Rabia Gondur

Tony Wen

CISC 6000

December 18th, 2023

The Effect of Autoencoder Audio Feature Reduction on Deepfake Audio Detection with Convolutional Neural Network

Abstract

Artificially (AI) generated audios and deepfakes pose a large threat to public security. With the fast paced developments in modern technology, everyday more people rely on their phones and computers to check their bank account, communicate with their friends and work. Therefore, reliable and efficient safety measures are needed in various platforms to prevent any potential attacks that can compromise user's safety. Recently, one of the most common cyber attacks have been deepfakes, which are hyper-realistic fakes of videos, audios, and images created with AI applications. Due to this, there are a lot of researchers that develop new tools to address this issue by creating models that can distinguish between what is real and what is AI generated. In our project, we build on the previous research that looked into this problem and extend their approaches by proposing a hybrid model. We develop a model that uses an autoencoder to extract features from transformed audio data followed by a convolutional neural network (CNN) to classify the data as real or AI generated. We compare our model with two baseline models, a regular CNN and a Feed-Forward Network in terms of their prediction accuracy and equal error rate score (EER). We show that using our hybrid model that leverages the power of autoencoders to extract richer features outperforms these two baselines and provides a more robust way of classifying audio data.

Introduction

Fake audio classification has been one of the most important tasks of safe AI movement. With the introduction of more nuanced deep learning models everyday, it gets harder for people to distinguish between what is authentic and what is fake, or AI generated. Although this is an issue in a wide range of fields such as computer vision, text generation etc. this paper will focus on audio data. Deepfake audios are used more and more to prey on people through the internet or via phone (Flitter et al. 2023; Hughes 2023; Jones 2023). Due to these serious implications, methods that explore how to prevent these attacks, or techniques that can help distinguish between genuine and fake data are of great interest to cybersecurity and ethical AI.

***Related Work**:* There is a lot of work that looks into classifying audio data to check if they are authentic or not. Hamza et al. (2022) investigated deepfake audio detection using deep neural networks (DNNs). In their research, they specifically focused on one of the components of audio data called Mel-frequency cepstral coefficients (MFCC). MFCC features are mid-level audio features, also known to capture the spectral characteristics of sound. They are often extracted using a fast fourier transform, log-amplitude spectrum, mel scaling, and discrete cosine transform. Hamza et al. (2022) used these MFCC features on the Fake-or-Real dataset, which is a benchmark dataset for audio classification, with support vector machine (SVM) and pre-trained VGG-16 model and achieved state of the art results for audio classifications compared to other neural network models. Another research by Chintha et al. (2020) investigated the deepfake detection in benchmark audio datasets as well as in video datasets to detect AI generated audio to prevent cyberattacks. They combined convolutional latent representations with bidirectional recurrent structures to extract richer and more informative features from the audio data that can

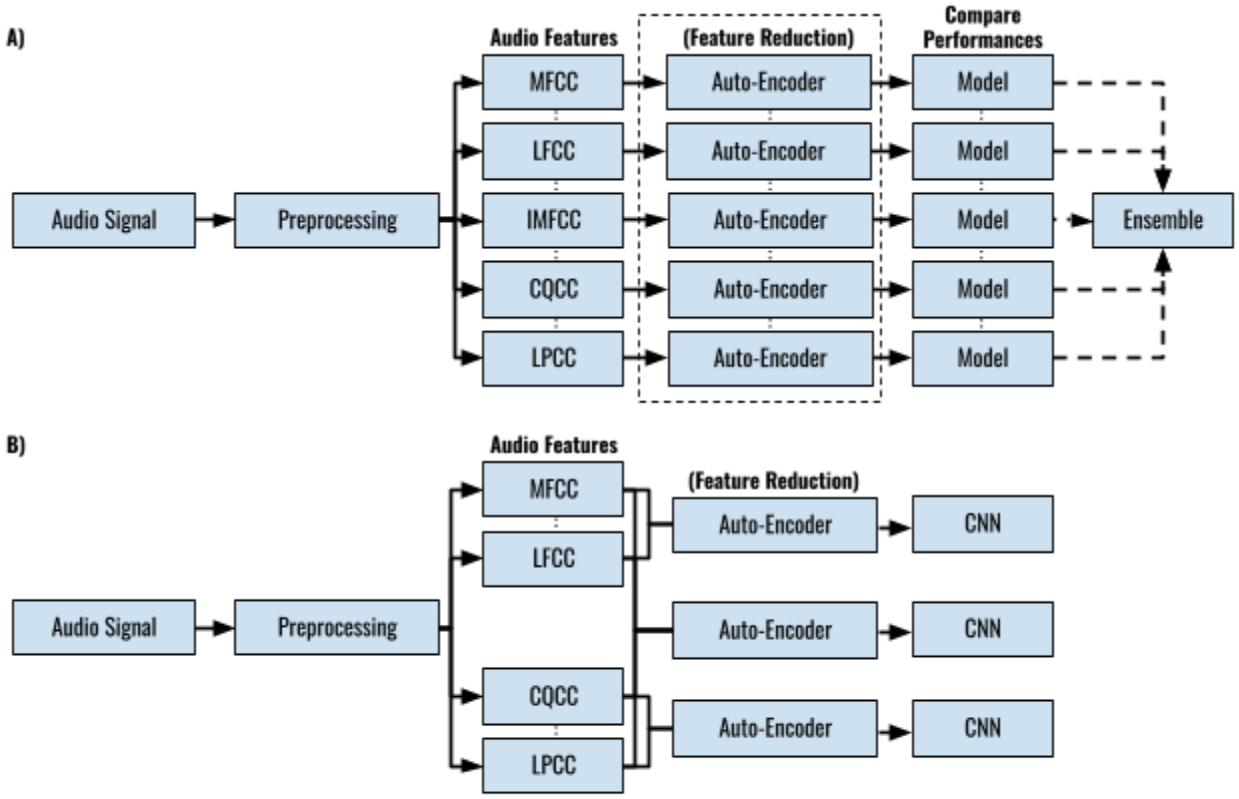


Figure 1. A) General Template of Detection System Proposed by past works. **B)** Detection System proposed by this project. Its performance will be evaluated against one without an auto-encoder and a simple feedforward network.

help neural network models distinguish real audio from deepfake audio. Another work by Wijethunga et al. (2021) again used MFCC features to classify deepfake audios specifically in text-to-speech (TTS) systems. They created a model composed of various components such as extensive pre-processing of the audio, using recurrent neural networks, and then using CNN's for final classification.

Building on this previous research, we propose a hybrid model, Figure 1.B, that combines autoencoders and CNN's for optimum performance of audio classification. Previously, most research in deepfake audio classification specifically considered one part of the audio feature, namely MFCC. Although MFCC has been found very useful in these classification tasks, there are other features of audio data that can also be utilized to detect deepfakes. Audio data

inherently consists of various features, but despite this, not all of them are utilized in neural network models for classification since they can be redundant or unnecessary. Thus, in this project we picked two features that are not as common in audio classification alongside two popular features for audio classification to train our hybrid model. Namely we picked Constant Q Cepstral Coefficients (CQCC) and Linear Frequency Cepstral Coefficients (LPCC), for the uncommon features, along with MFCC and Linear Frequency Cepstral Coefficients (LFCC), for the common features. LFCC features are largely similar to MFCC except LFCC filter bank coefficients equally cover all speech frequency ranges and give them equal importance whereas MFCC boosts higher frequencies. In addition to that LFCC uses a linear frequency scale instead of a Mel scale used in MFCC. These are both widely used spectral audio features for audio classification. On the other hand, CQCC is an extension of MFCCs that better captures pitch variations in audio signals, and LPCC is based on linear prediction analysis and it is often used for audio signal processing. Therefore both CQCC and LPCC are informative features that can also be used to classify audio, and it is important to investigate how they compare to other popular features like LFCC and MFCC.

In addition to audio features, another important contributing factor for successful deepfake classification is the model architecture. Previous work on this explored many avenues and combined different models such as CNNs and RNNs to provide robust architectures with high accuracy for audio classification. However, currently there are no deepfake classification models that combine autoencoders with CNNs to detect deepfake audio data. Due to the architecture and the use case of autoencoders, they are ideal models to compress data into its most informative components. This extraction results in richer components that can be more informative than the original data itself. Thus, we believe that training an autoencoder on our audio data and then

feeding this bottleneck learned representation to a CNN can result in better audio classification. To test this, we compare this proposed hybrid model to two baselines, a regular CNN and a Feed-Forward Neural Network. We use precision, F1-score and equal error rate to compare these three models to see which architecture results in better audio classification.

Therefore, the objective of our project is to create a hybrid model composed of an autoencoder and a CNN for more robust classification of deepfake audio, as well as compare two popular audio features with two less used features.

Methodology

Data Source: The audio data for this project is sourced from the Deepfake (DF) subset of the Speech Data provided by the Automatic Speaker Verification Spoofing and Countermeasures Challenge 2021. ASVspoof is a biennial challenge that aims to foster the development of generalized countermeasures capable of detecting various audio spoofing attacks. The DF subset comprises bona fide and spoofed speech, created using ffmpeg and sox toolkits, and processed with different lossy codecs, emphasizing detection in compressed audio used in television, news websites, and social media. Unlike logical access (LA) and physical access (PA) subsets, the DF subset excludes the use of an ASV system, and its evaluation metric is the equal error rate (EER). It combines data from ASVspoof 2019, Voice Conversion Challenge (VCC) 2018, and VCC 2020, introducing 100+ spoofing attack algorithms and addressing codec generalization across different sources. The evaluation conditions involve nine settings, exploring various codecs, bit rate configurations, and dual-codec applications. The DF subset contains 611829 audio files in flac format, sampled at a rate of 16 kHz and stored in 16-bit. Due to computational limitations, for our models we took a balanced sample of 45,200 audio files from it, where we

had 22,600 samples for each label. And we used half of the data (22,600) for training and the other half for testing.

Feature Extraction: We iterated through audio files in the DF subset, applying prepackaged feature extraction functions (MFCC, LFCC, CQCC, LPCC) from the `spafe` library to each file. All 4 functions were parameterized to: 1) pre-emphasize the audio with a coefficient of 0.97; 2) apply hamming windowing with a 25-msec window length and a 10-msec overlap; 3) apply mean-variance normalization. Once features are computed, they are stacked together in pairs (MFCC-LFCC, CQCC-LPCC) and appended to an array, along with the corresponding label indicating whether the audio is classified as 'spoof' (0) or 'real' (1) based on the metadata provided. The resulting arrays ([(n, 13, 2), (66, 13, 2), 1]) are saved in .npy format for subsequent model training.

Since we are feeding feature arrays into models that expect fixed input dimensions, we implemented 2 approaches to address the time dimension. Due to an unresolved `spafe` issue with `cqcc` and `lpcc` functions, the dimension of the resulting feature array stack CQCC-LPCC is fixed, instead of reflecting the length of the input audio as expected, which seems to clip off input audio beyond 66 frames. For standalone MFCC-LFCC inputs, we clipped off input audio beyond the frames of the shortest audio in the dataset (in this case 43 frames). For all feature inputs, we utilized `torchvision.transforms.Resize()` with anti-aliasing to resize MFCC-LFCC into (66,13,2) before stacking with CQCC-LPCC to form the final input array of shape (66,13,4)

Training Details: After feature extraction, we trained the baseline models and AE+CNN model with Adam optimizer, a learning rate of 0.0005 with 0.001 weight decay, a batch size of 64, for 50 epochs each. AE models were set to have latent vectors of length 8 and trained on mean square error loss, while FFN and CNN were trained on cross entropy loss.

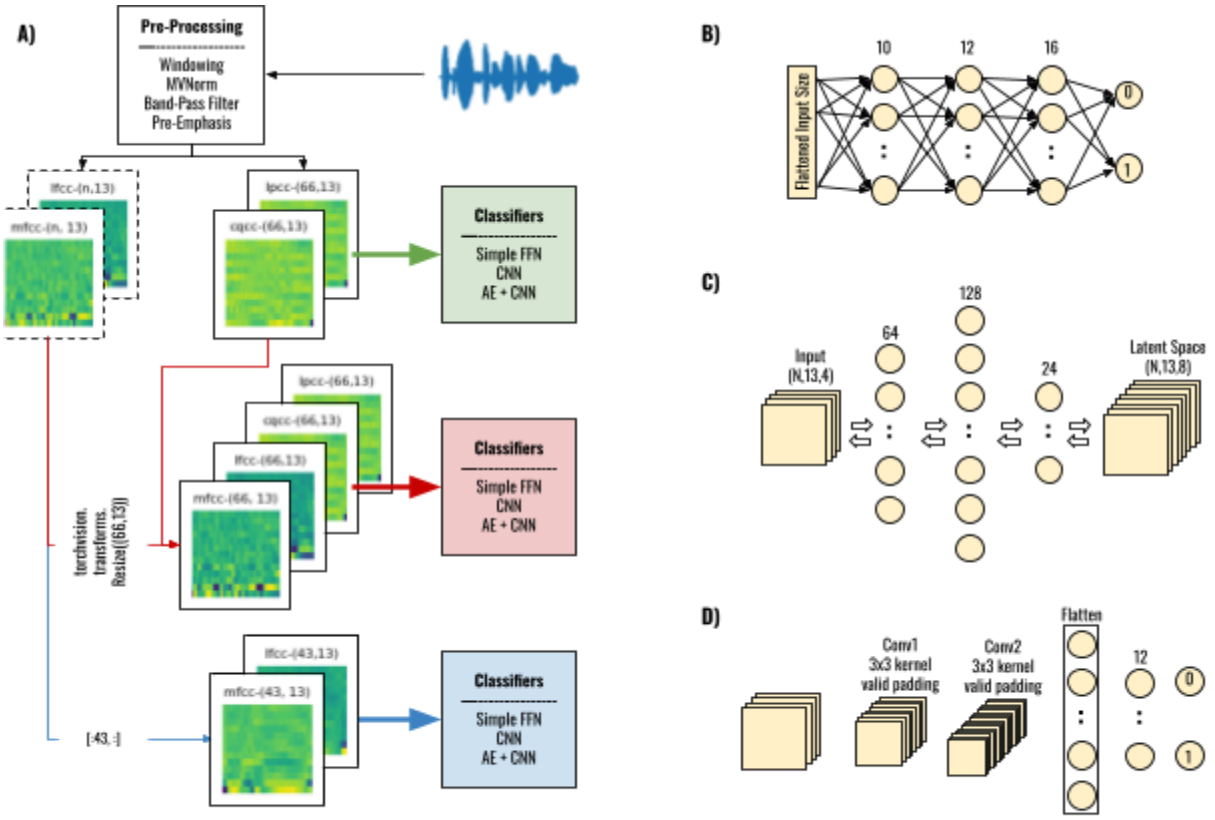


Figure 2. A) Proposed system's architectural diagram demonstrates the flow of audio samples and classifiers used; **B)** Feed Forward Network Architecture; **C)** Overcomplete Autoencoder Architecture; **D)** CNN Architecture

Results

We systematically evaluated the performance of three distinct models—Feedforward Neural Network (FFN), Convolutional Neural Network (CNN), and an Autoencoder combined with CNN (AE+CNN)—within the domain of audio classification for deepfake detection. Our investigation focused on understanding the impact of different feature sets, encompassing MFCC, LFCC, CQCC, and LPCC, on the models' efficacy.

Across key performance metrics — Test Accuracy, Equal Error Rate (EER), F1 Score, and Precision — we observed notable trends (Table 1). The AE+CNN model, when trained on the comprehensive feature set with all 4 features, consistently outperformed its counterparts.

Achieving a test accuracy of 0.9156, this model demonstrated a superior ability to correctly classify both deepfake and authentic audio instances. The AE+CNN model's exceptional performance extended to the EER metric, where it exhibited the lowest rate at 9.57%. This outcome underscores the model's capacity to strike a balanced trade-off between false positives and false negatives, a crucial aspect in the context of deepfake detection. The F1 Score, representing the harmonic mean of precision and recall, reflected the AE+CNN model's ability to maintain a favorable balance between these two metrics. With an F1 score of 0.9156, the model showcased robust performance in mitigating both types of errors. Furthermore, when trained on all features, the model yielded the highest precision values at 88.22%, emphasizing its proficiency in reducing false positive instances.

		Test Accuracy	Equal Error Rate	F1-Score	Precision
FFN	mfcc-lfcc cqcc-lpcc ALL	0.8308 0.7987 0.8676	18.21% 22.87% 13.61%	0.8308 0.7982 0.8676	0.8314 0.7987 0.8676
CNN	mfcc-lfcc cqcc-lpcc ALL	0.8145 0.8393 0.8691	22.90% 18.36% 14.02%	0.8133 0.8391 0.8691	0.8228 0.8393 0.8694
AE + CNN	mfcc-lfcc cqcc-lpcc ALL	0.8816 0.8682 0.9156	13.19% 15.55% 9.57%	0.8816 0.8680 0.9156	0.8822 0.8682 0.9159

Table 1: Compare different composition of features as input for classification for all models

Collectively, these findings position the AE+CNN model, leveraging diverse feature sets, as a promising architecture for deepfake audio classification. The observed effectiveness holds implications for enhancing security measures on platforms hosting media content, such as news websites and social media platforms, where the detection of manipulated audio content is of paramount importance.

***Limitations**:* Although the results show promising EER scores and precision, the model training is not very stable. One of the possible methods to aid in this is normalization prior to loading the data to the model, however we did not find significant changes in the stability of the training and the precision and EER scores were negatively affected by this change. Therefore, we opted out of normalizing the data and just worked with the pre-processed data which despite the unstable training provided good results. One way to overcome this in future work is to increase the data size and fine-tune the model further to ensure that there is a stable training scheme for the model. Another limitation was the computational limitation where due to expensive storage of the full audio data, we had to use < 4% of the total data. Although we ensured that the dataset was balanced, the model would benefit from having even more examples and we would predict that the precision and error scores would reflect this.

Discussion

AI generated audio poses a significant risk to public security and user privacy. Algorithms and models that target this issue are of great interest to deep learning and cybersecurity. Despite the easy availability of large datasets of AI and human generated audio, as well as the wide success of many models, the current approaches are limited. Although many state of the art models provide high accuracy for deepfake audio classification, they do so with little novelty by using the already established common audio features and architectures. In our work we address these limitations by using two different audio features in addition to two most widely used features to analyze their exploratory strength in detecting deep fake audio. We also propose a novel hybrid model that uses the rich representations from an autoencoder as inputs to CNN to classify a given

audio as fake or real. We show that using our hybrid approach achieves higher precision and lower EER compared to two baselines. In addition to this, we show that using lesser used audio features are also informative for audio classification. These results open a new avenue for exploration in deep learning for the expressive power of different audio features for classification as well the strength of autoencoders for extracting richer representations of the data that are more informative for deepfake classification tasks.

References

Flitter, E., & Cowley, S. (2023, August 30). *Voice deepfakes are coming for your bank balance*. The New York Times.

Hughes, Alex. (2023). *AI: Why the next call from your family could be a deepfake scammer.* BBC Science Focus.

Hamza, A., Javed, A. R. R., Iqbal, F., Kryvinska, N., Almadhor, A. S., Jalil, Z., & Borghol, R. (2022). Deepfake audio detection via MFCC features using machine learning. *IEEE Access*, *10*, 134018-134028.

Jones, V. A. (2020). *Artificial intelligence enabled deepfake technology: The emergence of a new threat* (Doctoral dissertation, Utica College).

Liu, X., Wang, X., Sahidullah, M., Patino, J., Delgado, H., Kinnunen, T., Todisco, M., Yamagishi, J., Evans, N., Nautsch A., & Lee, K. A. (2023). ASVSPOOF 2021: Towards spoofed and Deepfake Speech Detection in the wild. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 31,* 2507-2522. https://doi.org/10.1109/taslp.2023.3285283

Balamurali, B. T., Lin, K. E., Lui, S., Chen, J. M., & Herremans, D. (2019). Toward robust audio spoofing detection: A detailed comparison of traditional and learned features. *IEEE Access, 7, 84229-84241.* https://doi.org/10.1109/access.2019.2923806

Mcuba, M., Singh, A., Ikuesan, R. A., & Venterm, H. (2023). The effect of deep learning methods on Deepfake audio detection for digital investigation. *Procedia Computer Science, 219*. 211-219, https://doi.org/10.1016/j.procs.2023.01.283

Adiban, M., Shehnepoor, S., Sameti, H. (2020). Replay spoofing countermeasure using autoencoder and siamese networks on ASVSPOOF 2019 Challenge. *Computer Speech & Language, 2019.* https://doi.org/10.1016/j.csl.2020.101105

Westerlund, M. (2019). The emergence of deepfake technology: A review. Technology Innovation Management Review, 9 (11), 39–52.

Yamagishi, Junichi, et al. "ASVspoof 2021: accelerating progress in spoofed and deepfake speech detection." *arXiv preprint arXiv:2109.00537* (2021).

Appendix

Autoencoder + CNN (architecture and training logs)

Autoencoder

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─Linear: 1-1                            [-1, 8, 43, 13, 64]       192
├─Linear: 1-2                            [-1, 8, 43, 13, 128]      8,320
├─Linear: 1-3                            [-1, 8, 43, 13, 24]       3,096
├─Linear: 1-4                            [-1, 8, 43, 13, 8]        200
├─Linear: 1-5                            [-1, 8, 43, 13, 24]       216
├─Linear: 1-6                            [-1, 8, 43, 13, 64]       1,600
├─Linear: 1-7                            [-1, 8, 43, 13, 128]      8,320
├─Linear: 1-8                            [-1, 8, 43, 13, 2]        258
==========================================================================================
Total params: 22,202
Trainable params: 22,202
Non-trainable params: 0
Total mult-adds (M): 0.02
==========================================================================================
```

CNN

```
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
├─Conv2d: 1-1                            6,208
├─ReLU: 1-2                              --
├─Conv2d: 1-3                            1,450
├─ReLU: 1-4                              --
├─MaxPool2d: 1-5                         --
├─Flatten: 1-6                           --
├─Linear: 1-7                            24,492
├─ReLU: 1-8                              --
├─Linear: 1-9                            26
==========================================================================
Total params: 32,176
Trainable params: 32,176
Non-trainable params: 0
==========================================================================
```

Autoencoder training

```
Epoch 1/50, Loss: 0.10930981002491975
Epoch 6/50, Loss: 0.1841591283805394
Epoch 11/50, Loss: 0.158506048638019
Epoch 16/50, Loss: 0.17039167822218693
Epoch 21/50, Loss: 0.166948741564507 9
Epoch 26/50, Loss: 0.08920069098687179
Epoch 31/50, Loss: 0.1679673668716889
Epoch 36/50, Loss: 0.14229424586320383
Epoch 41/50, Loss: 0.1111861703890213
Epoch 46/50, Loss: 0.11937469005396499
```

CNN training

```
Epoch 1/50, Loss: 0.31720268726348877
Epoch 6/50, Loss: 0.3990677297115326
Epoch 11/50, Loss: 0.26995688676834106
Epoch 16/50, Loss: 0.3855403959751129
Epoch 21/50, Loss: 0.573445737361908
Epoch 26/50, Loss: 0.2889520525932312
Epoch 31/50, Loss: 0.2625783383846283
Epoch 36/50, Loss: 0.4008801579475403
Epoch 41/50, Loss: 0.2381124645471573
Epoch 46/50, Loss: 0.04518934339284897
```

CNN (architecture and training logs)

CNN

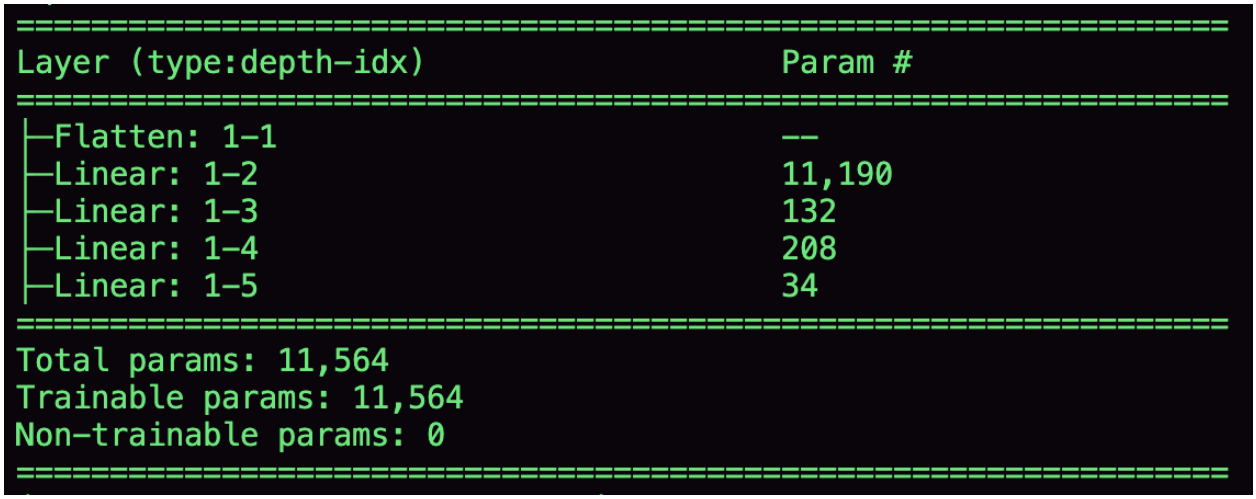
```
==========================================================================
Layer (type:depth-idx)                                   Param #
==========================================================================
├─Conv2d: 1-1                                           6,208
├─ReLU: 1-2                                             --
├─Conv2d: 1-3                                           1,450
├─ReLU: 1-4                                             --
├─MaxPool2d: 1-5                                        --
├─Flatten: 1-6                                          --
├─Linear: 1-7                                           12,252
├─ReLU: 1-8                                             --
├─Linear: 1-9                                           26
==========================================================================
Total params: 19,936
Trainable params: 19,936
Non-trainable params: 0
==========================================================================
```

Training

```
Epoch 1/50, Loss: 0.6252338886260986
Epoch 6/50, Loss: 0.35097986459732056
Epoch 11/50, Loss: 0.18804916739463806
Epoch 16/50, Loss: 0.2577013373374939
Epoch 21/50, Loss: 1.0649923086166382
Epoch 26/50, Loss: 0.5399150848388672
Epoch 31/50, Loss: 0.4685262441635132
Epoch 36/50, Loss: 0.5074021816253662
Epoch 41/50, Loss: 0.6510556936264038
Epoch 46/50, Loss: 0.35824263095855713
```

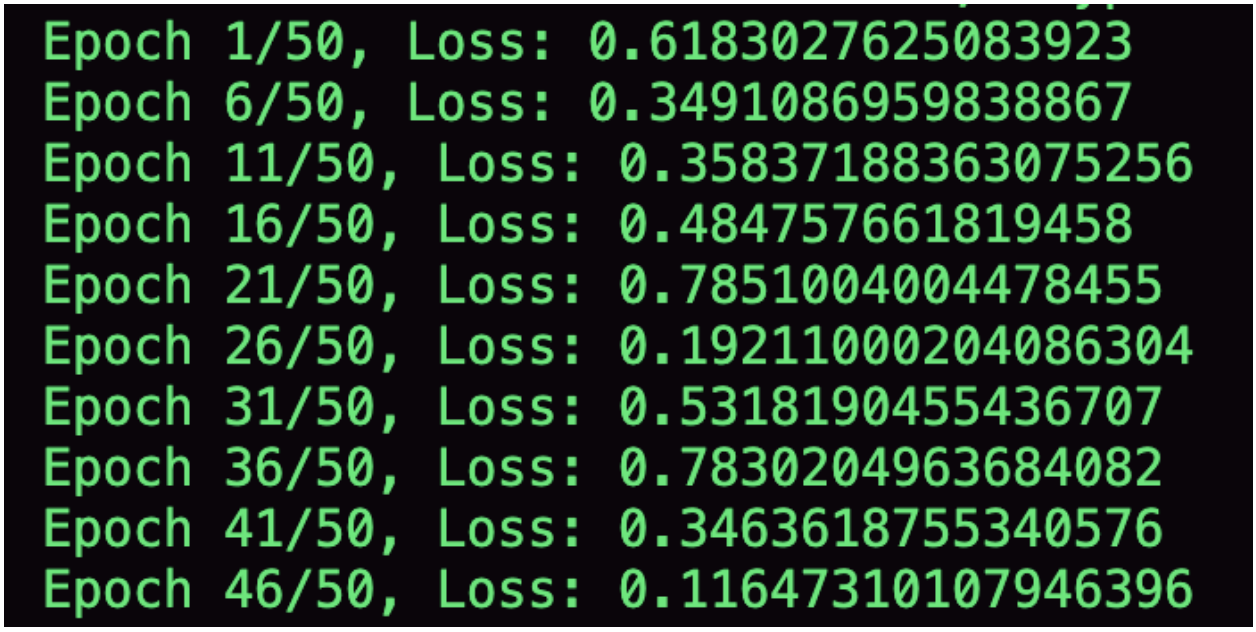
Feed Forward Neural Network (architecture and training logs)

Feed Forward Neural Network



```
==========================================================================
Layer (type:depth-idx)                             Param #
==========================================================================
├─Flatten: 1-1                                    --
├─Linear: 1-2                                     11,190
├─Linear: 1-3                                     132
├─Linear: 1-4                                     208
├─Linear: 1-5                                     34
==========================================================================
Total params: 11,564
Trainable params: 11,564
Non-trainable params: 0
==========================================================================
```

Training



```
Epoch 1/50, Loss: 0.6183027625083923
Epoch 6/50, Loss: 0.3491086959838867
Epoch 11/50, Loss: 0.35837188363075256
Epoch 16/50, Loss: 0.484757661819458
Epoch 21/50, Loss: 0.7851004004478455
Epoch 26/50, Loss: 0.19211000204086304
Epoch 31/50, Loss: 0.5318190455436707
Epoch 36/50, Loss: 0.7830204963684082
Epoch 41/50, Loss: 0.3463618755340576
Epoch 46/50, Loss: 0.11647310107946396
```

The effect of increasing the training set on model performance:

		Test Accuracy	Equal Error Rate (EER)	F1-Score	Precision
FFN	mfcc-lfcc cqcc-lpcc	0.8690 0.8579	13.23% 17.00%	0.8690 0.8576	0.8690 0.8604
CNN	mfcc-lfcc cqcc-lpcc	0.8544 0.8593	16.61% 15.31%	0.8543 0.8588	0.8558 0.8588
AE+CNN	mfcc-lfcc cqcc-lpcc	0.9232 0.9176	11.43% 11.86%	0.9232 0.9174	0.9273 0.9214

Table 2: Using MFCC-LFCC, CQCC-LPCC features for classification for all three models with 80% training 20% testing