AML Challenge II: Anomalous Sound Detection

EURECOM

Group 7: Emerson CARDOSO, Sameer HANS, Deepika SIVASANKARAN, Gaganjot SHAN

1. Introduction

The challenge aims to build a model that can be used to predict whether an unknown sound (this unknown sound belongs to one of the given machine IDs in the training dataset) is normal or anomalous. Anomalous sound detection provides a way to identify faulty machines and help alleviate the problem right then, to fix the machines

2. Data exploration

The development dataset contains 2370 train samples and 1101 test samples of the 3 machine IDs. The evaluation dataset contains 2370 train samples and 834 test samples of the 3 machine IDs.

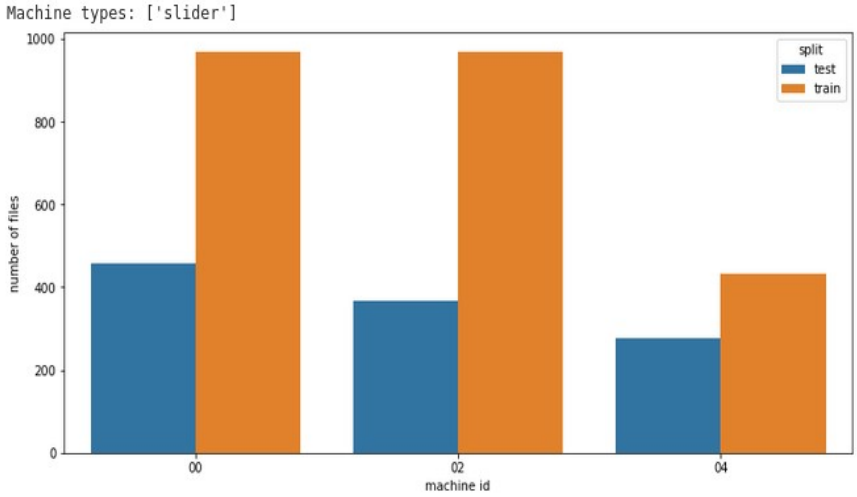


Fig 1. Data Distribution (development dataset)

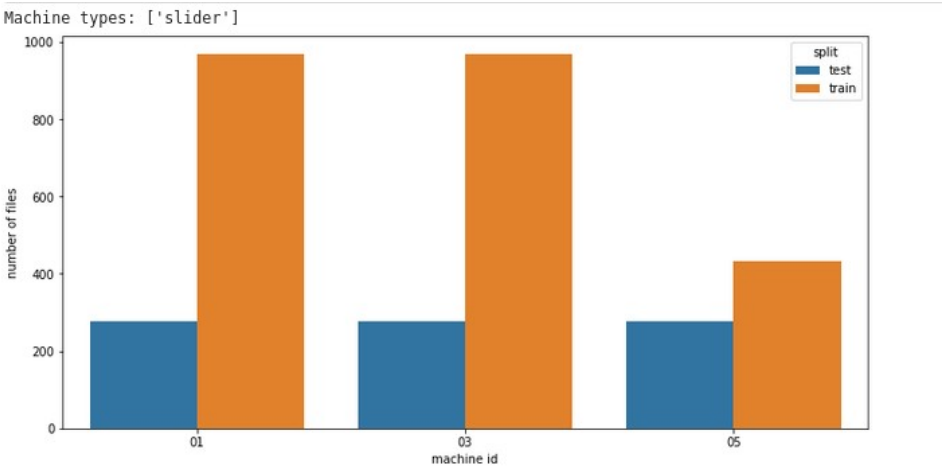


Fig 2. Data Distribution (evaluation dataset)

3. Data preparation

The code provided is calculated first, short-term Fourier transform (STFT) of the audio waveform is computed using Hann window. The frame length and the shift length are 1,024 and 512, respectively.

The STFT spectrogram is then converted to a log Mel-spectrogram with the 128-dimensional Mel basis. Finally, the log Mel-spectrogram is scaled to have zero mean and unit variance using the statistics over the training data. The Mel-Spectrograms are fed to the various models to train on.

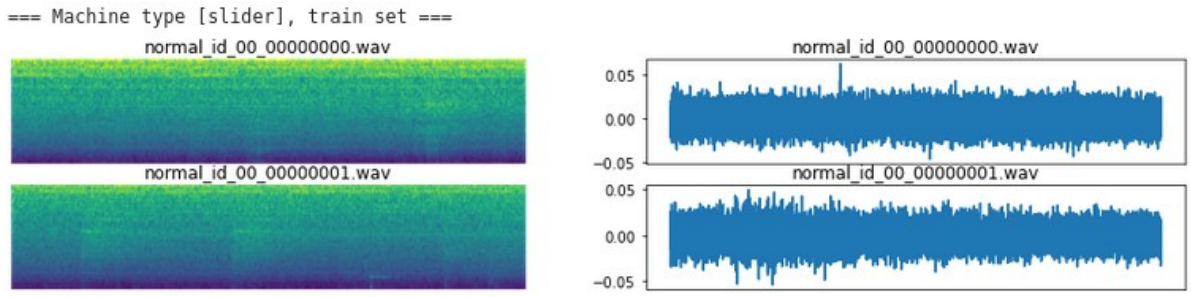


Fig 3. Mel-Spectrogram of Normal audio

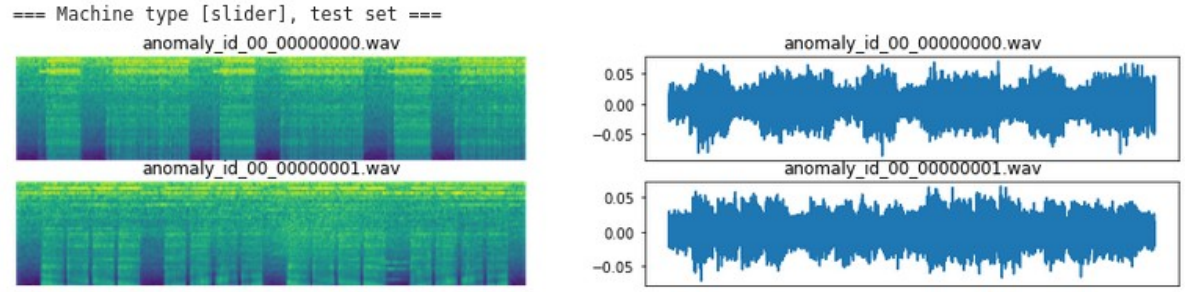


Fig 4. Mel-Spectrogram of Anomalous audio

From Fig 3-4, it can be observed that anomalous audio signals contain instances of "glitches" unlike the smooth representation of that of a normal audio signal.

4. Network

In this Challenge we tried four different approaches to boost the performance of our models, the first architecture that we tried is a feed-forward auto encoder which we called the baseline, with the baseline model we try introduce the Batch Normalization after the linear layer and before activation function for decoder and encoder and we called this model Baseline_Batch_Norm, then we introduce also the dropout layer (keeping the BatchNorm) for the first linear layer of encoder and decoder and we call this model Baseline_Dropout, our fourth try was a simple fully connected architecture with 9 blocks (Dense – BatchNorm – Activation), referred to as Sequential model. An alternate method was also tried by adding more layers

Below we show in more detail the architecture of the models used.

a) Baseline

Layer (type)	Output Shape	Param #
Linear-1	[-1, 400]	256,400
Linear-2	[-1, 400]	160,400
Linear-3	[-1, 20]	8,020
Linear-4	[-1, 20]	8,020
Linear-5	[-1, 400]	8,400
Linear-6	[-1, 400]	160,400
Linear-7	[-1, 640]	256,640

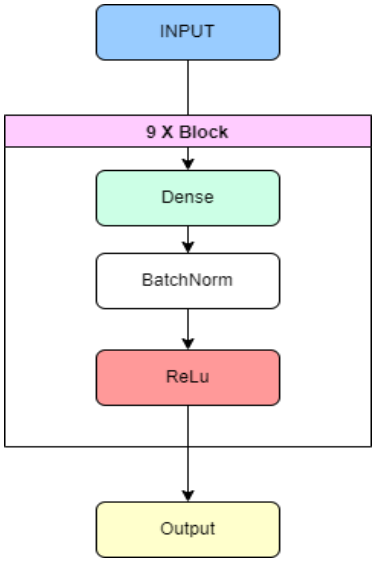
b) Baseline + BatchNorm

Layer (type)	Output Shape	Param #
Linear-1	[-1, 400]	256,400
BatchNorm1d-2	[-1, 400]	800
Linear-3	[-1, 400]	160,400
BatchNorm1d-4	[-1, 400]	800
Linear-5	[-1, 20]	8,020
Linear-6	[-1, 20]	8,020
Linear-7	[-1, 400]	8,400
BatchNorm1d-8	[-1, 400]	800
Linear-9	[-1, 400]	160,400
BatchNorm1d-10	[-1, 400]	800
Linear-11	[-1, 640]	256,640

c) Baseline + Dropout

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                 [-1, 400]         256,400
       BatchNorm1d-2                 [-1, 400]             800
           Dropout-3                 [-1, 400]               0
            Linear-4                 [-1, 400]         160,400
       BatchNorm1d-5                 [-1, 400]             800
            Linear-6                  [-1, 20]           8,020
            Linear-7                  [-1, 20]           8,020
            Linear-8                 [-1, 400]           8,400
       BatchNorm1d-9                 [-1, 400]             800
          Dropout-10                 [-1, 400]               0
           Linear-11                 [-1, 400]         160,400
      BatchNorm1d-12                 [-1, 400]             800
           Linear-13                 [-1, 640]         256,640
================================================================
```

d) Sequential Model



5. Experiment and Results

The introduction of the Batch normalization on the Baseline model leads to a significant increase in the model performance,

IDs	00		02		04		AVG	
Models	**AUC**	**pAUC**	**AUC**	**pAUC**	**AUC**	**pAUC**	**AUC**	**pAUC**
Baseline	0.957	0.795	0.763	0.617	0.910	0.673	0.877	0.695
Baseline + BatchNorm	0.961	0.810	0.776	0.639	0.944	0.7356	0.894	0.728
Baseline + Dropout	0.966	0.835	0.779	0.640	0.929	0.685	0.891	0.720
Sequential Model	0.967	0.837	0.790	0.631	0.923	0.700	0.893	0.723

Below are the attached pAuC and AuC graphs to visualise the values.

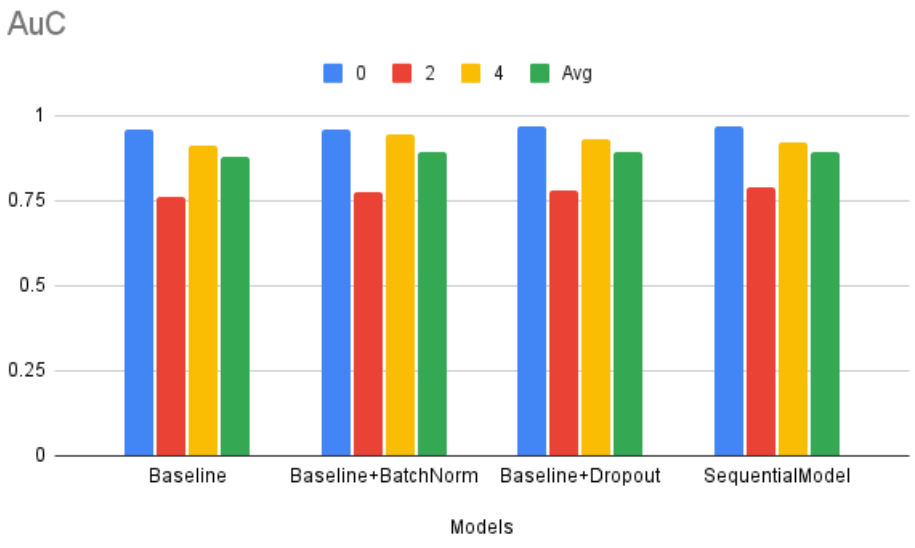


Fig 5. AuC scores for the dev data

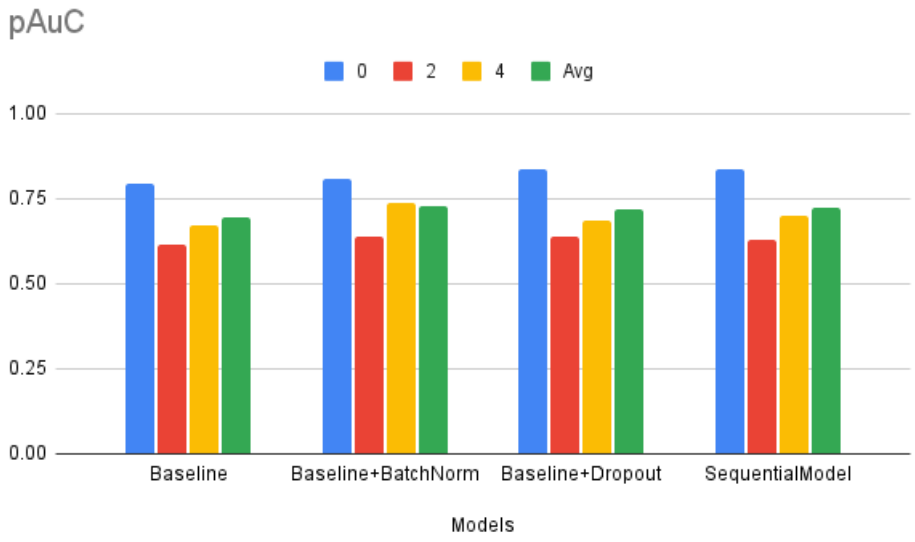


Fig 6. pAuC Scores for the dev data

We trained our baseline models using 100 epochs, for the optimizer function we use Adam with the learning rate of 0.001 and the batch size of 512 and the sequential model was trained using the same parameters for 200 epochs.

MODEL	LEARNING RATE	EPOCH	BATCH SIZE
Baseline	0.001	100	1000
Baseline+BatchNorm	0.001	100	512
Baseline+Dropout	0.001	100	512
Sequential	0.001	200	256

6. Conclusion

In conclusion, we found the tensorflow sequential model performing better than the rest. Addition of layers or increasing the epochs for training did not improve the performance beyond. Furthermore, improvement can be achieved by fine-tuning the parameters.