CHAPTER 2

Methodology

This section introduces the hypothesis and the analytical validation of the proposed solution.

2.1 Proposed hypothesis

In source microphone detection through recorded audio samples, we will be given a recorded audio sample and we want to identify on which mobile device was it recorded. This problem falls under the supervised audio classification.As show in Figure 2.1, Our model is based on CNN and LSTM-based classifier network. We extract useful features present in the audio file through log-filter bank. The features are passed as input to both networks. Attention-based LSTM computes useful features from the sequential input. The output from both these layers is then multiplied and passed onto a classification network to make accurate predictions about the class of audio device and gender of the user in the audio device.

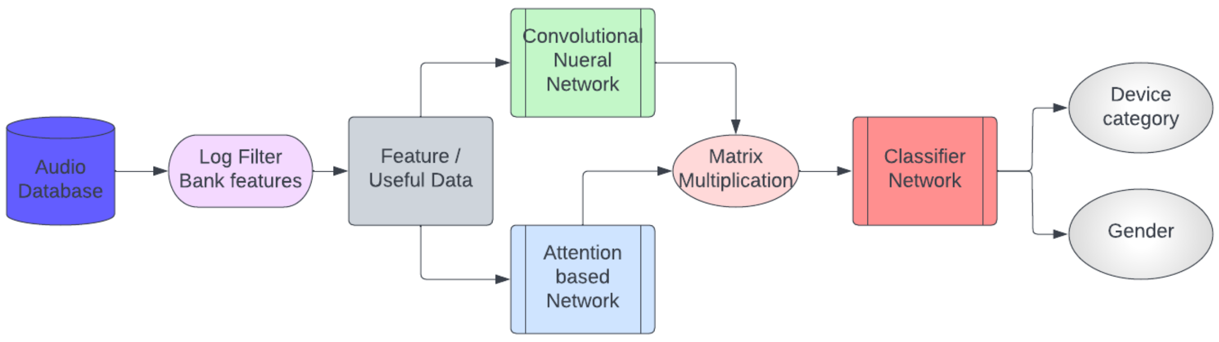

Figure 2.1: Proposed model workflow consists of three blocks - CNN, LSTM and Classifier Network.

2.2 Novelty

The novel idea here is to avoid overfitting we will be tasking our model to predict the gender of the user also. No other research paper has used this important metadata information to increment the model performance.This way our system would be multi-tasking which would make it difficult to start overfitting the dataset which was the main problem in all the research papers as the open-sourced dataset for such a problem is very small in size.

2.3 Modelling Layers

2.3.1 Convolutional Neural Network

These are neural networks that are specialized in the Image format data, which is resembled MFCC graphs. These are formed using blocks that consist of Convolutional layers, Pooling layers followed by batch normalization. A convolutional layer does the main processing of using filters/kernels and overlapping it all over the data to generate feature maps as shown in Figure 2.2. Using multiple of these gives us features of depth that are equal to the number of filters we used.Pooling is used to minimize the area of interest for the next step. We have used Average pooling, followed by batch normalization which normalizes the outputs to regulate overfitting.

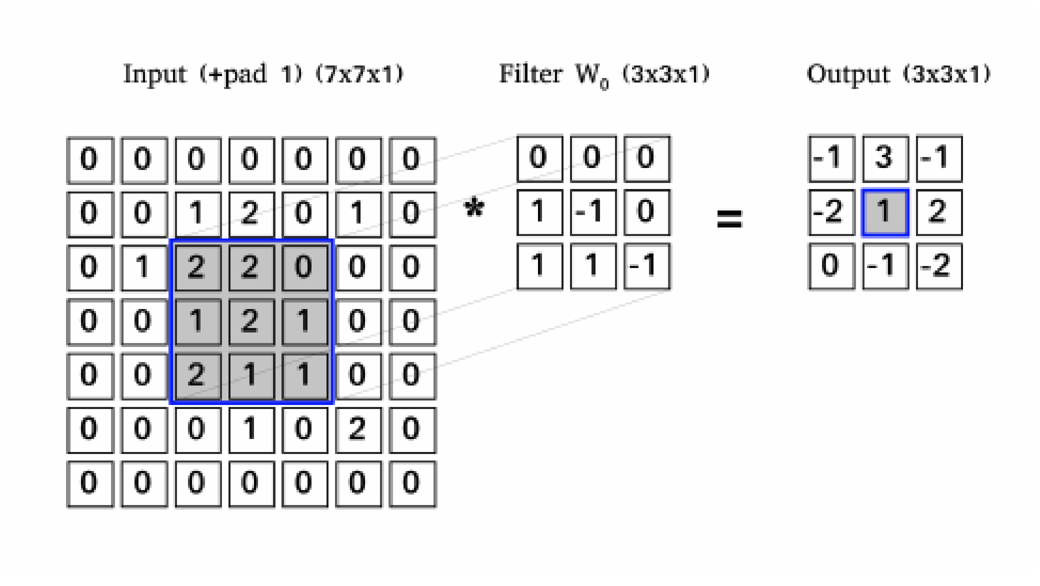

Figure 2.2: Overview of Convolutional Operation in a Convolutional layer.

2.4 Dataset

We have used the public Mobiphone dataset comprises 21 different mobile devices with 24 audio files each comprising 12 males and 12 females [10]. The different mobile device manufacturers are Apple, HTC, LG, Nokia, Samsung, Sony, and Vodafone. This is the most popular open-sourced common data used to set benchmark results. We would be computing 24 log filter bank features. The audio file is filtered first for pre-emphasis which boosts the high-frequency components while ignoring the low-frequency components of the signal. The signal is divided into overlapping frames. The power spectrum is then calculated for each frame. The sum along frames gives the energy of the signal for the frame. We also compute the 24 Mel - filter bank coefficients. And multiply it with the frame energy we calculated earlier. This is our filter bank feature and then we do a simple log over the values hence the replacement of zero in the previous step. These features are then normalized.

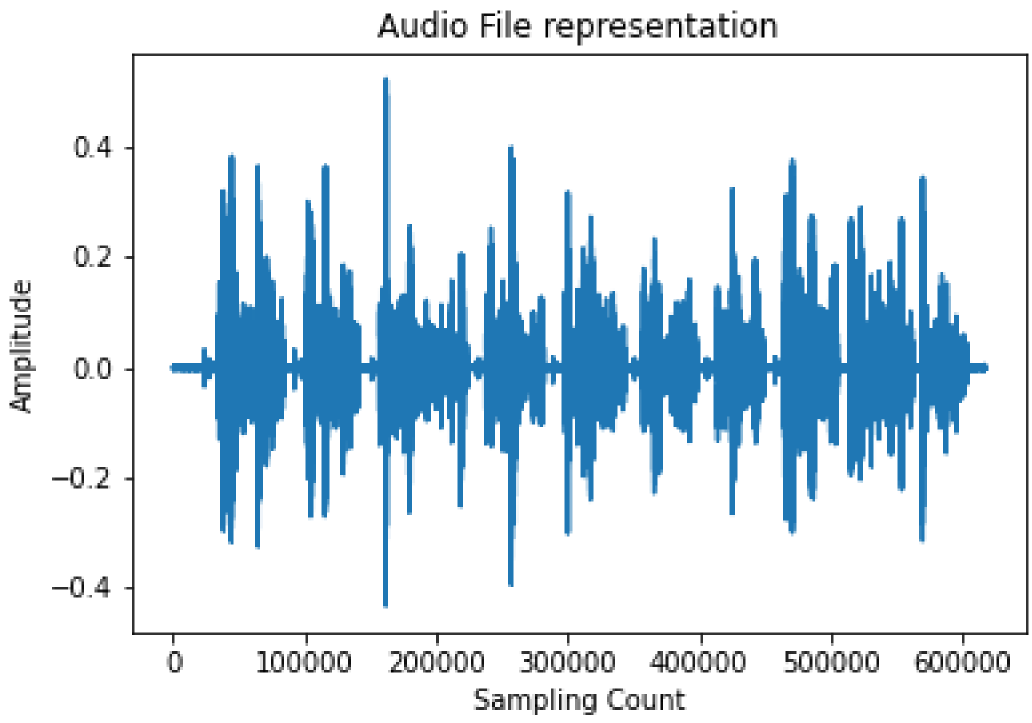

Figure 2.5: Amplitude Representation of a randomly selected audio file having 16Khz sampling rate.

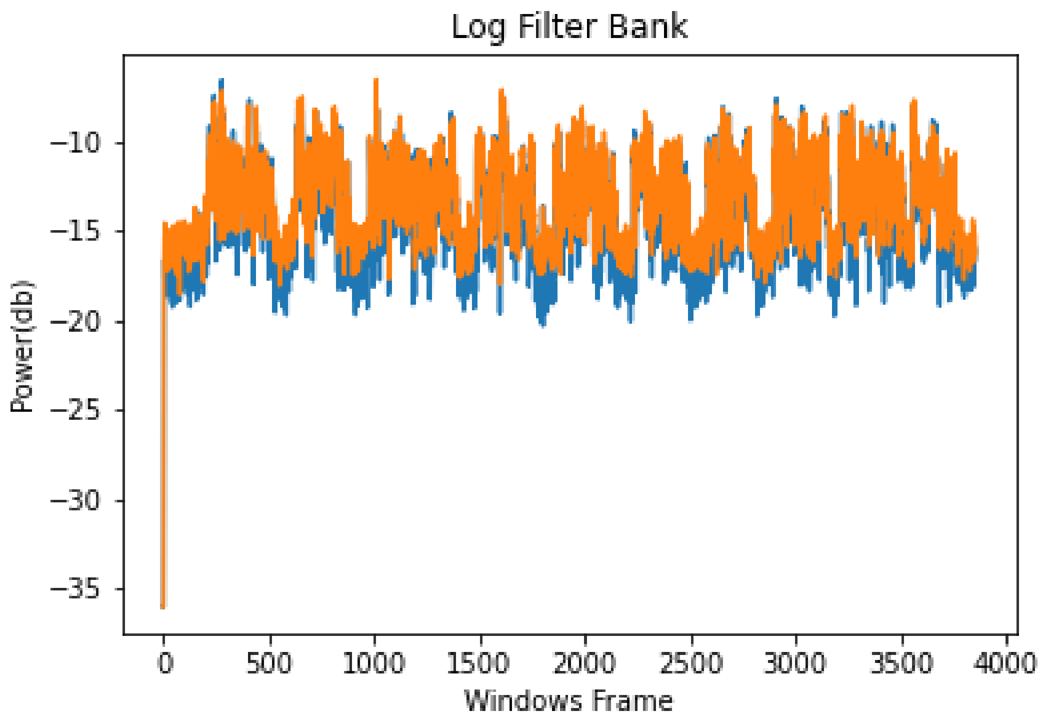

Figure 2.6: Log Filter bank representation of a randomly selected audio file.

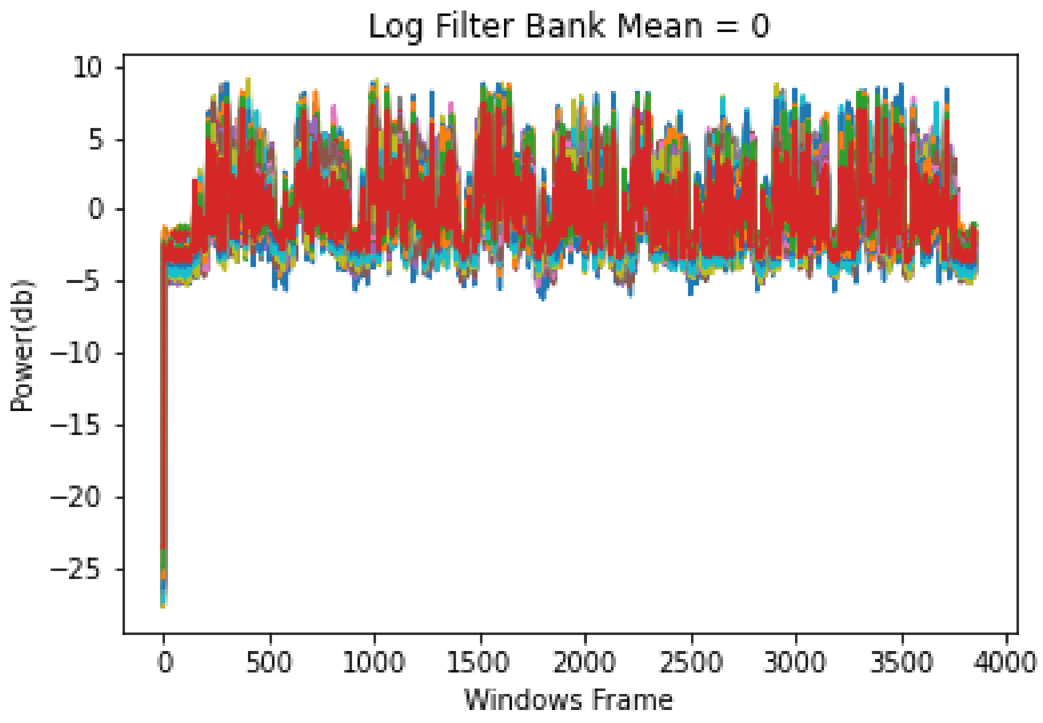

Figure 2.7: The Signal mean is shifted to the value zero.

3.1.1 Experiment 1

This is our Baseline model made up of Convolutional and Attention network. We would be comparing this experiment with other and seeking out to find any type of improvement.Three-fold Cross validation was made on the category of Audio device.This acted as a baseline model to our novel approach in order to measure how much can we improve upon. The model was trained to just focus on the device category.The dataset was used without any prior modifications.

3.1.1.1 Results

We can see in Figure 3.1 that baseline model achieved 73% accuracy on the validation folds.It took 26-27 epoch to achieve the result.The averaged accuracy was 70% on the test fold.

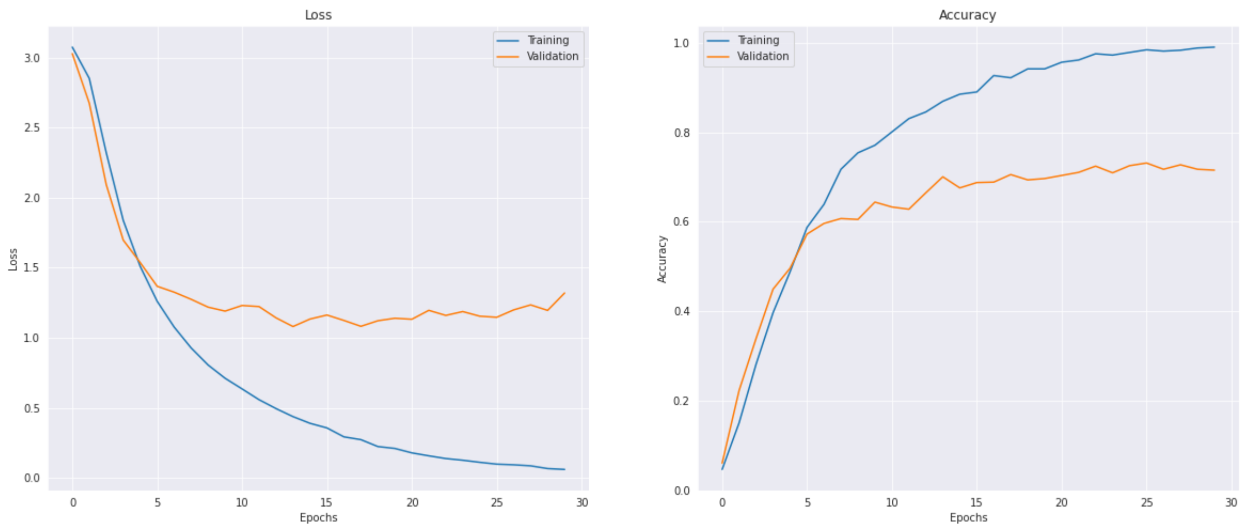

Figure 3.1: Training & Validation Loss & Accuracy Curve

3.1.2 Experiment 2

Making cross-validation based on user class which showed a very poor performance in validation accuracy as well as test accuracy. It was obvious that the Cross-validation strategy had to be applied to the device category.

3.1.2.1 Results

The model achieved 68% accuracy on the validation folds as seen in Figure 3.2. It took about all 30 epoch to achieve the result. The accuracy was 64% on the test fold. The model performed very poorly and showed a significant drop in performance by 5%.

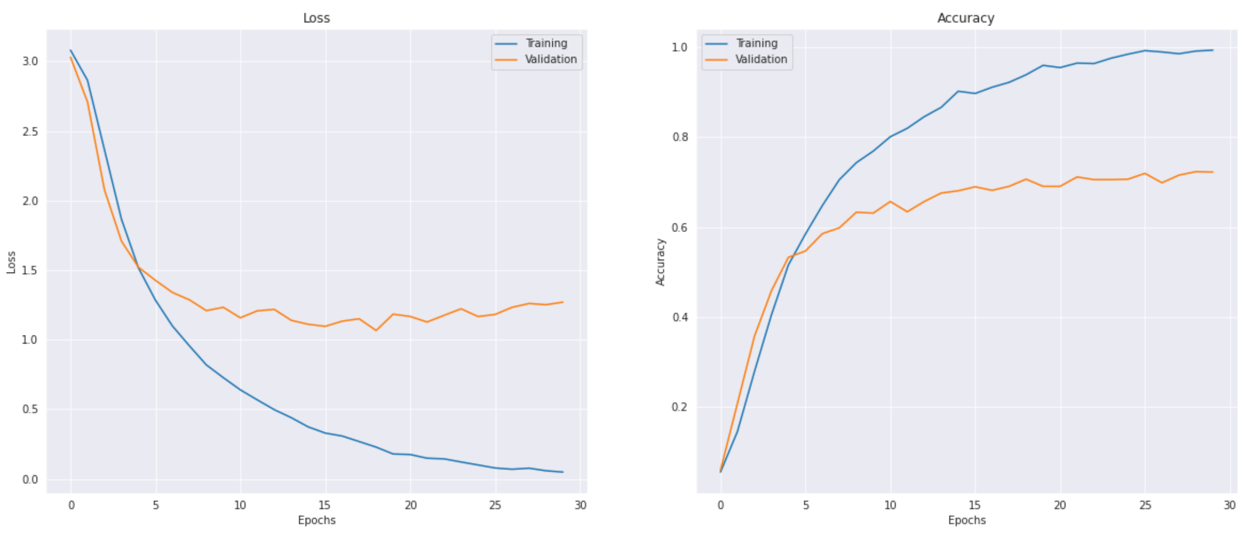

Figure 3.2: Training & Validation Loss & Accuracy Curve

3.1.3 Experiment 3

Changing the Model architecture to Auxiliary output. We also added novelty of multi-tasking to our model.

3.1.3.1 Results

Figure 3.3 show us that the model achieved 76% accuracy on the validation folds.It took just 18 epoch to reach the optimal performance. The system also achieved 77% on the test fold. We can see that change in architecture helped our model to converge faster and better.

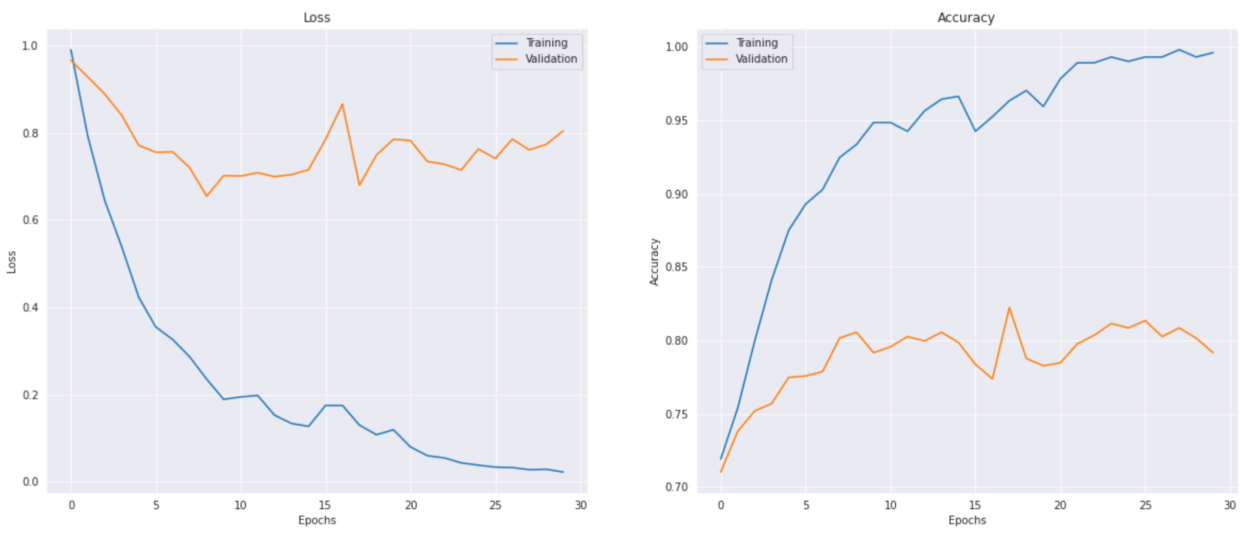

Figure 3.3: Training & Validation Loss & Accuracy Curve

3.1.4 Experiment 4

Introducing the concept of white noise to the audio sample to make the dataset more complex and model more robust.

3.1.4.1 Results

Augmenting the audio signal with white noise made the signal a bit more complex and difficult to understand.As seen in the Figure 3.4 we achieved 77% accuracy on Validation fold and gave 4% boost to our best test fold score until now. It took whole 30 epoch to reach the optimal accuracy.

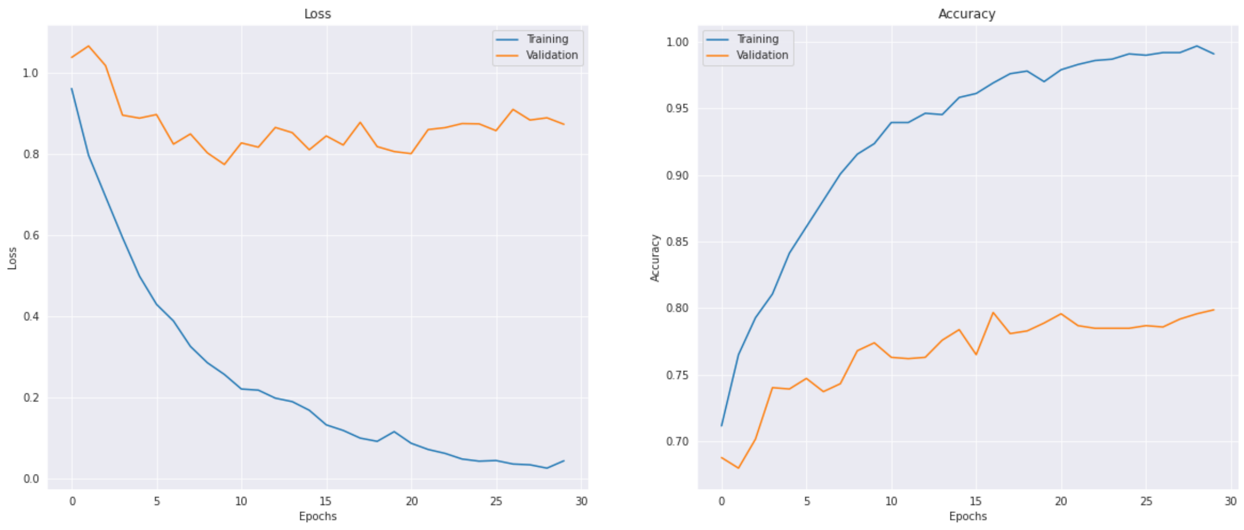

Figure 3.4: Training & Validation Loss & Accuracy Curve

3.1.5 Experiment 5

3.1.5.1 Experiment description

Changing kernel size of the Convolutional layers to a incremental form of (3,3) , (5,5) and (7,7).

3.1.5.2 Results

The model achieved 83% accuracy on validation folds as seen in Figure 3.5. It took 24 epoch to reach the optimal result. We achieved 81% accuracy on the test fold.We can say that this hyperparameter tuning of kernel size didn't yield any improvement.

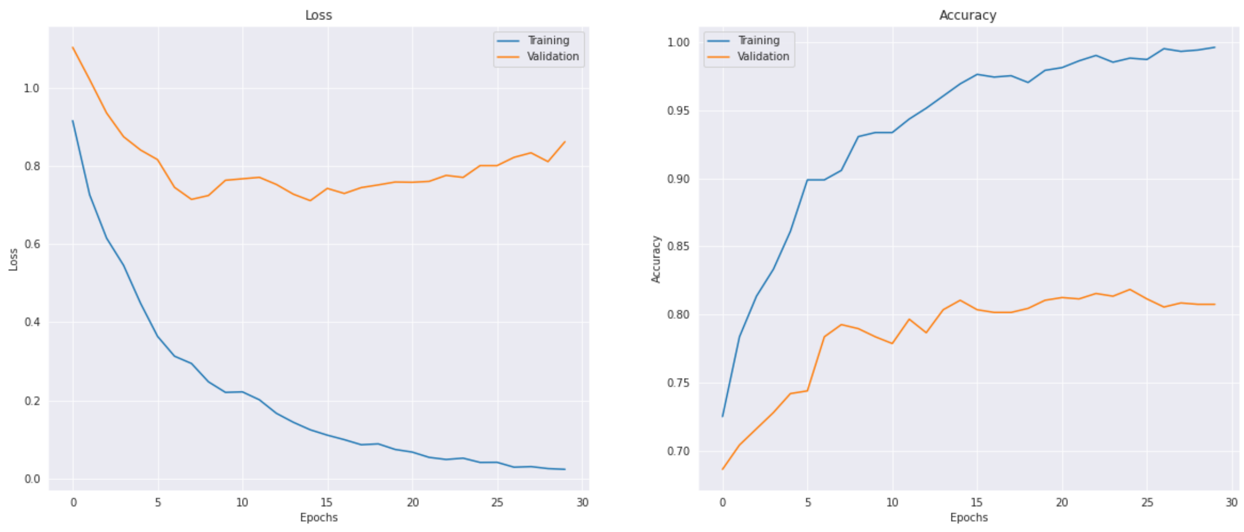

Figure 3.5: Training & Validation Loss & Accuracy Curve

3.1.6 Experiment 6

Tweaking the loss weightage proportion between audio device classification and the gender of the user and setting it to 70:30.

3.1.6.1 Results

From Figure 3.6 we can infer that model achieved 82% accuracy on validation folds. It took all 30 epoch to reach the optimal result. We achieved 80% accuracy on the test fold. We can conclude from this that hyper-parameter tuning didn't yield any significant improvement.

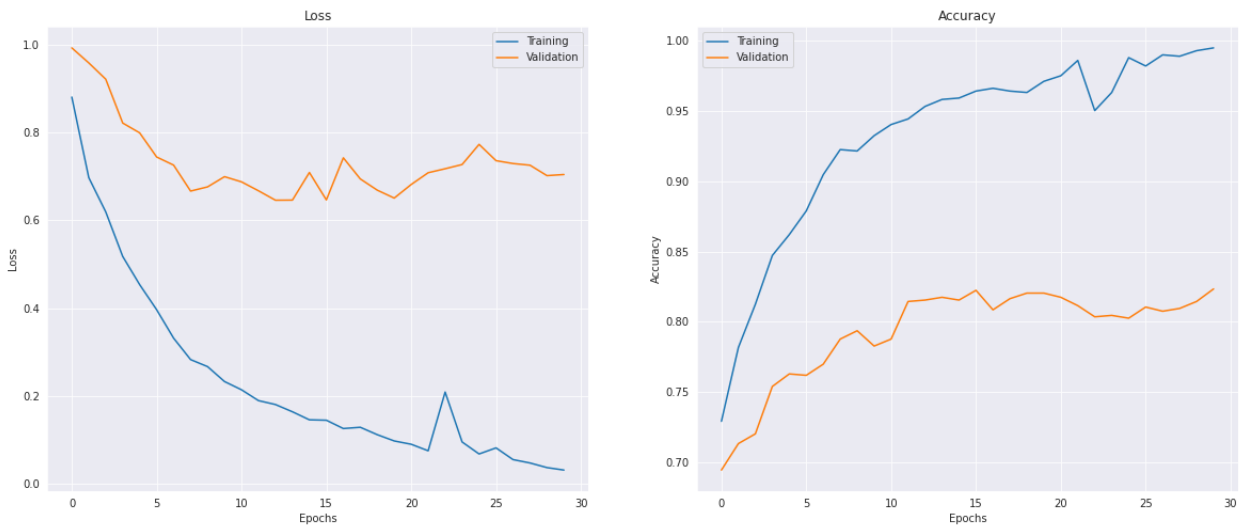

Figure 3.6: Training & Validation Loss & Accuracy Curve