN (Speech Engine) are concatenated with the embeddings obtained from BERT (Text Engine) and are fed into an LDA-pLDA classifier inspired in the original implementation of VoxCeleb [4], which scores each pair of utterance according to how related they are (whether they were generated by the same speaker).

3.2. Goel et. al

In this paper, the authors attempted a cross lingual approach to Speech Emotion Recognition using Transfer Learning. They took an emotion-labeled dataset of speech recordings in 4 different languages- German, English, Italian and Mandarin.The size of datasets of English and Mandarin was large (high resource), whereas the size of the dataset of German and Italian was small (low resource).Two experiments were conducted independently. In the first experiment, speech emotion recognition was done on the datasets individually for each language using a TDNN architecture [5]. In the second experiment, Transfer Learning was utilized to leverage the knowledge contained in the original TDNN trained for high-resource languages (English and Mandarin) as a feature extractor. This feature extractor was later utilized as the first section of a classifier fine-tuned on the low-resource data. The results of the paper illustrated that Transfer Learning provided higher accuracy for the lower resource languages than a TDNN trained directly on the low-resource data, which shows the benefits of transfer learning in the realm of emotion recognition.

3.3. Zhou et. al

This paper proposes a method to apply transfer learning for speech emotion recognition from automatic speech recognition (ASR) using a TDNN [6]. The method utilizes IEMOCAP for both training and testing the final model. The architecture is based on the TEDLIUM release 2 recipe, which means that the original model for ASR was trained on the TEDLIUM2 dataset. The full MFCC features with all 40 coefficients are utilized for all the speech processing tasks. The TDNN has a total of 13 tdnn layers, and the 12th layer is utilized for extracting the emotion embeddings for the transfer learning portion. The embeddings obtained from this layer are fed into a dense layer connected to an output softmax layer. The accuracy for the model is measured using this softmax layer after the model is fine-tune on 4/5 of the IEMOCAP data (4 out of the 5 sessions). The last session in IEMOCAP is used for evaluation.

3.4. Tripathi et. al

This paper proposes a method for multimodal emotion classification utilizing the audio, transcriptions, and motion capture information contained in IEMOCAP. The authors of this paper considered different neural network architectures to evaluate how each one performs on the multimodal emotion recognition task. Their idea is to train three models independently and then extract the embeddings before the softmax layer, concatenate them and feed them into a dense layer that is directly connected to another softmax layer for evaluation. For speech, text, and motion capture, the authors consider three main architectures: a fully connected network with three layered fully connected MLP layers [7], a recurrent neural network with two stacked LSTM layers, and a convolutional neural network with 4 convolutional layers. This paper also utilized a 4/5 split of the IEMOCAP dataset for training and the rest of the data for testing. Some of the architectures that we consider in our paper were derived from the architectures proposed in this paper.

4. PROPOSED SYSTEM

4.1. Details

Our approach to designing these experiments is creating a Federated Learning training module and data pre-processing pipeline. To do this, we used Tensorflow Federated as well as Python.

With this infrastructure in place, we began populating the data for our simulations – namely, partitioning our pre-processed data into several different distributions for our clients. In our paper, we discuss different distribution options discussed below, and write code for the processing of each of these.

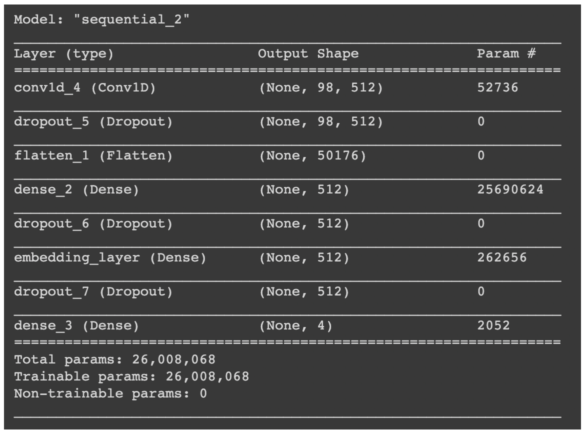
Finally, we begin our simulations by providing a number of hyper-parameters for the system, which include the model to train, the number of clients, and the number of rounds.

4.2. Architecture

For architecture, we mainly want to test how different, broad styles of deep learning impact the task at hand.

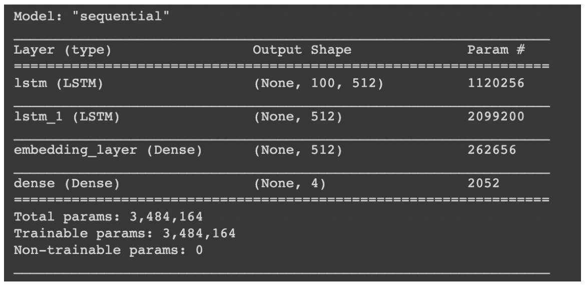
[1]an audio-visual dataset consisting of short clips, extracted from interview videos uploaded to YouTube. It has over 2000 hours of recording of over 7000 speakers.

[2]an audio-visual dataset labeled with emotions like happy, sad, angry, fear, disgust and neutral. It has 7442 clips of 91 actors from diverse ethnic background.



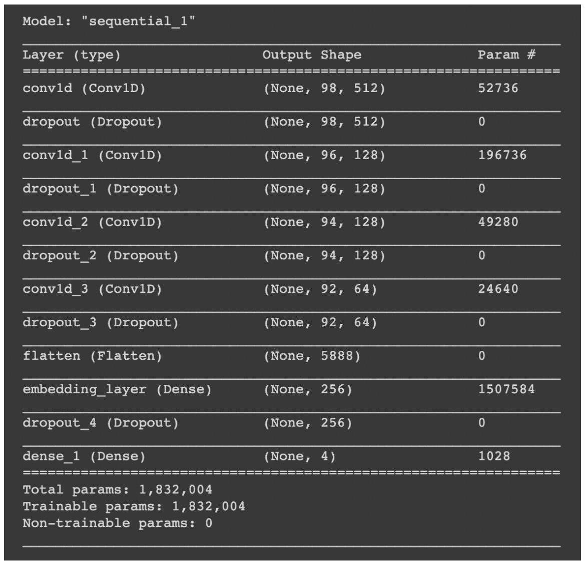
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_4 (Conv1D)            (None, 98, 512)           52736
_________________________________________________________________
dropout_5 (Dropout)          (None, 98, 512)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               25690624
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
embedding_layer (Dense)      (None, 512)               262656
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 2052
=================================================================
Total params: 26,008,068
Trainable params: 26,008,068
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4. Dense Summary



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 512)          1120256
_________________________________________________________________
lstm_1 (LSTM)                (None, 512)               2099200
_________________________________________________________________
embedding_layer (Dense)      (None, 512)               262656
_________________________________________________________________
dense (Dense)                (None, 4)                 2052
=================================================================
Total params: 3,484,164
Trainable params: 3,484,164
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5. LSTM Summary



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 98, 512)           52736
_________________________________________________________________
dropout (Dropout)            (None, 98, 512)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 96, 128)           196736
_________________________________________________________________
dropout_1 (Dropout)          (None, 96, 128)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 94, 128)           49280
_________________________________________________________________
dropout_2 (Dropout)          (None, 94, 128)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 92, 64)            24640
_________________________________________________________________
dropout_3 (Dropout)          (None, 92, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 5888)              0
_________________________________________________________________
embedding_layer (Dense)      (None, 256)               1507584
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 1,832,004
Trainable params: 1,832,004
Non-trainable params: 0
_________________________________________________________________
```

Fig. 6. Conv Architecture Summary

Our first model is a deep neural network (Figure 4), which we note has the largest amount of parameters (26 million).

Our second model is a convolutional neural network (Fig 6), which has significantly less parameters (2 million), but takes advantage of temporal locality within our data.

Our last model is a LSTM nueral network (Fig 5), which also has a small number of parameters (3 million), but also takes advantage of state and allows us to learn how data acts temporally.

4.3. Implementation and Challenges

We attempted to run our experiments on Google Colab, which provides both access to CPU and GPU.

All models were created in Tensorflow/Keras, which is compatible with Tensorflow Federated. In our Federated Learning (FL) setup, we used an out-of-the-box optimization helper, *Federated Averaging*, as well as Adam optimizers with different learning rates for the client and server. In addition, we wrote preprocessing and data generating functions (per client) from scratch for our experiments. These data generating functions allow us to split the one epoch of the dataset amongst an arbitrary number of clients, with a specified distribution (see below).

In running our experiments, we found that there was a technical conflict with our Tensorflow Federated library and the Tensorflow GPU library we were using, as well as RAM issues on the GPU instance offered. In our final experiments, we ended up using the CPU with 16 GB of RAM for our Federated Learning procedures, and the V100 NVIDIA GPU to train our baselines.

To discuss fair comparisons for baselines, we decided to use the F1-metrics and accuracy scores, as well as time to completion of our fully converged, GPU trained models, as compared to a Federated Learning model trained with 4 clients and 20 rounds with a batch-size of 20. We used the metrics offered by scikit-learn and Keras to produce the data on our testing set.

5. EXPERIMENTS

5.1. Experiment 1: Baseline and FL comparisons

5.1.1. Differences in Client Distributions

Using our implemented code, we test the non-i.i.d problem within FL by purposely manipulating client data distributions. For example, in one of our Federated Learning systems, we made our four clients have completely random data of all emotions and speakers. This will simulate the ideal situation, and emulates the natural noise someone might have when having data on their own device. The other FL system will have a perfectly skewed distribution towards a single class (each client will have access to a single emotion), which is a situation in which there is little noise in observations. We want to see if this has an effect on convergence and accuracy. We train each architecture for each setup for 20 epochs (or 20 rounds for the federated learning system), and then record the training time, and the accuracy, precision, recall, and F1 score on the testing set.

5.1.2. *Metrics*

Accuracies and F1 Scores: The goal of this metric is to see if FL can get any reasonable scores with our current setup in comparison to the baseline models. The hope is that FL proves to be similar, to baselines, and if this depends on the distribution of data amongst the clients.

5.1.3. *Time to Completion*

In practice, this metric is not as important, because FL is certainly going to be worse in the real world (in the simulation, everything is centralized, but in reality, edge devices are locally far away and dependent on slow networks to communicate data). However, we find that it may be interesting to measure if Federated Learning suffers from time issues even when centralized, and if this can speak to anything regarding model architecture, data distribution, etc.

5.2. Experiment 2: Number of Clients in FL System

As we mentioned above, we can use our data-generating functions for any arbitrary number of clients, not only k = 4. In this case, we would like to find if having different numbers of clients makes an impact on our metrics in Experiment 1. In our experiment, we run our simulation for k = 1 ... 8, with the random/i.i.d client distributions on the Deep and Conv models for 10 rounds, and compare the F1 scores as our main metric.

6. RESULTS

6.1. Experiment 1

Through this experiment we compare the different metrics mentioned in Section 5.1 between the baseline models (models trained directly on IEMOCAP without any federation), the models with randomly assigned data (models trained with data randomly assigned to each of the 4 clients), and the models with a single emotion per client (models in which clients were trained on a single emotion). All the results for this experiment are compiled below in three tables, each for a single type of architecture. Table 1 contains the results for the Deep architecture for all three different systems, baseline without federation, federated learning with randomly distributed data among clients, and federated learning with a single emotion per client. Table 2 contains the results for the Convolutional architecture, while Table 3 contains the results for the LSTM architecture. The confusion matrix for each model can be found in the Appendix at the end of this paper.

As shown in the tables, the federated system with the random assignment of data to each client resulted in the highest accuracy, recall, precision and F1 score. This illustrates the benefit of simulating real-users on the performance of the overall system, instead of just artificially splitting the data according to emotion labels (as the federated system with a single emotion for each client is the one that performed the worst). The poor performance of the single emotion model can be justified as each client is responsible for identifying a single emotion, which is a harder task to do without having other data values to contrast against. This situation leads to a worse client models and therefore a biased and inaccurate central model during evaluation. During our experiments, most federated models trained faster than the baseline even if the individual client models were trained for the same number of epochs since each client model has a subset of the data and the simulation is running locally so the accumulation and weighted aggregation of each client's weights is not affected by long communication times that would be present in a real-life scenario.

6.2. Experiment 2

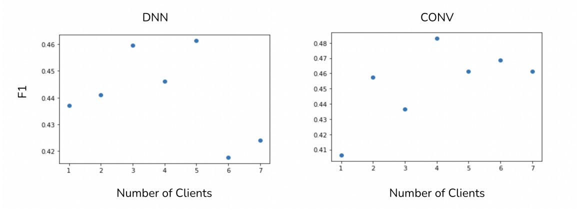


Fig. 7. Effect of # of Clients on F1 Score

In this experiment, we find that there is actually a benefit to including more clients and a more evenly distributed dataset amongst clients, but that benefit actually decreases over time (DNN for 6 and 7 clients seems to degrade performance). This suggests a trade-off between the number of clients, and the amount of data each client communicates. It seems when singular clients communicate too much data, this may not bode well for optimization algorithms, whereas when clients communicate too little data, there may not be enough information for the client models to learn. Therefore, there may be a sweet spot to consider when developing these FL models.

7. FUTURE WORK

As future expansions of this work we could consider running more simulations where data is partitioned across different speakers and each client contains exclusively data for a single speaker. This was not possible for this iteration of the

Metric	Baseline	Random Assignment	Single Emotion
Accuracy	26.00%	51.00%	21.00%
Recall	23.00%	49.00%	25.00%
Precision	23.00%	51.00%	5.00%
F1 Score	0.22	0.49	0.09
Training Time (in sec)	480	627	480

Table 1. Results for dense architecture under the different experimental setups

Metric	Baseline	Random Assignment	Single Emotion
Accuracy	32.00%	41.00%	23.00%
Recall	25.00%	38.00%	25.00%
Precision	24.00%	41.00%	6.00%
F1 Score	0.16	0.37	0.09
Training Time (in sec)	460	316	277

Table 2. Results for convolutional architecture under the different experimental setups

Metric	Baseline	Random Assignment	Single Emotion
Accuracy	41.00%	43.00%	22.00%
Recall	35.00%	40.00%	25.00%
Precision	44.00%	43.00%	6.00%
F1 Score	0.31	0.35	0.09
Training Time (in sec)	5740	5407	4645

Table 3. Results for LSTM architecture under the different experimental setups

project as there are very few utterances for each speaker, so the models will not perform well if only such few number of utterances were used for supervised classification training. Another future expansion could be the inclusion of a testing module where the emotion classification happens in real-time and the user has the opportunity to qualify the classification so the system could use these labels to perform supervised learning online. We could also extend the work to a multi-modal setting, where both speech and text are considered so that the system could be extended to other languages where intonation denotes meaning rather than emotion (like in Mandarin).

8. CONCLUSION

Overall, we seem to have reached our goal in exploring the several questions and problems in Federated Learning we posit in our introduction. We have shown that Federated Learning models are not reduced in capacity and can meet or exceed the models trained on a centralized dataset (given the most ideal conditions). In addition, we found that Federated Learning is very sensitive to the number of clients and the data distribution for those clients. We find that approaching I.I.D generally helps performance, and that there are optimal settings for the number of clients, which may depend on specific hyperparameters (more clients does not necessarily mean better, etc). Finally, we raise some of the issues we experienced when developing this project for Federated Learning practitioners: Be wary of thinking too much about simulations, as Federated Learning in practice will be much more difficult. Think about your neural architectures carefully as well as their capacities and time to converge. Secondly, be mindful of compute bottlenecks when computing simulations, as they will be very different from your standard singular deep learning training.

9. REFERENCES

[1] Rajib Rana, “Poster: Context-driven mood mining,” *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion*, 2019.

[2] Feiyang Chen, Ziqian Luo, Yanyan Xu, , and Dengfeng Ke, “Complementary fusion of multi-features and multimodalities in sentiment analysis,” *EasyChair*, 2019.

[3] Abhinav Dhall, OV Ramana Murthy, Roland Goecke, Jyoti Joshi, and Tom Gedeon, “Video and image based emotion recognition challenges in the wild: Emotiw 2015,” *Proceedings of the 2015 ACM on international conference on multimodal interaction*, 2015.

[4] Amith Ananthram, Kailash Karthik Saravanakumar, Jessica Huynh, and Homayoon Beigi, “Multi-modal emo-

tion detection with transfer learning," *arXiv:2011.07065*, November 2020.

[5] Shivali Goel and Homayoon Beigi, "Cross-lingual cross-corpus speech emotion recognition," *arXiv:2003.07996*, March 2020.

[6] Sitong Zhou and Homayoon Beigi, "A transfer learning method for speech emotion recognition from automatic speech recognition," *arXiv:2008.02863*, August 2020.

[7] Samarth Tripathi, Sarthak Tripathi, and Homayoon Beigi, "Multi-modal emotion recognition on iemocap dataset using deep learning," *arXiv:1804.05788*, 2019.

10. APPENDIX

```
[INFO] evaluating network...
              precision    recall  f1-score   support

         ang       0.30      0.06      0.11       109
         exc       0.20      0.01      0.02       104
         neu       0.33      0.88      0.48       168
         sad       0.14      0.03      0.04       113

    accuracy                           0.32       494
   macro avg       0.24      0.25      0.16       494
weighted avg       0.26      0.32      0.20       494
```

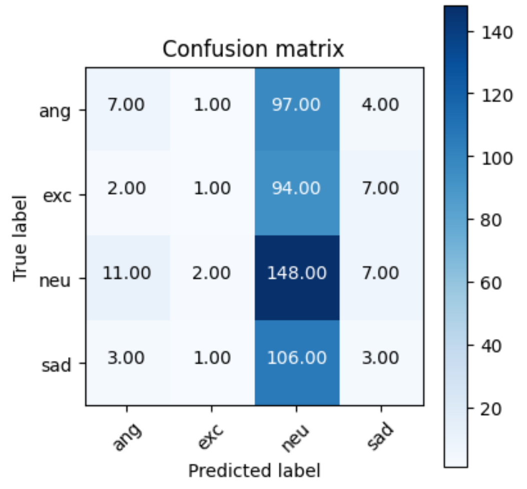


Fig. 8. Confusion Matrix for baseline convolutional model

```
[INFO] evaluating network...
              precision    recall  f1-score   support

         ang       0.21      0.26      0.23       109
         exc       0.20      0.12      0.15       104
         neu       0.34      0.45      0.38       168
         sad       0.17      0.12      0.14       113

    accuracy                           0.26       494
   macro avg       0.23      0.23      0.22       494
weighted avg       0.24      0.26      0.24       494
```

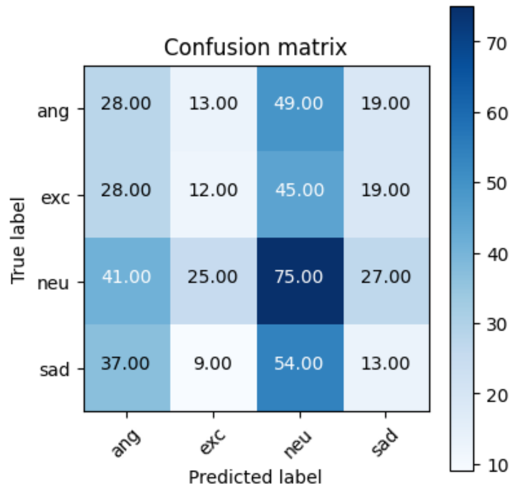


Fig. 9. Confusion Matrix for baseline deep model

```
[INFO] evaluating network...
              precision    recall  f1-score   support

         ang       0.45      0.40      0.43       109
         exc       0.42      0.05      0.09       104
         neu       0.39      0.81      0.52       168
         sad       0.50      0.14      0.22       113

    accuracy                           0.41       494
   macro avg       0.44      0.35      0.31       494
weighted avg       0.43      0.41      0.34       494
```

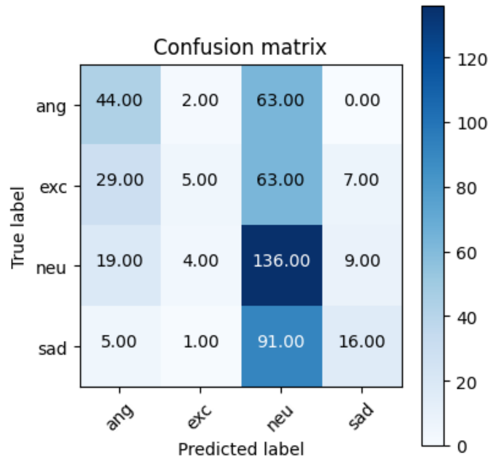
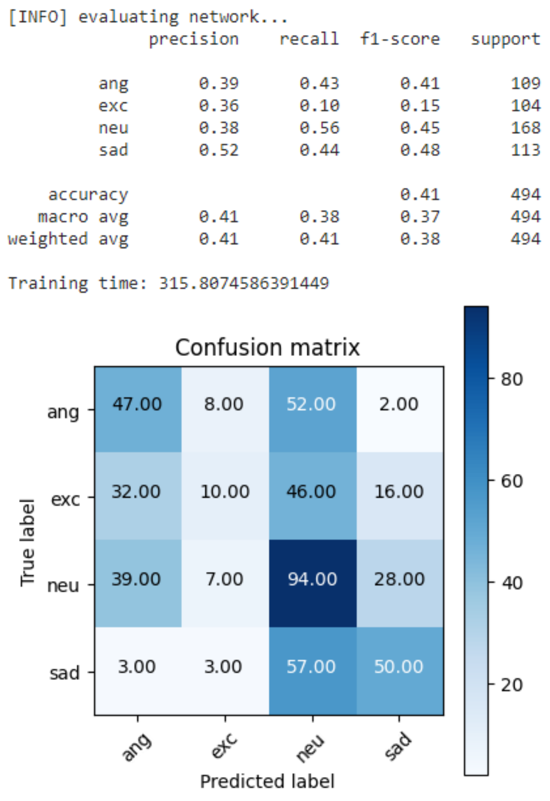


Fig. 10. Confusion Matrix for baseline LSTM model



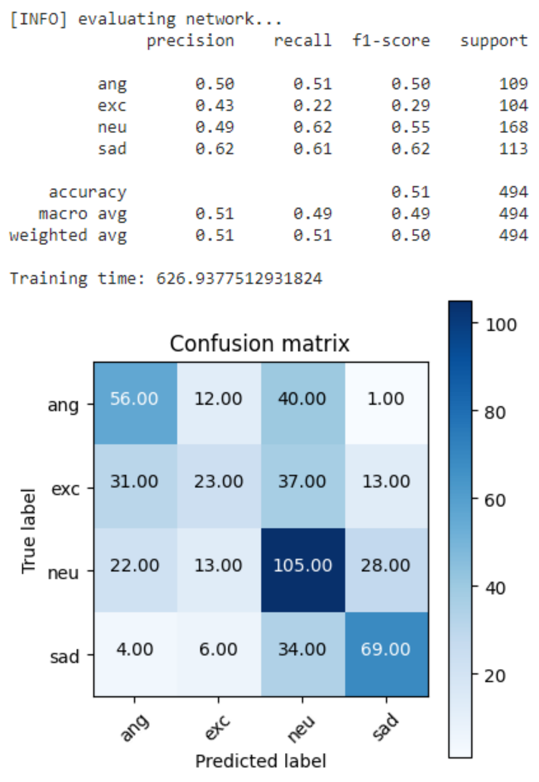
```
[INFO] evaluating network...
              precision    recall  f1-score   support

         ang       0.39      0.43      0.41       109
         exc       0.36      0.10      0.15       104
         neu       0.38      0.56      0.45       168
         sad       0.52      0.44      0.48       113

    accuracy                           0.41       494
   macro avg       0.41      0.38      0.37       494
weighted avg       0.41      0.41      0.38       494

Training time: 315.8074586391449
```

Fig. 11. Confusion Matrix for convolutional model with randomly distributed data



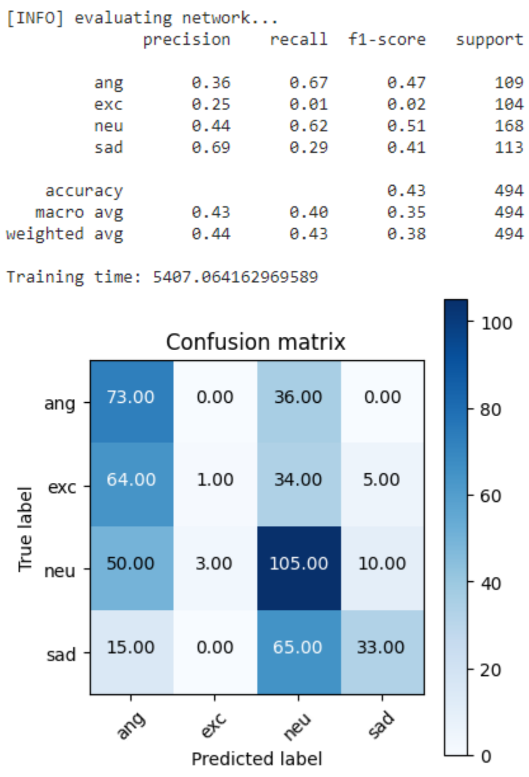
```
[INFO] evaluating network...
              precision    recall  f1-score   support

         ang       0.50      0.51      0.50       109
         exc       0.43      0.22      0.29       104
         neu       0.49      0.62      0.55       168
         sad       0.62      0.61      0.62       113

    accuracy                           0.51       494
   macro avg       0.51      0.49      0.49       494
weighted avg       0.51      0.51      0.50       494

Training time: 626.9377512931824
```

Fig. 12. Confusion Matrix for deep model with randomly distributed data



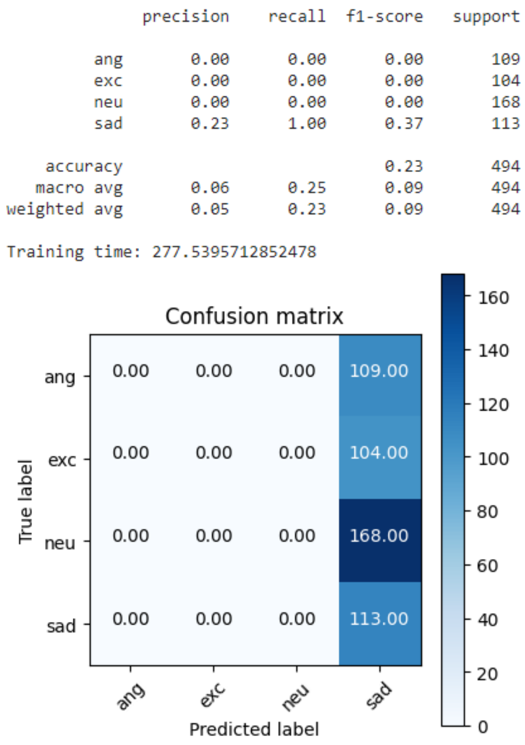
```
[INFO] evaluating network...
              precision    recall  f1-score   support

         ang       0.36      0.67      0.47       109
         exc       0.25      0.01      0.02       104
         neu       0.44      0.62      0.51       168
         sad       0.69      0.29      0.41       113

    accuracy                           0.43       494
   macro avg       0.43      0.40      0.35       494
weighted avg       0.44      0.43      0.38       494

Training time: 5407.064162969589
```

Fig. 13. Confusion Matrix for LSTM model with randomly distributed data



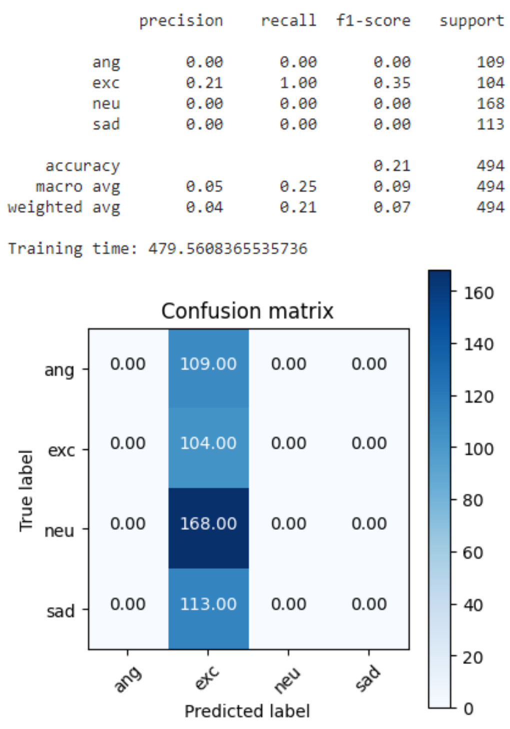
```
              precision    recall  f1-score   support

         ang       0.00      0.00      0.00       109
         exc       0.00      0.00      0.00       104
         neu       0.00      0.00      0.00       168
         sad       0.23      1.00      0.37       113

    accuracy                           0.23       494
   macro avg       0.06      0.25      0.09       494
weighted avg       0.05      0.23      0.09       494

Training time: 277.5395712852478
```

Fig. 14. Confusion Matrix for convolutional model with a single emotion per client



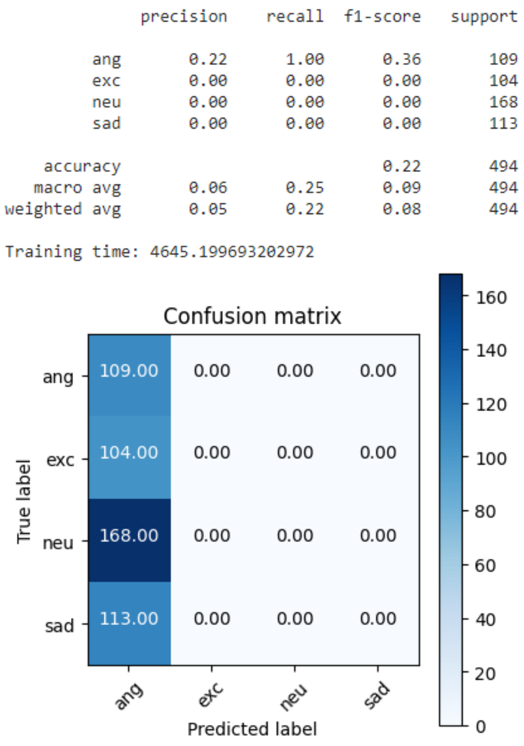
```
              precision    recall  f1-score   support

         ang       0.00      0.00      0.00       109
         exc       0.21      1.00      0.35       104
         neu       0.00      0.00      0.00       168
         sad       0.00      0.00      0.00       113

    accuracy                           0.21       494
   macro avg       0.05      0.25      0.09       494
weighted avg       0.04      0.21      0.07       494

Training time: 479.5608365535736
```

Fig. 15. Confusion Matrix for deep model with a single emotion per client



```
              precision    recall  f1-score   support

         ang       0.22      1.00      0.36       109
         exc       0.00      0.00      0.00       104
         neu       0.00      0.00      0.00       168
         sad       0.00      0.00      0.00       113

    accuracy                           0.22       494
   macro avg       0.06      0.25      0.09       494
weighted avg       0.05      0.22      0.08       494

Training time: 4645.199693202972
```

Fig. 16. Confusion Matrix for LSTM model with a single emotion per client