AAA PROJECT
NEURAL NETWORKS

COMPARISON OF CNN AND RNN ARCHITECTURES FOR SPEECH EMOTION RECOGNITION FROM RAW AUDIO

Duarte Balata
Miguel Oliveira

Advanced Machine Learning - 2020/2021

Faculty of Sciences of the University of Lisbon

May 2, 2021

INTRODUCTION

Understating all the mechanisms through which sound is perceived and interpreted by humans has been an ongoing challenge in many research fields over the years (Belin et al., 2012). In their simplest form, emotions are usually represented as a static single image, usually associated to a facial expression. However, in the real world, emotions are much more dynamic and not only recognized visually. As an event in time and space, sound produces a non-visible frequency that is independently interpreted by the listener via vibrations that are picked up by the eardrums and then processed via the nervous system. The way sounds are interpreted by humans accounts for many variables such as volume, pitch, intonation and many others (Oxenham et al, 2018).

The recognition of these complex patterns that allow for the recognition of specific conveyed feelings and emotions has also been a topic of research in Machine Learning for some years now. One of the main challenges in this area has to do with how can sound be represented in a way that is machine readable, without losing the information needed to interpret its intricate patterns. One of the ways found to tackle this issue was the transformation of digitally recorded sound waves into Mel-Frequency Cepstrums (MFC) (Ittichaichareon et al., 2012). MFCs are composed of multiple coefficients (MFCCs), that can be interpreted as features representing phonemes (distinct units of sound) as the shape of the vocal tract, which is responsible for sound generation. MFCCs can be generated from spectrogram representations of sound, which are defined as visual representations of the spectrum of frequencies of a signal varying with time. The process of sound information capture under the form of MFCCs is complex and won't be described in detail here, but can be shortly summarized as *the result of a cosine transform of the real logarithm of the short-term energy spectrum expressed on a mel-frequency scale* (Tiwari et al., 2010).

On the present work, we explore the potential of MFCC representations to perform classification tasks on sound data using different Neural Network architectures. In this context, we will use an highly controlled dataset that simulates vocal expressions with different emotions, the Ryerson Audio-Visual Database of Emotional Speech and Song dataset - RAVDESS Livingstone and Russo, 2018). This dataset published by Livingstone and Russo (2018) on the PLOS ONE magazine, consists of 7354 audio-visual recordings of actors expressing multiple emotions. The recordings were independently rated 10 times by 247 individuals for emotion validity. In the context of this work, we will exclusively make use of the emotional speech audio data, discarding the songs and video footage.

By using this dataset, we firstly try to predict the actor's gender based on their speech audio and then the emotion that is being expressed. For that, we make use of two types of networks that have been recently used with considerable success in the prediction of sound data - Convolutional Neural Networks (used by Demir et al., 2020) and Recurrent Neural Networks (used by Lezhenin et al., 2019).

APPROACH

For the emotion classification task, we decided only to use emotions that were labeled as intensely strong so that they can be more clearly distinguishable. In addition, the data was labeled with the emotion it portrayed as well as the gender of actor/actress. The emotions used were: fear, disgust, sadness, anger, surprise, calmness and happiness. Each emotion had 96 samples, and male and female voices were evenly split (336 each).

In the data pre-processing phase of the work we made use of Python's audio library Librosa (McFee et al., 2015). This library allowed us to plot the waveforms of the samples (annex figure A.1) by using the library's 'display waveform' function and listen to the audio directly on the notebook, which allowed for better data exploration. As the dataset used was built in a highly controlled environment, the audio did not contain any background noises or other data that could interfere with the predictions, therefore not requiring any type of filtering or improvement. For feature extraction, Librosa allowed for the transformation of data into the previously described Mel Frequency Cepstral Coefficients. This method provided us with features that are representative of the power and intensity of a sound. This was plotted with two dimensions, as the y-axis represented the multiple MFCC bands and the x-axis the timesteps on the recordings. This way, we are able to extract each emotion's features and how they differ in the course of a speech (annex figure A.2).

The MFCC values were then stored on an array for each sample. In order to make the dataset suitable for a simple preliminary supervised machine learning task, a timeseries containing information on the average value of all twelve MFCC bands for each sample was saved, reducing the dimensionality of data into one dimension and allowing for the implementation of simpler and faster models, such as the 1-dimensional CNN (annex figure A.3).

To prepare the data, MFCC values for each emotion (either simplified by average or in a 2-dimensional matrix), along with two target columns “emotion” or “gender” were added to a dataframe in order to later be divided into training, validation and test sets for the ML models, with the proportions of 7 - 1.5 - 1.5. This way, the target on the classification tasks could easily be switched between “emotion” and “gender”. In addition, a function was also created where the “emotion” target could be filtered by either “male” or “female” only, allowing for the creation of more uniform sub-sampled datasets that would be later used to train models with an higher specificity. As the targets were not numerical, Sklearn’s LabelEconder was used to prepare the targets for use in the ML models.

Having prepared the data, we will begin by experimenting with the Convolutional Neural Network (CNN), with and without data normalization. The network will be first tested to predict the gender of the speaker, by using a 1-Dimensional input shape on the model training. The same model will then used to predict the emotion of the speaker, on the entire dataset and divided by gender. The architecture will be amended to contain some techniques capable of increasing the predictive performance of the algorithm. These techniques will be further described on the next chapter of this work.

Having tuned the 1-Dimensional CNN, a 2D CNN with similar parameter tuning will then used in order to test for the effect of dimensionality reduction and data simplification in the predictive capabilities of the model. Using this network, we try to predict the speaker’s gender and emotions on the whole dataset, and emotion separated by gender.

Finally, we will try to improve upon the results of the 2D CNN by using a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) to predict emotions on the whole dataset, and again, divided by gender.

IMPLEMENTATION

1D Convolutional Neural Network - Baseline model

The input layer on this network accepts one dimensional inputs of size corresponding to the number of timesteps on each audio recording (216). This is possible because, as previously stated, for this model, an average of the values for the 12 calculated MFCC bands is used on each timestep.

The architecture of this network is influenced by the work of Su et al. (2019), with some adaptations based on trial and error procedure and consultation from other ML online resources. The input layer is followed by a set of convolutional layers, whose size and quantity were defined based on the consultation of multiple implementation available on GitHub and Kaggle.

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations where a small array of numbers, called a kernel, is applied across the input tensor. The process is repeated by applying multiple kernels to form an arbitrary number of feature maps, which represent different characteristics of the input, meaning that different kernels can be considered as different feature extractors. As such the key hyperparameters that are frequently used to define the convolution operation are size and the number of used kernels (Rikiya Yamashita el al., 2018).

On every developed network, all convolutional layers use ReLU as their activation function and a kernel size of 8. The first two convolutional layers have 256 neurons and are followed by 4 convolutional layers with 128 neurons each. Finally we have 2 layers with 64 neurons each, summing to a total of 8 convolutional layers.

In addition to convolutional layers this model already included two pooling (or sub-sampling) layers, between hidden layers with different numbers of neurons. Pooling layers are frequently combined with convolution layers in order to reduce the resolution of the feature maps and therefore reduce the model’s sensitivity to distortion and avoid overfitting. Max-pooling is one of the most frequently used pooling algorithms and is available as part of the Keras framework. This algorithm runs over the its inputs in steps of the provided length, which we defined as 8 in the present work, and selects from each set of neurons the one presenting the maximum value. The algorithm then outputs a simplified matrix where each element is the max of a region in the original input (figure 1) (We et al., 2015).

After the last convolutional layer on the network, there's a need to include a flattening layer. The process of flattening consists of converting the data into a 1-dimensional array for inputting it to the next layer. The output of the convolutional and max-pooling layers needs to be flattened in order to create a single long feature vector, which is then connected to the final layer on the network, a fully-connected (or dense) layer that's responsible for the output of the final data classification from the extracted features (figure 2).

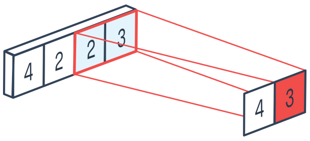


Figure 1: Max-pooling applied to a 1-dimensional input vector.

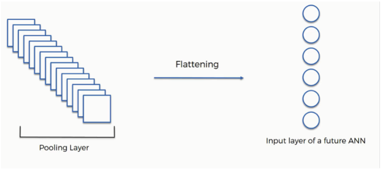


Figure 2: Flattening of outputs from pooling layer

The first architecture of the 1-dimensional convolutional neural network used on this project can be seen on annex figure A.4 (the number of neurons on the output layer varies with the number of features being predicted on each task).

1D Convolutional Neural Network - Improved model

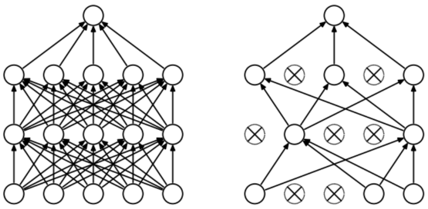
In order to improve upon the previously described baseline model (adapted from bibliography), batch normalization and dropout layers were added to the architecture.

Batch normalization, firstly introduced by Ioffe and Szegedy (2015) is a technique frequently used to solve the problem of internal covariate shift. This problem derives from the fact that the distributions of each layer's input changes during the training process. In order to avoid this, the batch normalization layer will normalize the output of its previous layer by subtracting the batch mean μ and dividing by the batch standard deviation σ, and pass the resulting values to the subsequent layer (Bjorck et al., 2018).

$$\hat{O} = \frac{O - \mu}{\sigma}$$

Batch normalization is widely used because it reduces the influence of the initially set learning rates on the final model performance and also accelerates the learning process by allowing for much higher learning rates (Bjorck et al., 2018).

The model was further improved by adding Dropout layers. Dropout is a technique introduced by Srivastava et al. (2015) that's used to deal with overfitting by randomly dropping neurons and their related connections in the network during the training process (figure 3). This forces the network to only use a fraction of its available units to learn from the data on each step and therefore avoid the co-adaptation of units, meaning that the optimal weights found for a given neuron will be less dependant on the weights of neurons from their adjacency (Baldi and Sadowski, 2013).



(a) Standard Neural Net (b) After applying dropout.

Figure 3: Representation of unit Dropout in a simple Neural Network

Throughout this work, the dropout implement on the multiple models was kept at a rate of 0.2, since testing as shown that this value resulted in better performances.

The improved architecture of the 1-dimensional convolutional neural network can be seen on annex figure A.5 (the number of neurons on the output layer varies with the number of features being predicted on each task).

2D Convolutional Neural Network

After performing the due improvements on the 1-dimensional CNN, the model was transformed in order to be able to receive 2-dimensional inputs. This extra feature complexity was added in hope of increasing the model's predictive performance at the cost of a more time demanding learning process.

In order to maintain the previously described successful architecture, without increasing the run time in a way that would render this work too time consuming to perform, we decreased the number of neuron on each convolutional layer by half of the original values. The final architecture for this network can be consulted on annex figure A.6.

Recurrent Neural Network (Long Short-Term Memory)

The implemented Recurring Neural Network was based on the work by Raza et al. (2019) and consists of two LSTM layers with 128 neurons, followed by two fully connected (dense) layers of 512 neurons and final output layer of size corresponding to the number of possible labels for the task at hand.

Stacked LSTMs, as are called models with consecutive hidden LSTM layers, were introduced by Graves, et al. in their application of LSTMs to speech recognition and have been widely used since then.

Long Short Term Memory networks can be described as a special kind of RNN, capable of learning long-term dependencies. These networks, introduced by Hochreiter Schmidhuber (1997), present gates that can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions, avoiding the known short-term memory issue on classical RNN's (Gers et al., 1999). The final architecture for the LSTM network can be consulted on annex figure A.7.

Results

When trying to predict actor gender using a 1-dimensional CNN without data normalization, batch normalization or dropout, we were able to achieve an accuracy of around **88%** on the training data. However, by looking at model's loss and accuracy over training epochs , it was possible to see that the values were constantly spiking, even after testing with multiple batch size values (figure 4).

After performing some optimizations on the architecture, the model was fit to test data again, and it was able to predict the actor gender labels with an accuracy of **91%**. The spikes that were visible on the previous plot were also significantly reduced likely due to the effect of batch normalization (figure 5). However, this time it was possible to notice that there were clear signs of overfitting by looking at the loss curves, since loss on the validation data started to increase after some time, while the loss value on the training data kept decreasing. As such, an early stop function was added, with a patience of 20 epochs, so that the model would not run for the entirety of the defined 150 epochs with increasing loss on the validation data. The effect of this early stop is noticeable on figure 6. It not only resulted in an higher gender prediction accuracy on the testing set - **93%** - but also significantly reduced the model's training time.

Subsequently, this same architecture was used to train a model to predict actor's emotions from the entire dataset. When tested on the test data, it resulted on an accuracy of around **55%**. Which is far better than a random guess, considering there were seven classes of emotions. However, the performance was determined to be held back by the simplicity of this model.

Since male and female voices tend to present significant differences in terms of pitch, volume and intonation, the emotion classification task was then carried on on sub-datasets divided by actor gender in hopes to increase the accuracy of the predictions. By doing so, it was possible to see that the emotion on female actresses were

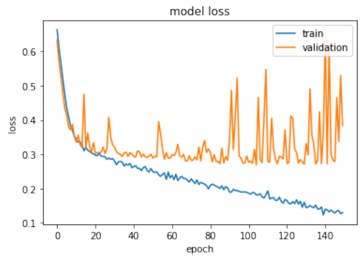


Figure 4: Loss over time for the base model.

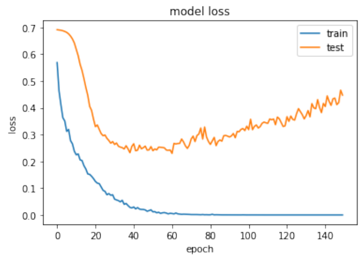


Figure 5: Loss over time for the optimized model.

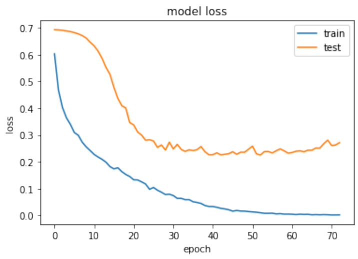


Figure 6: Loss over time for the optimized model with early stop.

hard for this model to predict - with an accuracy **44%** - while the emotions on male actors were predicted with an increased accuracy of **60%**. Considering the performances on both sub-datasets, there's no clear advantage in dividing the data at this point since the average performance didn't show any improvement.

All things considered, the fully tuned 1-dimensional CNN displayed very good results on gender prediction tasks, predicting the right class more than **93%** of the times with a very fast training time. As such, this model was can be considered a good option when performance is important and the task at hand is relatively simple. However, when the classification complexity increased, the model displayed sub-optimal results, that can nonetheless be used as a baseline for subsequent testing.

By using the previously described 2-dimensional CNN architecture to predict actor gender on the test samples, we were able to achieve an accuracy of **100%**, even tough the training process was more costly in terms of time. As for emotion prediction using the 2D CNN, we were able to achieve an accuracy of **76.2%** (figure 7) when training on the entire dataset, which represents an increase of more than 20% in accuracy when compared to the 1D model. This indicates that features as intricate as emotions require an higher amount of information contained on the MFCC bands to explain their complexity.

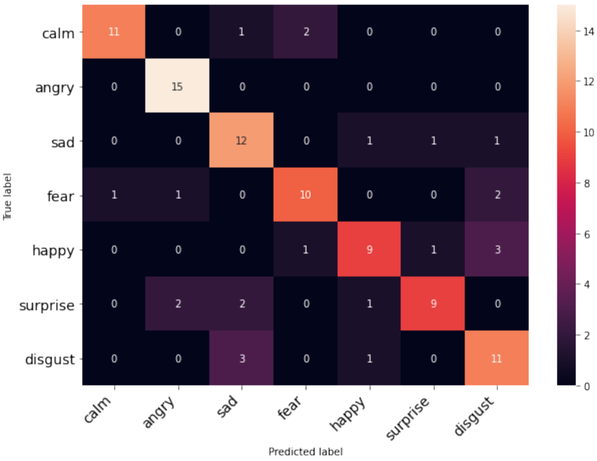


Figure 7: Contingency matrix of emotion prediction using 2D CNN.

By looking at figure 7, we also can see that Happiness and Surprise were the most difficult emotions to predict between male and female subjects, while Anger has been perfectly classified, indicating that it presents a unique sound pattern, clearly distinguishable from the rest of the emotions.

We then performed the previously mentioned experiment of sub-sampling the data by gender and verified that it was now able to significantly increase the performance, in comparison to training the model on the entire dataset. The accuracy for female emotion prediction using this model was of **94%**, while for males it was **88%**. Even though this values might be overestimated due to the low sample sizes, after multiple trials it was possible to understand that there's a clear advantage in sub-sampling the data by gender. Since the model specialized in gender prediction resulted in an accuracy of 100%, we consider that a pipeline of the three specialized models could give origin to the best results in a real world scenario (Graves et al., 2012).

Finally, since we had already achieved a perfect performance on gender prediction using the 2-dimensional CNN, we tested the possibility of improving upon emotion classification by using the previously described RNN architecture. By training this model on the complete training set, it was able to achieve a prediction accuracy of **67.3%**, which although much better than the baseline performance set by the 1D CNN, falls short of the 2D CNN performance under the same conditions by almost 10%. Given the margin between performances on the two models for the entire dataset, sub-sampling was not deemed necessary to prove that 2-dimensional convolutional network is the more performant model for this task.

Throughout all the described tests, we found out that the best results could be consistently achieved by using the RMSProp optimization algorithm, with a categorical crossentropy loss function, over other options such as Gradient Descent or the Adam optimizer. The . RMSprop uses an adaptive learning rate instead of treating the learning rate as a hyperparameter, meaning that the learning rate changes over time, which is helpful to avoid the vanishing gradients problem. A batch size of 16 was also considered optimal to tackle the classification problem at hand on all types of models, displaying a good compromise between train time performance and prediction accuracy.

Additional outputs, such as models' learning curves and contigency matrices, can be consulted in the Notebook annex to this work.

FINAL COMMENTS

The extraction of accurate metadata from content is a challenge for today's data driven business models that seek to improve their products. Services like recommendation systems on streaming platforms rely on data from previously watched content to provide suggestions on what to watch next. Improving the quality of the metadata that is extracted can be key to improving recommendation services and overall user experience. Emotional speech recognition as been proposed as one way to generate metadata for such services that can enable the classification of audiovisual data. Additionally, this type of data can also be useful on customer service operations such as call centers where emotional speech recognition can be used to understand customer satisfaction and improve the service. Other possible uses include security or policing systems, where emotional speech recognition may alert authorities in cases where there a significant level of fear or distress. Despite its possible advantages, ethical and privacy concerns must be taken into serious account before collecting and analysing speech from individuals.

On our implementations of this classification task, separating individuals by gender allowed us to achieve a greater level of success that when using the entirety of the dataset. Using both 1D and 2D CNN models enabled us to predict the gender of individuals with an high accuracy in the test group, with the 2-dimensional model displaying a perfect classification, therefore indicating that results in a real-world setting, while not perfect, may lead to extremely satisfactory results. Emotion prediction, on the other hand, was shown to require inputs of higher complexity that are capable of conveying the many intricacies emotion expression. The simpler CNN's (1D) did not have such positive results for this type of task, however two-dimensional CNN's lead to quite satisfactory results. Due to the temporal component of audio data, we expected that RNN would provide stronger performances. Despite testing different configurations, the RNN model was able to outperform the 1D CNN but the same did not occur for the 2D CNN, that clearly outperformed the RNN. Further architecture testing and the availability of more computational resources, could eventually lead to the discovery of a more performant model.

It is important also to address some limitations on this project. The dataset used was not very large, and data augmentation techniques could have been implemented to increase its size, given more time. The dataset was extremely clean and uniform from the start, without any background noises and the same recording environment for every actor. This can be limit the ability of the model to generalize in real world applications, therefore future work should look to use different data sets with a less controlled environment. A more in depth approach could have combined different types of data, such as images or videos, with facial expressions of the emotions expressed. Finally, without current time restrictions, a higher number of Neural Networks architectures could have been experimented with to compare results, as well as further parameter tuning.

BIBLIOGRAPHY

1. Baldi, P., Sadowski, P. J. (2013). Understanding dropout. Advances in neural information processing systems, 26, 2814-2822.

2. Belin, P., Bestelmeyer, P. E., Latinus, M., Watson, R. (2011). Understanding voice perception. British Journal of Psychology, 102(4), 711-725.

3. Bjorck, J., Gomes, C., Selman, B., Weinberger, K. Q. (2018). Understanding batch normalization. arXiv preprint arXiv:1806.02375.

4. Demir, F., Abdullah, D. A., Sengur, A. (2020). A new deep CNN model for environmental sound classification. IEEE Access, 8, 66529-66537.

5. Gers, F. A., Schmidhuber, J., Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

6. Graves, A. (2012). Supervised sequence labelling. In Supervised sequence labelling with recurrent neural networks (pp. 5-13). Springer, Berlin, Heidelberg.

7. Hochreiter, S., Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. Advances in neural information processing systems, 473-479.

8. Ittichaichareon, C., Suksri, S., Yingthawornsuk, T. (2012, July). Speech recognition using MFCC. In International conference on computer graphics, simulation and modeling (pp. 135-138).

9. Lezhenin, I., Bogach, N., Pyshkin, E. (2019, September). Urban sound classification using long short-term memory neural network. In 2019 Federated Conference on Computer Science and Information Systems (FedCSIS) (pp. 57-60). IEEE.

10. Livingstone, S. R., Russo, F. A. (2018). 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), e0196391.

11. McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., Nieto, O. (2015, July). librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference (Vol. 8, pp. 18-25).

12. Oxenham, A. J. (2018). How we hear: The perception and neural coding of sound. Annual review of psychology, 69.

13. Raza, A., Mehmood, A., Ullah, S., Ahmad, M., Choi, G. S., On, B. W. (2019). Heartbeat sound signal classification using deep learning. Sensors, 19(21), 4819.

14. Rintala, J. (2020). Speech Emotion Recognition from Raw Audio using Deep Learning.

15. Su, Y., Zhang, K., Wang, J., Madani, K. (2019). Environment sound classification using a two-stream CNN based on decision-level fusion. Sensors, 19(7), 1733.

16. Tiwari, V. (2010). MFCC and its applications in speaker recognition. International journal on emerging technologies, 1(1), 19-22.

17. Wu, H., Gu, X. (2015). Max-pooling dropout for regularization of convolutional neural networks. In International Conference on Neural Information Processing (pp. 46-54). Springer, Cham.

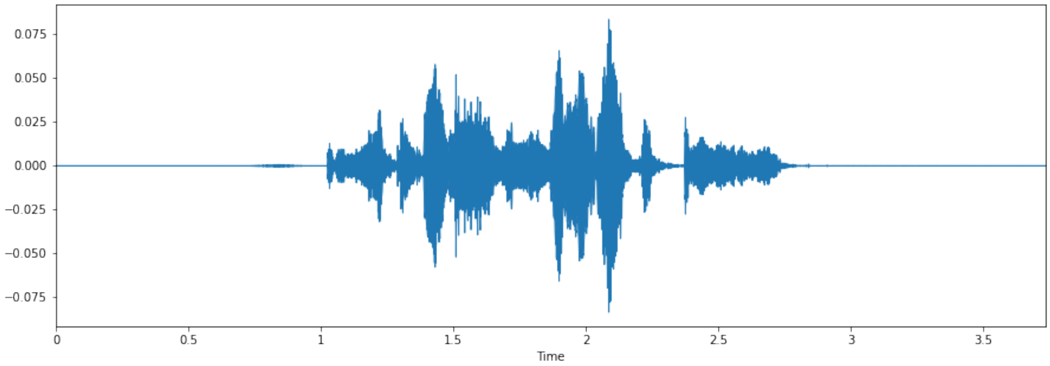


Figure A.1: Soundwave ploted from the audio of a female actress expressing surprise

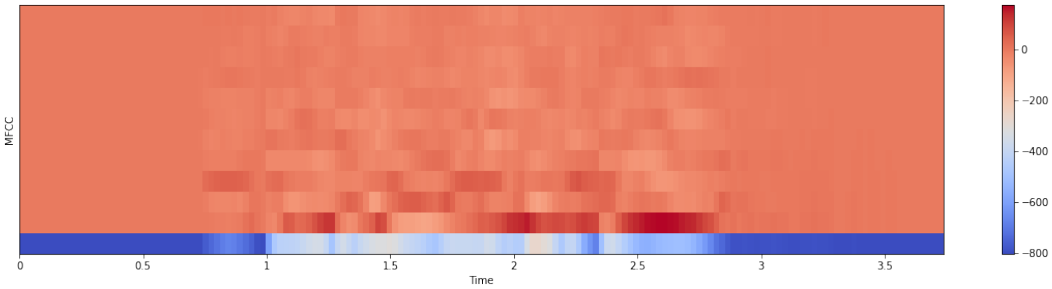


Figure A.2: MFC with 12 bands generated from the audio of a female actress expressing surprise

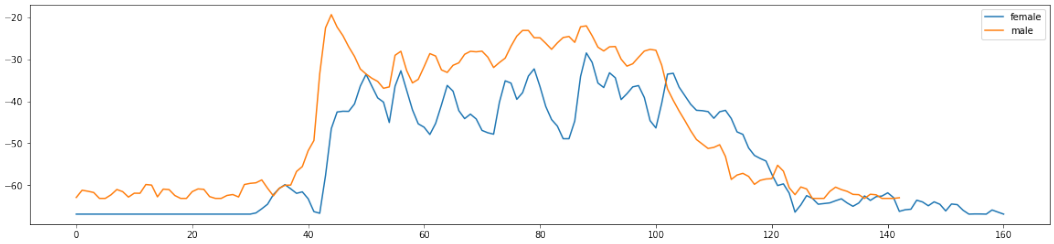


Figure A.3: Average values for the 12 MFCC bands of a male and female actors expressing the surprise emotion.

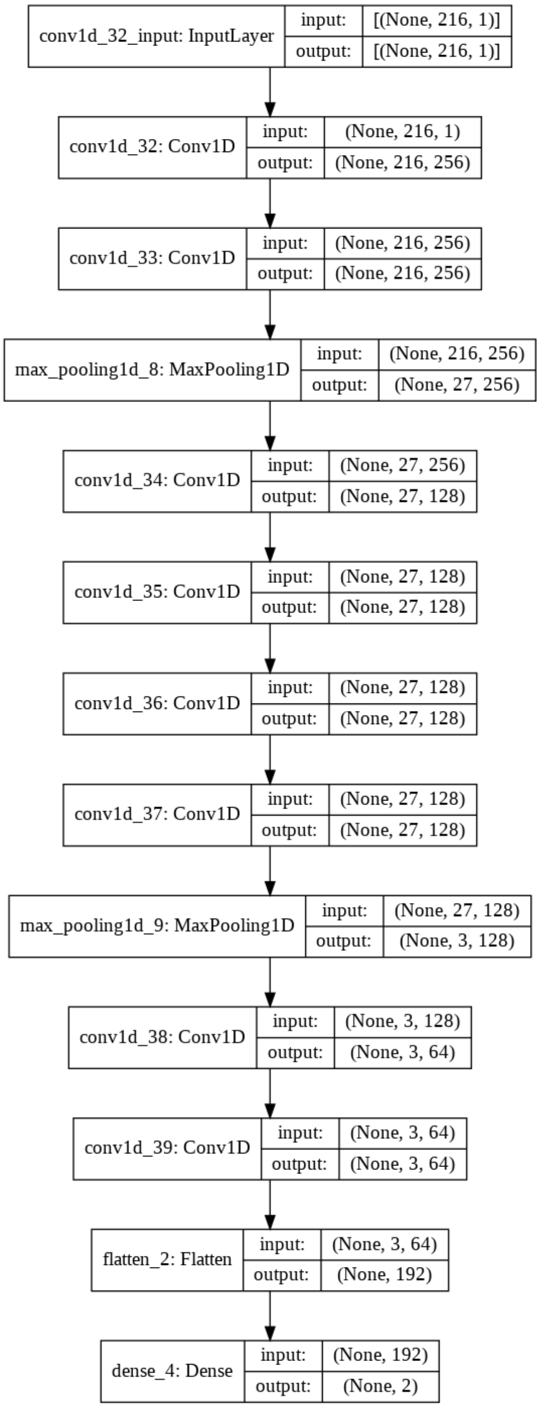


Figure A.4: Base 1D CNN architecture.

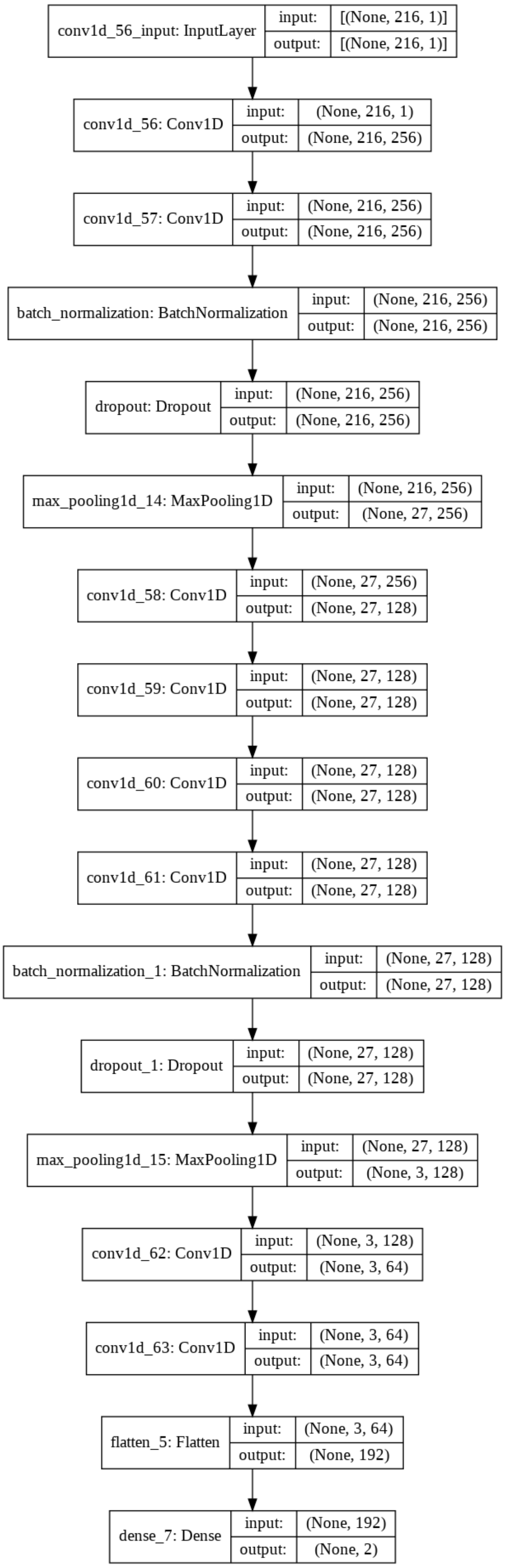


Figure A.5: Optimized 1D CNN architecture.

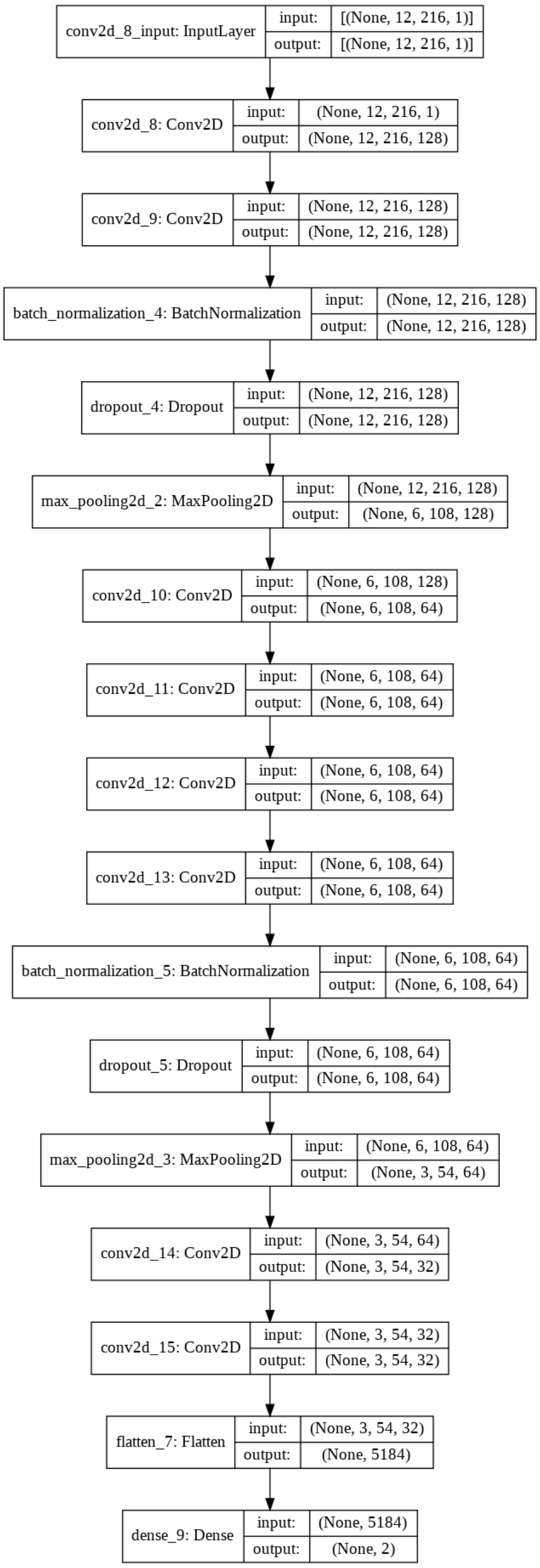

conv2d_8_input: InputLayer
input: [(None, 12, 216, 1)]
output: [(None, 12, 216, 1)]
conv2d_8: Conv2D
input: (None, 12, 216, 1)
output: (None, 12, 216, 128)
conv2d_9: Conv2D
input: (None, 12, 216, 128)
output: (None, 12, 216, 128)
batch_normalization_4: BatchNormalization
input: (None, 12, 216, 128)
output: (None, 12, 216, 128)
dropout_4: Dropout
input: (None, 12, 216, 128)
output: (None, 12, 216, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 12, 216, 128)
output: (None, 6, 108, 128)
conv2d_10: Conv2D
input: (None, 6, 108, 128)
output: (None, 6, 108, 64)
conv2d_11: Conv2D
input: (None, 6, 108, 64)
output: (None, 6, 108, 64)
conv2d_12: Conv2D
input: (None, 6, 108, 64)
output: (None, 6, 108, 64)
conv2d_13: Conv2D
input: (None, 6, 108, 64)
output: (None, 6, 108, 64)
batch_normalization_5: BatchNormalization
input: (None, 6, 108, 64)
output: (None, 6, 108, 64)
dropout_5: Dropout
input: (None, 6, 108, 64)
output: (None, 6, 108, 64)
max_pooling2d_3: MaxPooling2D
input: (None, 6, 108, 64)
output: (None, 3, 54, 64)
conv2d_14: Conv2D
input: (None, 3, 54, 64)
output: (None, 3, 54, 32)
conv2d_15: Conv2D
input: (None, 3, 54, 32)
output: (None, 3, 54, 32)
flatten_7: Flatten
input: (None, 3, 54, 32)
output: (None, 5184)
dense_9: Dense
input: (None, 5184)
output: (None, 2)

Figure A.6: 2D CNN architecture.

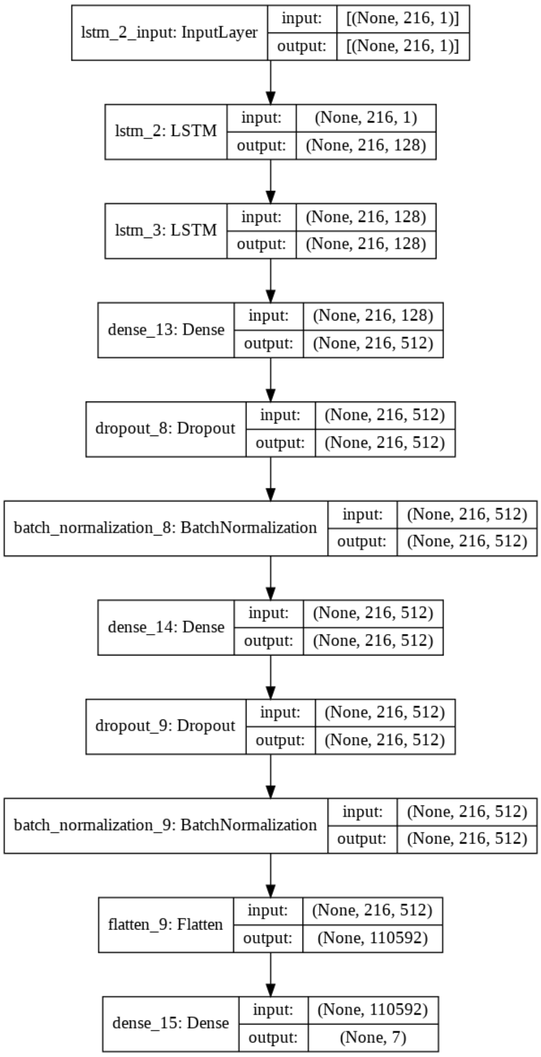


Figure A.7: RNN(LSTM) architecture.