NCSR Demokritos & University of Peloponnese

Speech Emotion Recognition in Augmented Data

Authors:
Aristeidis Kalokyris (dit2106)

Nikolaos Katsimpras (dit2108)

July 7, 2022

I. Motivation

Sound signal recognition is one of the most well-acclaimed fields regarding the applications of Machine Learning, as a predecessor of Deep Learning, and Artificial Intelligence in the modern scientific and business interest. More specifically, the Speech Emotion Recognition (SER) is something with more than a handful applications either for psychology, social science, linguistics, and neurophysiology [1], or more professional applications such as in call centers, smart phone apps for well-being or emergency detection, etc. The most astounding of the results in the SER domain started appearing after the first implementations of Deep Learning (DL), Deep Neural Networks (DNN) [2] by extracting high-level features and training learning models in a more successful way. Although DL and DNNs are offering a great platform for improvement regarding the results received by today's computational resources, the one of the important parts in SER is the actual understanding of the information deriving from the features extracted by speech samples. On the other hand, after properly understanding that information, it is also equally important to integrate a DL model that accommodates each respective study or experiment in a most efficient way possible. The last part surely depends on an above-average background in mathematics, statistics and much more but the hands-on exercise, the experimentation with code, the DNN architecture selection and hyper-parameter tuning are at least as much significant as the first mentioned. In the following project we are conducting an experiment of training and testing different architectures of DL models by comparing their learning process and accuracy while performing SER on a corpus of annotated data.

II. Speech Emotion Recognition and project outline

As mentioned before, SER is a major challenge while chasing the DL approach. Beside the fact of feature extraction and data understanding, there is also the problem that there are never enough data to train a model thoroughly. Observing recent academic projects [3], the problem of lack of samples is solved by "infecting" one or more existing data sets with frequency augmentation techniques and utterly aiming in producing new speech samples, that are appended in the existing data corpus. Besides providing solution to the lack-of-data problem, data augmentation offers a more spherical spectrum of frequencies and hand-crafted features extracted out of the machine generated sound signals. In this experiment, we are developing a sequential DL model combining 1-4 convolution layers for high-level feature extraction and a recurrent model in the form of a Long short-term memory (LSTM), inspired by [4], while performing comparisons with a two-dimensional Convolutional Neural Network (2DCNN) and a simpler LSTM architecture. Additionally, we are trying different approaches on early stoppings during epochs, keeping model checkpoints while optimizing important metrics and reducing learning rate during learning phase. The final purpose is to manage training the model effectively and validate the training process by accurately predicting the emotions out of the validation data.

III. RAVDESS Data set

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [5] is a well-established database of speech and song recorded visually and auditorily by 24 actors, equally balanced between males and females, while expressing a certain emotion. The emotions recorded were calmness, happiness, sadness, anger, and fear. All samples are available in face-and-voice, face-only, and voice-only formats. In this experiment we are using only the speech data by extracting features out of 192 on average sound samples, form different male and female actors, of each emotion. It is important that the sample is fairly balanced, but as mentioned before it is not the best-case scenario for DL model training size-wise. On the other hand, the contribution of the selected data set has already been identified among scientific community as it is used in many instances of academic bibliography with the North American English giving more credit when used for product development by companies of the SER domain inside USA. In Figure 1 we have a first visual representation for the studied data set. Last but not least, the data are replicated in one of our academic Google Drive folders and a share invitation will be sent to the reader of this report to give the ability of conducting the experiments once again.

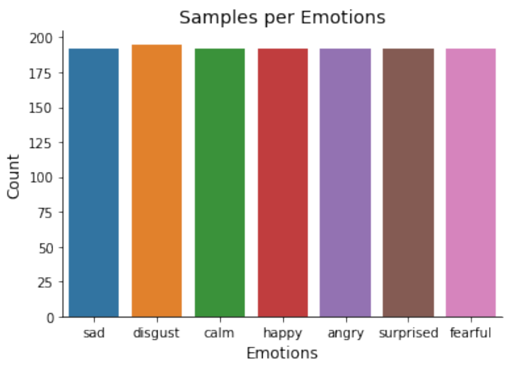

Figure 1: Emotion Count

IV. Data Augmentation

A large amount of training data is usually required for deep learning to achieve better results. One way to increase the number of training samples is through data augmentation. In this project we try to recreate new synthetic data samples by adding agitations on the initial training and test set. The augmentation techniques used were adding noise, stretch, pitching while extracting features from two different versions of each speech sample, turning our data set three times bigger than it could be without applying these two data augmentation techniques. Of course, it goes without saying that the new data records respected completely the annotation of the original samples. Every perturbation on the data is taking place during the feature extraction sub-phase of the experiment, by applying modifications on the sound samples using the Librosa Python library or by modifying the arithmetic arrays produced by the speech data. Let's have a better look on each technique individually:

- **Noise**

  Applying noise was done by multiplying the maximum frequencies of each data samples with a uniform and then a normal random value, while adding the result to the already existing speech sample.

- **Stretch**

  For the second augmentation technique we chose the pre-built Librosa function which performs time-stretch on imported audio by fixed rate and producing audio time series by the specified rate.

- **Pitch**

  The final one is also using a Librosa function which is altering the waveform by a number of defined steps, the output is also pitch-shifted audio time series.

In Figure 2 & 3 there is a simple representation of two wave-plots depicting two objectively different emotions.

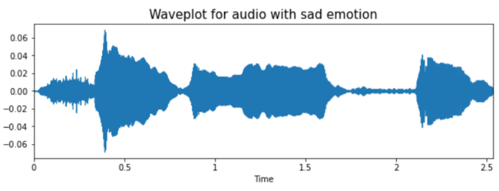

Figure 2: Sadness

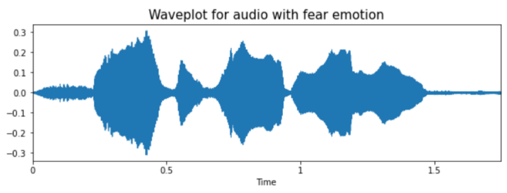

Figure 3: Fear

V. Neural Network Architectures

As already stated, there were three different DL approaches with the following three architectures, the 2DCNN was not using on training and testing augmented data, while the simple LSTM and the CNN+LSTM were taking augmented data into consideration:

- **Deep 2DCNNs**

This architecture creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs ("Deep Learning.ipynb"). By revisiting the algorithm one approach was to built several CNNs with different structure. Those CNNs models were trained with the MFCC data of the speech samples. In more detail, we extracted the MFCC data for the each speech sample. We chose to extract only 3.1 seconds from every sample with 500 milliseconds offset. We needed to fill with zeros the MFCC data that extracted from every sample that was under 3.6 seconds, until it was the wanted size of 223×20. We sampled our data by length of the FFT window to be 512 (as it is recommended by librosa library API) and the number of samples between successive frames to be 256. After our extraction we ended up with an numpy array of dimensions [num of samples] × [223] × [20], and an parallel array with had the label of every sample. Then we split our data to training data and test data with train size set to 75% of our samples. We set our models to be trained with learning rate that reduces when the loss of our model is not improving. Our 4 CNNs models have the structure of:

3×3 convolution layer with 1×1 stride and 32 filters with Relu activation
MaxPooling with pool size 2×2
3×3 convolution layer with and 32 filters with Relu activation
MaxPooling with pool size 2×2
Flatten data
Dence Layer with size 64 with Relu activation
Dence Layer with size 8 for our output
Trainable parameters 134,088 Accuracy: 54.57

3×3 convolution layer with 1×1 stride and 32 filters with Relu activation
MaxPooling with pool size 2×2
Flatten data
Dence Layer with size 64 with Relu activation
Dence Layer with size 8 for our output
Trainable parameters: 583,592 Accuracy: 57.61

3×3 convolution layer with 1×1 stride and 32 filters with Relu activation
MaxPooling with pool size 2×2
3×3 convolution layer with and 64 filters with Relu activation
MaxPooling with pool size 2×2
Flatten data
Dence Layer with size 64 with Relu activation
Dence Layer with size 8 for our output
Trainable parameters: 258,024 Accuracy: 62.88

3×3 convolution layer with 1×1 stride and 32 filters with Relu
activation
MaxPooling with Pool size 2×2
3×3 convolution layer with and 64 filters with Relu activation
MaxPooling with pool size 2×2
3×3 convolution layer with and 64 filters with Relu activation
MaxPooling with pool size 2×2
3×3 convolution layer with and 64 filters with Relu activation
MaxPooling with pool size 2×2
Flatten data

Dence Layer with size 64 with Relu activation
Dence Layer with size 8 for our output
Trainable parameters: 150,600 Accuracy: 64.265

We can see that the Deepest CNN had been more successful than the others. The number of trainable parameters also give a better performance but it is not very effective. Please consider examining Figure 4 for more information regarding the predictions.

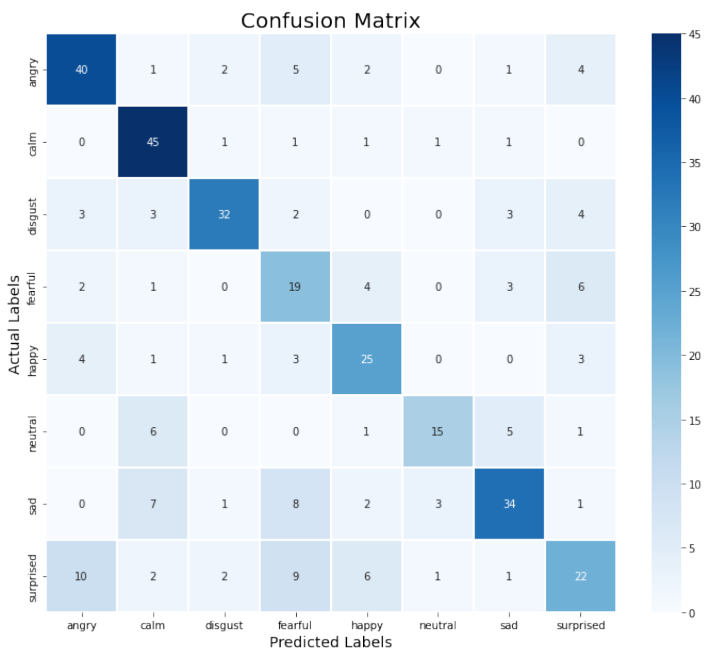

Figure 4: Deep CNN predictions per class

- **Simple LSTM**

As the second architecture prototype we used to a simple LSTM using the feature extracted from the augmented data for training and testing (emo rec DL only LSTM.ipynb). In this case the LSTM network used 128 dimensions as the dimensionality of the output space.

The activation function used was Relu, also known as the rectified linear activation function. This function is a piece-wise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

As a factor to prevent overfitting, the Dropout technique, provided by Keras, was used. Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. In this part of the experiment the Dropout factor among neurans was set to 20%.

Additionally, in this part a Softmax function was also used. Softmax is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.

Finally, the training phase took place among 60 epochs while also saving the model parameters after an optimum learning loss metric was observed with frequency of saving was happening every epoch, if needed, as seen in Figure 5. After running the DL model in the first 60 epochs, there were three more iterative steps with a hard threshold of 60 epochs again aiding the model to advance while optimizing training accuracy, testing loss and testing accuracy respectively.

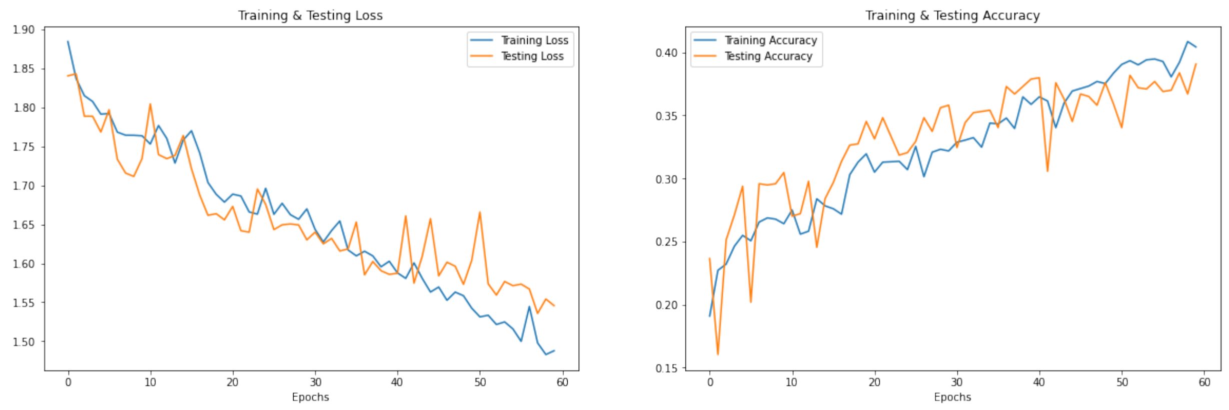

Figure 5: Simple LSTM Training Testing

The results, in terms of testing accuracy, were not so heartwarming but on the other hand it was a great test-bed to start improving the architecture by adding convolution. The final prediction accuracy achieved was 44% and as presented in Figure 7 we can have a better understanding of the various predicted classes.

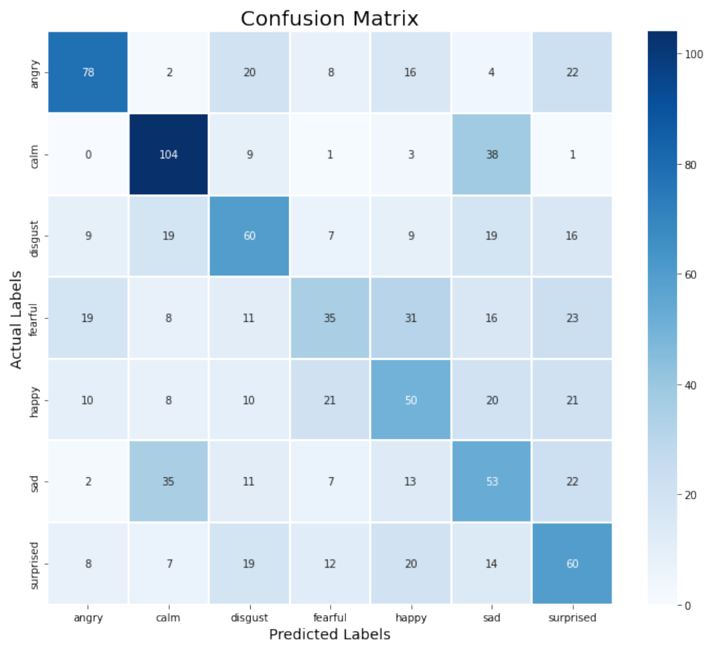

Figure 6: Simple LSTM predictions per class

- **Convolutional feature extraction fed in LSTM**

The final architecture tested that has the most promising results among the DL applications in SER is the cross-joint of a summary of convolutional layers in the beginning and the training using those high-level extracted features. Let us notice hear that the convolutional layers used are of mono-dimensional form and the essence of this experiment was that until one point, the more layers added the better the results were.

Also let us not leave unnoticed that the more layers added in the beginning the more high-level the features will get so this is a trade-off that either helps our model make better generalizations, or it is gonna be a negative factor by preserving latent information that will never benefit the training phase.

The most interesting part during the training of the final architecture was trying to understand the gap formed between Training and Testing loss through the main phase of the learning process, as seen in Figure 7. The recursive nature of the LSTM algorithm in addition to the high-level features extracted by the first CNN layers were an important factor to avoid over-fitting, although this cannot be seen in our model. Based on that the testing loss and testing accuracy was not improving by any sense, providing us a significant study for future experiments.
To summarize, the final confusion matrix in Figure 8 and the results for every different number of convolution layers before the training phase, in terms of testing accuracy can be seen below. The most successful among the final architecture of the experiment was found after using four convolutional layers before the deployment of the LSTM model:

- 1 1D-Convolutional layer fed in LSTM accuracy: 57.37%, reaching up to 63%

- 2 1D-Convolutional layer fed in LSTM accuracy: 59.94%, reaching up to 61%

- 3 1D-Convolutional layer fed in LSTM accuracy: 61.22%, reaching up to 66%

- 4 1D-Convolutional layer fed in LSTM accuracy: 63.99%, reaching up to 66%

The upper bounds defined in the end of each experiment are affected by the random initiation and the recursive nature of the LSTM model in addition to the Dropout techniques implemented.

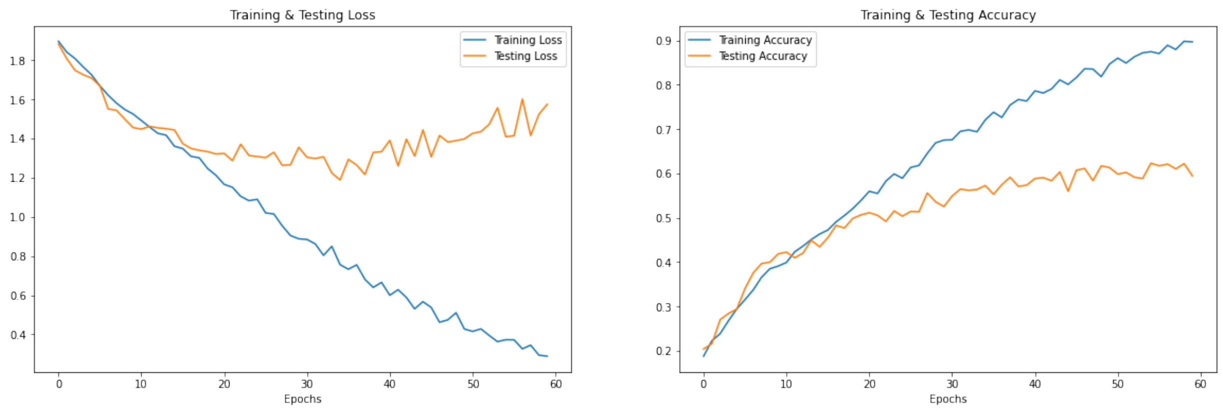

Figure 7: CNN + LSTM Training Testing

VI. Future Work

After our first introduction, with DL and NNs in general, the first thing coming to our minds is to try to append some more relevant data in our data corpus and test the presented models once again. After this a transfer learning scenario should be really interesting. Also, a similar application in cases where multiple persons communicating should be a really challenging task by recognising ad-hoc the emotion after each communication part. Last but not least, it is obvious that some more architectures and different augmentation techniques on various audio features extracted, is going to be a crucial step in understanding more parts of SER.

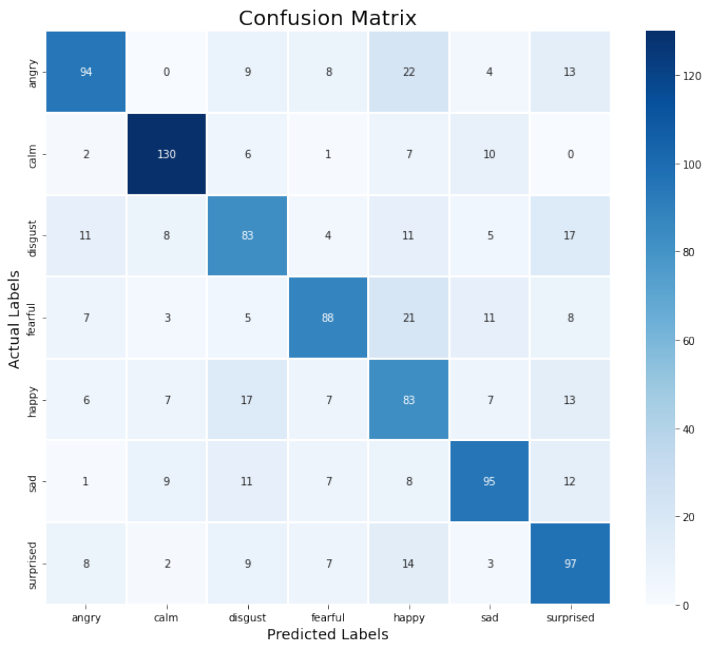

Figure 8: CNN + LSTM Final prediction among classes

References

1. Swain, M., Routray, A. Kabisatpathy, P. Databases, features and classifiers for speech emotion recognition: a review. Int J Speech Technol 21, 93–120 (2018).

2. Han, Kun, Dong Yu, and Ivan Tashev. "Speech emotion recognition using deep neural network and extreme learning machine." Interspeech 2014. 2014.

3. A. A. Abdelhamid et al., "Robust Speech Emotion Recognition Using CNN+LSTM Based on Stochastic Fractal Search Optimization Algorithm," in IEEE Access, vol. 10, pp. 49265-49284, 2022, doi: 10.1109/ACCESS.2022.3172954.

4. Etienne, Caroline, et al. "Cnn+ lstm architecture for speech emotion recognition with data augmentation." arXiv preprint arXiv:1802.05630 (2018).

5. Livingstone, Steven R., and Frank A. Russo. "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English." PloS one 13.5 (2018): e0196391.