URBAN SOUND CLASSIFICATION

Deep Learning

DEMOKRITOS
NATIONAL CENTRE FOR SCIENTIFIC RESEARCH

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ
UNIVERSITY of the PELOPONNESE

Vince Ranches
2022202304002
dit2302dsc@go.uop.gr
George Raptis
2022202304023
dit2323dsc@go.uop.gr

June 26, 2024

Contents

List of Figures

1 UrbanSound8k Dataset

This dataset contains 8732 labeled sound excerpts (¡=4s) of urban sounds from 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer, siren, and street_music. The classes are drawn from the urban sound taxonomy. All excerpts are taken from field recordings uploaded to freesound.org.

1.1 File Handling

First of all, we flattened the directory where the official sound was, we split our files into train and test folders where we kept the Train and Test CSVs respectively with the metadata included. Next at the train test folder, we separated the files into folders where each folder represents one of the classes of the dataset.

1.2 Feature Extraction

We extracted features in two ways.

- Firstly we extracted MFCC features using PyAudioAnalysis library, with mid-step = 1 and short step = 0.1. We did this to train a simple Fully connected classifier.
- Extracted mel-spectrogram using librosa default Hop-Size and Window-Size and Min Sample rate = 42.000 from Deep Audio Features library. We ended up with a spectrogram of size (128,81) if something was smaller than 81 it was 0-padded so everything was equal in size.

1.3 Fully Connected Training

You can see the architecture we chose to use ReLU as the activation function.

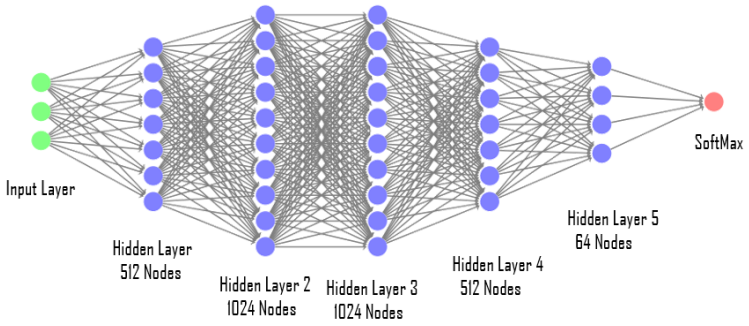

Figure 1: Fully Connected Architecture

The model was trained with these characteristics.

- Dropout: 0.2
- Learning Rate : 10^{-3}
- Batch Normalization
- Epochs: 159
- Early stopping based on Accuracy
- Patience 15
- Batch Size: 32

We chose that big of patience because the model has a slight increase which we were losing if we had a patience of 5. Also, it is important to state that we kept the best model and not the one we ended with after 15 epochs of patience.

1.4 Results

- Accuracy = 90.5%
- F1_Score = 90.5%

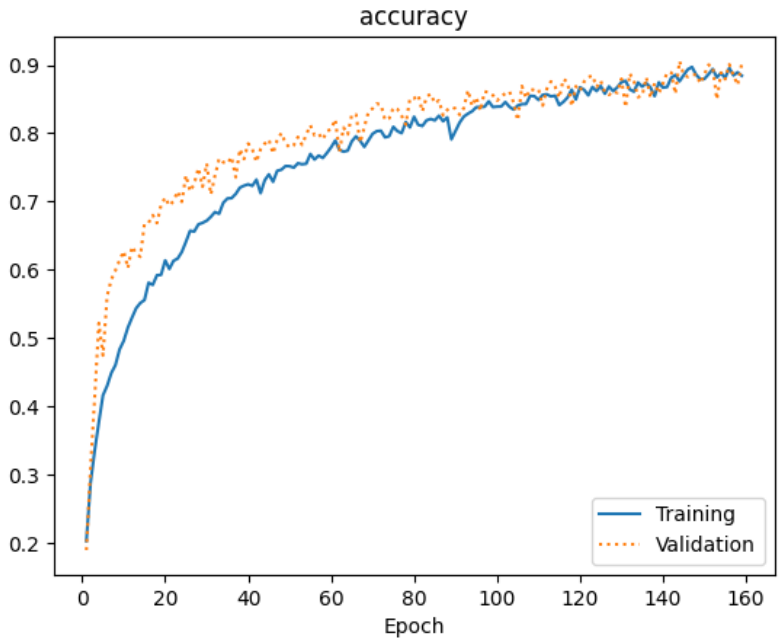

Figure 2: Accuracy Curve

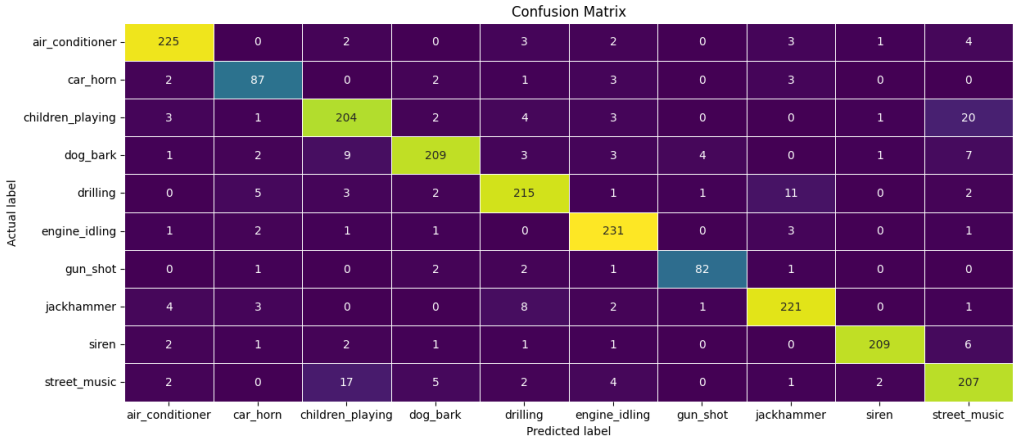

Figure 3: Confusion Matrix

1.5 Comments

As we can see a very simple Fully Connected model has pretty good results even with minimal sound prepossessing which means our data are of very good quality and with a big baseline of 90% percent accuracy we don't expect a huge accuracy increase in the rest of our models.

1.6 CNN Training

Next, we used the spectrograms to train a simple CNN classifier the architecture we used followed a pretty well-known state-of-the-art architecture like ResNet.

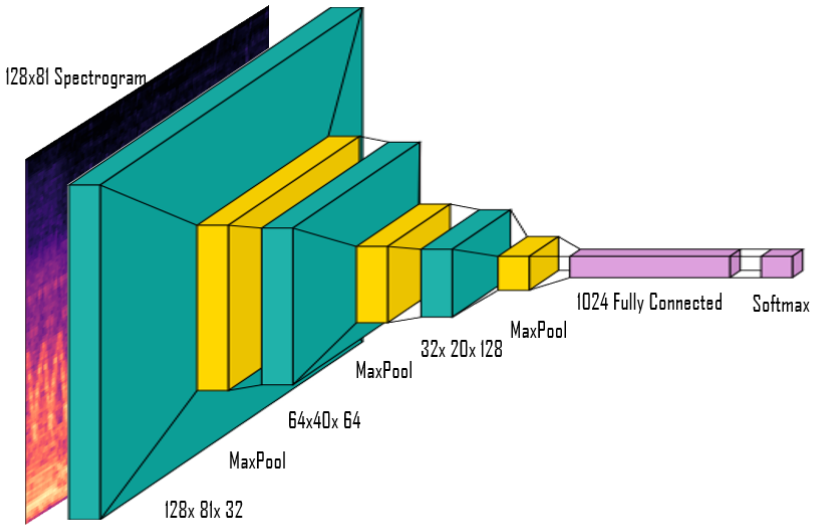

Figure 4: CNN Architecture

The model was trained with these characteristics.

- 0-padding
- Dropout: 0.2
- Learning Rate : 10^{-3}
- Batch Normalization
- Epochs: 50
- Early stopping based on Accuracy
- Patience 15
- Batch Size: 32

1.7 Results

- Accuracy = 91.1%
- F1_Score = 91.1%

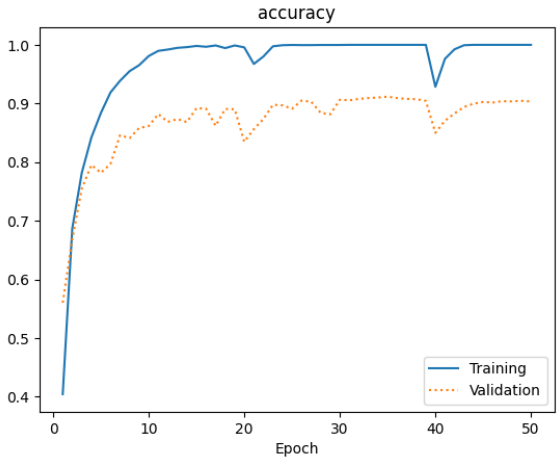

Figure 5: Accuracy Curve

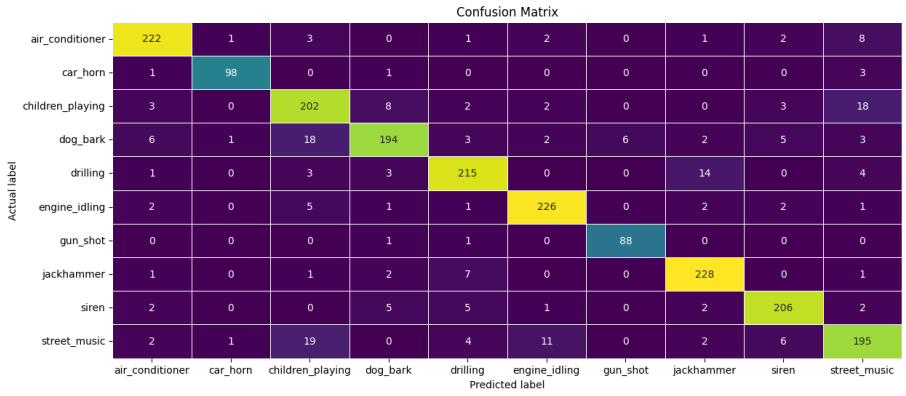

Figure 6: Confusion Matrix

1.8 Comments

What we can see is a slight improvement of 0.5% and through our testing, the feature that boosted our model was zero-padding. We kept these models as baseline and we next tried to use methods like transfer learning from a bigger model to boost our performance.

2 AudioSet

AudioSet consists of an expanding ontology of 632 audio event classes and a collection of 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds. For our purposes, we utilize the balanced_train and eval segments, which offer a balanced distribution of 632 classes with approximately 50 samples each in both the training and evaluation datasets. Each of these segments contains around 22,000 clips.

2.1 Dataset Processing

2.1.1 Downloading

At this point we want to mention that downloading the dataset was very time consuming and troublesome one of the main reasons was that the the API needed was not available. In the Audioset Pipeline Notebook in our GitHub repo you can find details and instructions on how to download it.

2.1.2 Files Processing

The next step was extracting the labels for each sound using the ontology.json file, which can be downloaded from the official AudioSet website. The challenge was that performing multi-class classification with over 600 labels is quite difficult, so we decided to group these classes into super categories.

We initially tried clustering the audio segments based on their labels using NLP methods, by extracting embeddings and clustering them into N categories. However, our results didn't match the effectiveness of the labels provided by GPT when we asked it to do the same job. Therefore, we used the labels generated by GPT.

Even with these methods, we still encountered a lot of mistakes and class imbalance. Due to time pressure, we proceeded with this approach and focused on tuning our model to learn "something" meaningful.

2.2 CNN on Audioset

The model architecture was very much like our previous model but this time we doubled the layer to better catch the complexity of the sounds. Maybe an even more complex architecture could have better results but we did not have the resources to train a model like this.

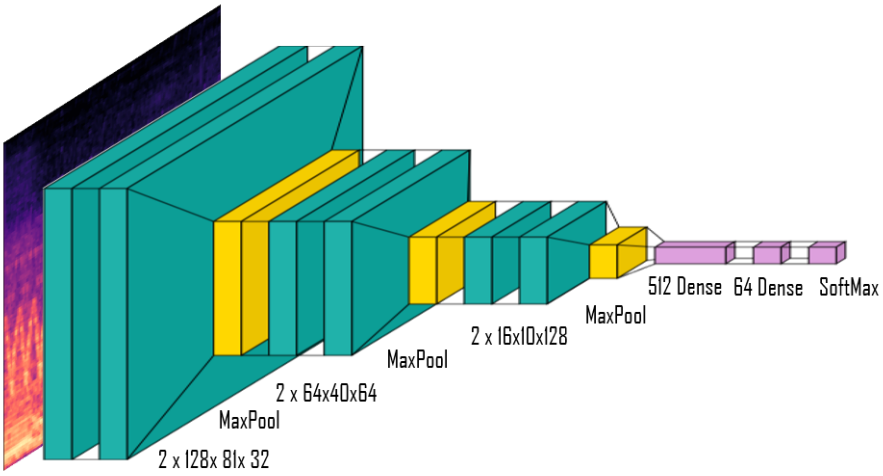

Figure 7: CNN Audioset Architecture

The model was trained with these characteristics.

- 0-padding
- Dropout: 0.2 (0.3 on Fully Connected)
- Learning Rate : 10^{-3}
- Batch Normalization
- Epochs: 61
- Early stopping based on F1-Score
- Patience 10
- Batch Size: 32

2.3 Results

- Accuracy = 48.8%
- F1_Score = 47.1%

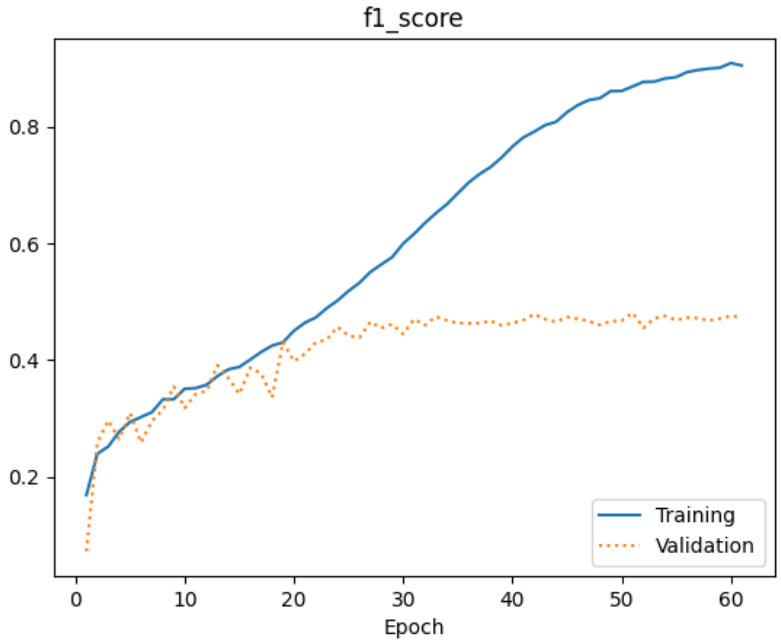

Figure 8: F1 Curve

Figure 9: Confusion Matrix

2.4 Comments

As can be observed from the confusion matrix, the results are quite poor. This is likely because the sounds contain many overlapping labels, and we are not addressing the original problem of the 600-class classification for which this dataset was designed. Additionally, some labels are certainly incorrect, as we noticed during a quick check.

Nevertheless, the model likely learns broad categories like Speech, Music, and Other. Despite its limitations, we can transfer the knowledge from this model to other tasks, as it was trained on significantly more data than the UrbanSound8k dataset.

3 Transfer Learning

3.1 Used Model Trained on AudioSet

What we did now and after a lot of testing was to freeze the first two layers of the model trained on AudioSet drop all the others and train a model of a similar architecture of our first CNN model for comparison to be trained on UrbanSound8k.

3.2 Architecture

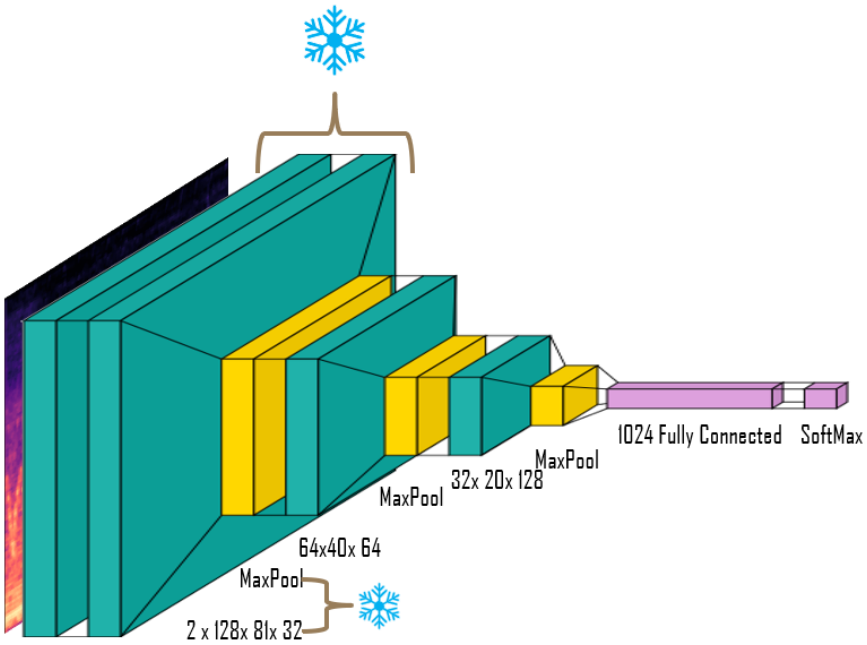

Figure 10: CNN Audioset Architecture

The model was trained with these characteristics.

- 0-padding
- Dropout: 0.2
- Learning Rate : 10^{-3}
- Batch Normalization
- Epochs: 32
- Early stopping based on F1-Score
- Patience 10
- Batch Size: 32

3.3 Results

- Accuracy = 93.1%
- F1_Score = 93.1%

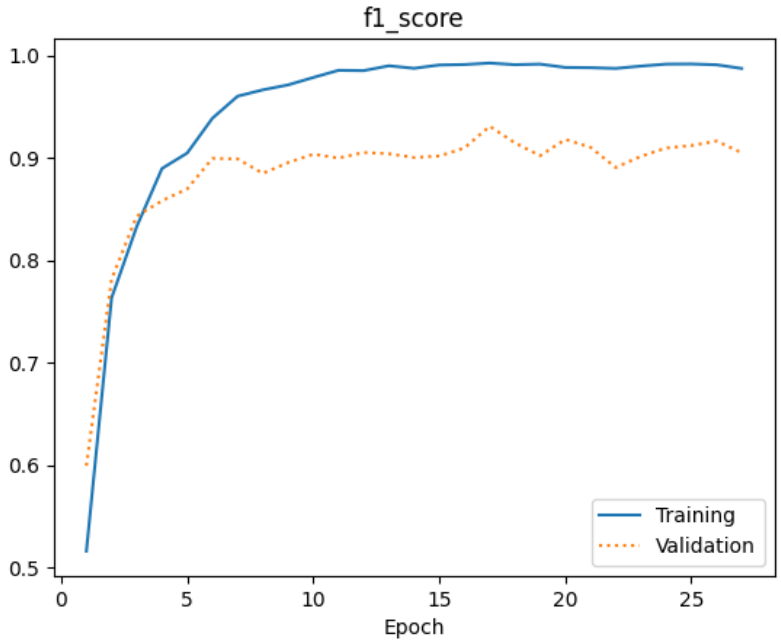

Figure 11: F1 Curve

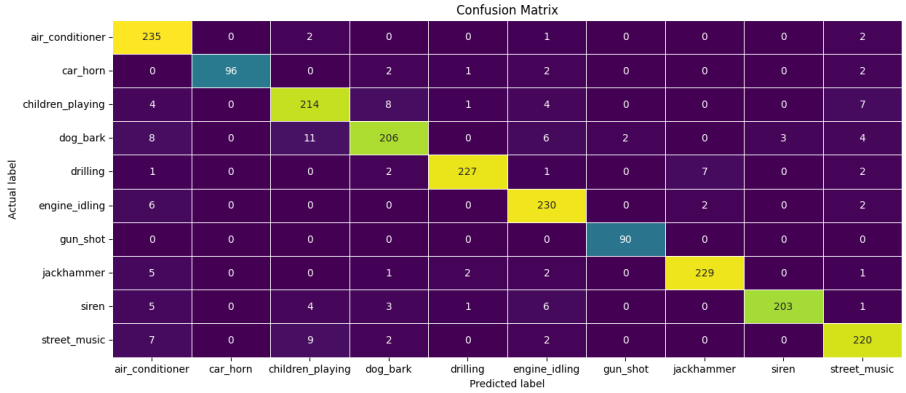

Figure 12: Confusion Matrix

3.4 Comments

We can observe that even with a very bad model like the previous one we had a boost in accuracy. This happens because the first Convolutional Layers learns meaningful generic information and the more the data the model is trained on the more info they provide. We can see with almost the same architecture as our first CNN we had 2% better results which is quite a thing if we consider our simplest model is at 90%

3.5 YamNet Transfer Learning

YAMNet is an audio classifier that categorizes audio waveforms into 521 classes from the AudioSet ontology. It uses MobileNet v1 and processes waveforms segmented into frames. Outputs include scores for each class, embeddings for feature extraction, and a log mel spectrogram for visualization.

YAMNet generates embeddings, represented as a float32 Tensor of shape (N, 1024). These embeddings capture semantic features extracted from the input audio frames. The embeddings are computed using an average-pooling method, which aggregates features to feed into the final classification layer of YAMNet.

We put our sound file as input in the model and we obtained the embedings. We next averaged these embeddings and ended up in the form of (1,1024) embedding for each file. Then we used these embeddings as input in a super simple Fully Connected neural network and we extracted the class.

3.6 YamNet Architecture

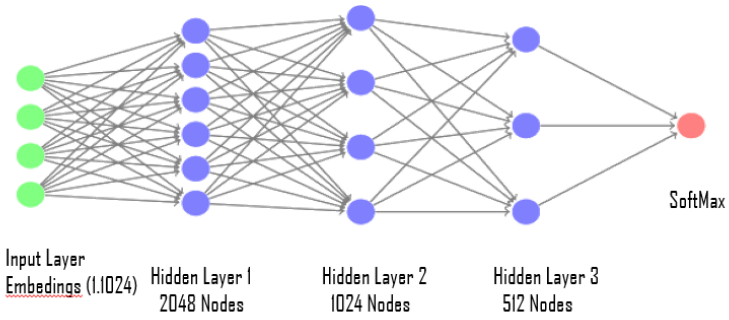

Figure 13: YamNet Fully Connected Architecture

The model was trained with these characteristics.

- Dropout: 0.2
- Learning Rate : $5 * 10^{-4}$
- Batch Normalization
- Epochs: 42
- Early stopping based on F1-Score
- Patience 15
- Batch Size: 32

3.7 Results

- Accuracy = 94.2%
- F1_Score = 94.2%

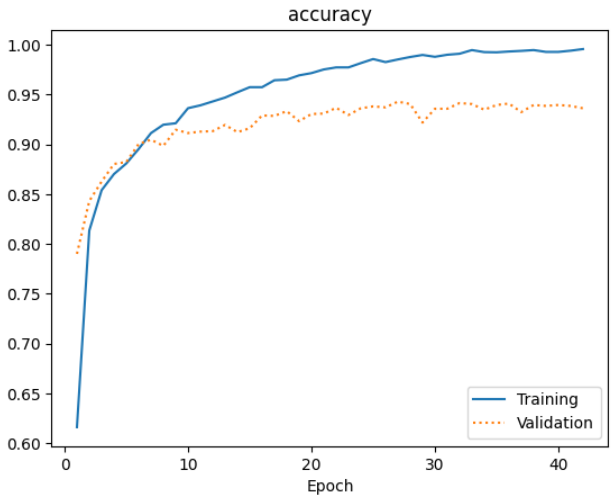

Figure 14: F1 Curve

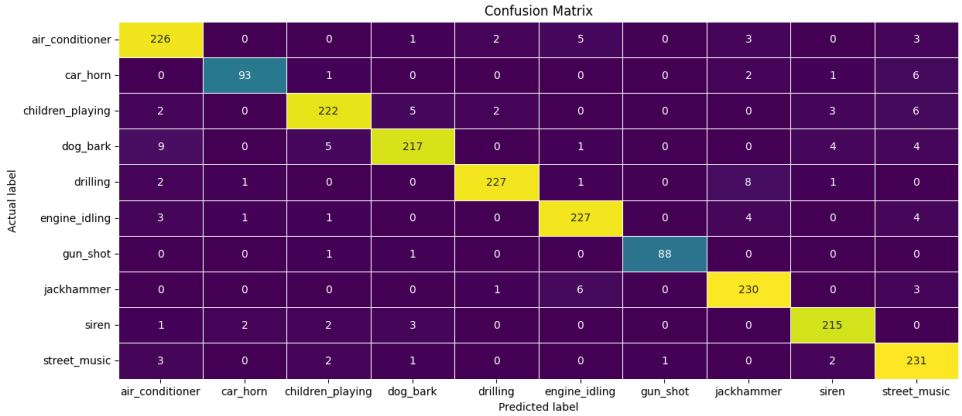

Figure 15: Confusion Matrix

3.8 Comments

As we expected this model produces the best result. We first tried just one 10024 fully connected layer but after 4-5 tests this architecture had better results maybe a simple SVM could give us back the same results but we did not try that.

4 Conclusion

	Training Params	Non Trainable
Fully Connected	2,210,634	0 (6,272 Batch Norm)
CNN	21,075,466	0
CNN on Audioset	10,806,952	0 (896 Batch Norm)
Transfer Learning	21,075,530	9376 (+Rest you see on Jupyter are Batch Norm)
YamNet Embeddings	4,989,962	0

Figure 16: Train Parameters

	Accuracy	F1_Score	Training Time
Fully Connected	0.9056	0.9057	Fast
CNN	0.9116	0.9113	Fast
Transfer Learning	0.9312	0.9313	Medium
YamNet Embeddings	0.9427	0.9427	Very Fast

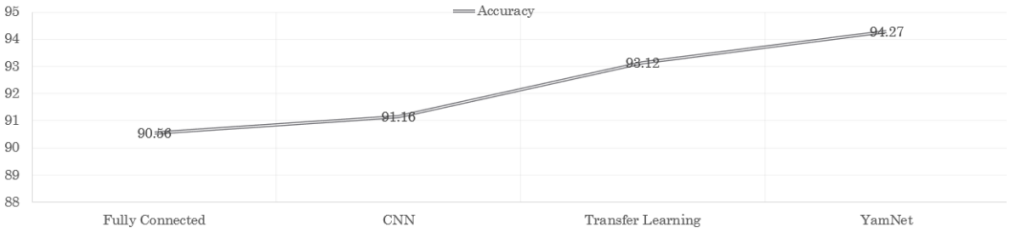

Figure 17: Results

In conclusion, we want to mention that the dataset needs to be evaluated using 10-fold cross-validation in the predefined folds to be comparable to other implementations. In our case, we used a simple train-test split of 60/20/20, so the only valid comparison is between these four models.

What should be taken from this project is that a model trained on a variety of extensive data can significantly improve the main task. From the graph above, we can see the progress in performance achieved with different implementations. In the end, we achieved a good accuracy of 93%, which is comparable to YamNet, the model we used as the upper bound.

5 Future Work

There are some details that we think could boost our performance. One of these is processing the Audioset dataset correctly by using 2-second segments instead of whole 10-second clips. By using majority voting for the final results, we could extract more meaningful information from the dataset and achieve better knowledge transfer to our task.

6 Resources

You can find all our code in this GitHub Repo.

You can find all the resources needed in this Google Drive.