AML - Anomaly Detection - Challenge 2
Group 4

Philipp Peron
philipp.peron@eurecom.fr

Khristen Thornburg
khristen.thornburg@eurecom.fr

Hugo Rechatin
hugo.rechatin@eurecom.fr

Ioannis Panagiotis Pitsiorlas
ioannis.pitsiorlas@eurecom.fr

1. Introduction

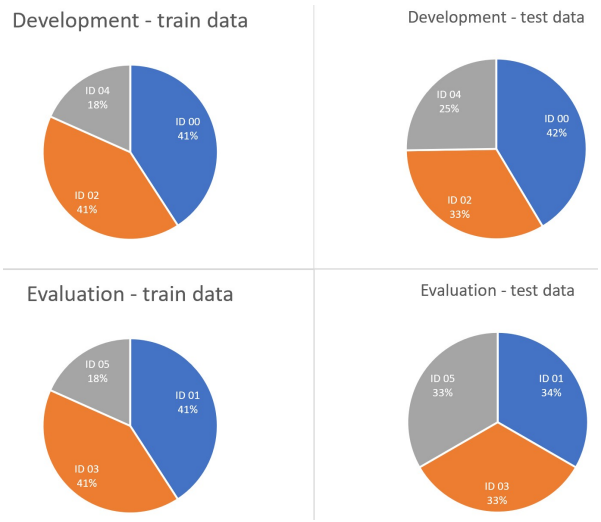
From ensuring the safety of combustion engines to replacing a server before it fails, predicting machinery failure before it occurs can save innumerable amounts of money, time and even human lives. Since far more data is being collected every second than can possibly be reviewed manually, Machine Learning approaches are needed to predict possible anomalies or failures. The goal of this challenge is to analyze recordings from one type of machine, a slider, and be able to predict if the recording represents normal operation or an anomaly.

We began our analysis by converting the data to the log mel spectrograms and performing data augmentation by taking N sideways crops of the images. After preprocessing, we considered several models : Arcface MobileNets, Amazon Autoencoder, and a Convelutional Neural Network. Ultimately, the model with the best performance was a CNN AE which had an AUC of 0.895 on the evaluation dataset.

2. Data Analysis

When considering the data, we noticed that for both the development and evaluation datasets, the third machine (ID 04 in Development or ID 05 in Evaluation) only represented 18% of the training samples but between 25% and 33% of the testing samples. Essentially, only half as many samples from the third machine were used in training compared to the other two machines while the third machine still represents about a third of the testing samples. We hypothesized that this could cause a model to be biased toward the first two machines while having more trouble correctly identifying anomalies vs normal sounds in the third machine. Figure 1 shows the percent of samples from each machine in training and testing for both the development and evaluation datasets.

Figure 1. Percent of samples by Machine ID



3. Feature Engineering

3.1. Data Bias

In order to address the model potentially being biased toward identifying anomalies in the first two machines, one approach we used was to do under-sampling in order to balance the different machine IDs. For example, in the evaluation data all three machines were split evenly in the testing. Since ID 05 only represented the smallest number of samples in the training data (434 compared to 968 for the other two machine IDs), we randomly selected 434 samples from ID 01 and ID 03 and used these for training alongside all of the ID 05 samples. While this does lose some of the information from the other samples, this was the simplest was to ensure that all three machines were weighted equally. This increase the AUC score by 0.003 but was not very significant overall. For the final model, we did not use this technique and instead used regularization to push for generalization.

3.2. Data Augmentation & Preparation

3.2.1 Spectrogram Preparation & Implicit Data Augmentation

There were two degrees to our data augmentation. The first was an implicit augmentation done by our mel spectrogram preparation. As we chose to use a convolutional AE (see section 4.3.), we wanted to crop the base spectrograms (split them along the time axis) to train the model on these smaller inputs. Because $N_c * W! = mel_W$ (with N_c the number of crops, W their width and mel_W the spectrogram width) we made sure that if the last crop overstepped, its starting point was set back to have it fit in the spectrogram.

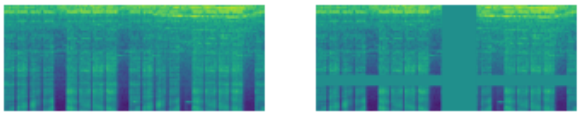
This implies that if $N_c * W - mel_W = a$ with $W > a > 0$, the last 2 crops will have a/W of their crops overlapping. For that reason, the spectrogram croping can lead to unwanted data augmentation that will bias the model towards the end of the recording. When trying different preprocessing parameters, we made sure that $N_c * W - mel_W < W$ to avoid creating identical crops.

As we worked on our final model, we came to the conclusion that the smaller crops ($W = 16$) would lead to training sets that were too big to fit in a reasonable time (especially when parameter tuning would require multiple passes) and the bigger crops($W = 64$) generally lead to poor performance. Our best performing models were obtained using $N_c = 9$ and $W = 32$.

3.2.2 Time/Frequency masks

Because melspectrograms cannot be augmented in the same way as regular images (rotation, translation and mirroring transformations change the information) we turned to this paper [2] which detailed a simple implementation that uses time and frequency masks to create new samples. By creating different policies that defined time/frequency mask widths and number of time/frquency masks we were able to generate new audio samples directly from the melspectrogramm space. Figure 2 shows an example of the original spectrogram and the spectrogram with a mask applied.

Figure 2. Application of time and frequency masks



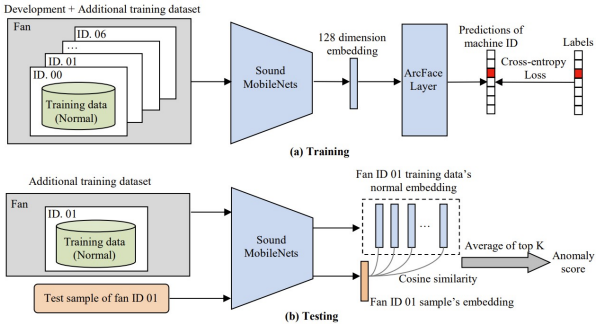
Furthermore, we added the option to generate multiple samples from a base spectrogram to see up to what point we could push the augmentation before the model performance would start falling due to overfitting. We realized that having more than two spectrograms generated per image resulted in the model overfitting. Having one generated spectrogram alongside the original image gave a slight improvement in model performance.

4. Model Selection

4.1. Arcface MobileNets

The first model we tried was based on a technical report from the DCASE 2020 task 2 challenge [3]. In the competition, the group reached an AUC score of 99.99% on the slider test samples. The fundamental idea behind the approach is to train a neural network to predict the machine ID of a training sample. The network is based on the MobileNetV2. After the supervised training, the classification head is replaced with a distance measure. In order to get the anomaly score of a test sample, the distance between the neural network's embedding output of the test sample and the training data output is calculated. The closest 10 training samples are used to calculate an average anomaly score for each test sample. The arcface loss is used as classification loss during training. This approach has lead to great success in face recognition with deep learning as well as in sound applications. Figure 3 shows more details of this technique.

Figure 3. Visualization of Arcface MobileNet approach [3]



Working with a paper that has no published code and having to implement everything ourselves was much more difficult than we expected. During training, the loss of the machine ID prediction converged really well. But when we calculated the distance measure from the test samples to the training samples, there was no detectable difference in the distance of samples with anomalous sounds and normal sounds. This resulted in a maximum AUC score of 0.53. Unfortunately, we were not able to identify the problem that lead to these disappointing results. When implementing the paper we had to make a number of assumptions since the report was not very technically detailed. We hypothesize that one of these assumptions could potentially be the problem with our model.

4.2. Amazon Autoencoder

Another model that we tried is an autoencoder that was implemented first by Amazon Web Services for sound anomaly detection [1]. First, we tried to extract the spectrograms as tabular features, by dividing the spectrogram of each signal into several sliding windows. After that, we concatenated all the windows into a single feature matrix with the features associated to each signal. Then we fed into the autoencoder the tabular-shaped features. In order to create the autoencoder model, we implemented 3 layers on the decoding part, the input layer with the dimensions of our input features and after, 2 Dense layers of 64 neurons concluding on the latent space layer with dimension 8. The encoder part was made by 2 Dense layers of 64 neurons, leading to the last Dense layer that has also the input dimensions. As a loss function we used Mean Squared Error, and for the optimizer we used Adam. During the training the loss was converging well, but when we tried to predict based on the test set the model did not perform well and achieved an AUC score of 0.55.

4.3. CNN [submission model]

Being well adapted to image processing, we naturally thought of building a Convolutional Neural Network. Once the wave files were converted to spectrograms, they could be treated as image data and passed into a CNN autoencoder. Because we trained the model on time-wise crops of the training spectrograms, we loaded the testing data in the same fashion and did our prediction on these individual crops. With this approach, our model could learn to differentiate normal spectrograms from abnormal ones at a more granular timescale. We then determine the reconstruction error of a given sample by selecting the mean/max/min of the reconstruction errors on the crops generated by said sample.

We worked overall on 2 networks, one being quite shallow and processing spectrograms generated with $n_mels = 128$, the other mounted with additional convolutional layers and deeper filters processing spectrograms generated with $n_mels = 256$.

For weight initialization, a Zero initializer was set, which lead to quite poor performances. We then found in the literature that the Xavier initialization was the benchmark when it came to autoencoders. We did end up with better performance compared to the conventional Normal initialization.

After creating these two networks, we started parameter tuning. We simply iterated through the various hyperparameters we jugged most important, namely :

- Mel_count the frequency resolution of the mel spectrograms
- $latent\_dimension$ of the AE
- $BatchSize$ and $epochs$

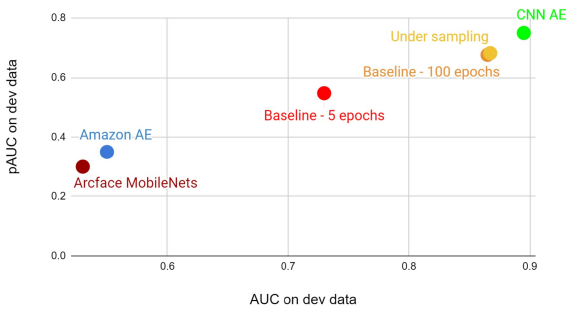
Table 1 shows the resulting AUC score with different combinations of these parameters. In the end, the parameters that gave the best score were in trial 2 and gave a score of 0.895. Since we by no means tried all different combinations of the hyperparameters, it is probable that these could be tuned even further to acheive an even higher AUC score.

5. Conclusion and Further Work

Figure 4 shows the pAUC vs AUC when tested on the development data for the various models compared to the baseline Variational Autoencoder provided. The Arcface MobileNet and the Amazon Autoencoder models really underperformed. Further work would be to look into these models more and try to see where we went wrong. Neither report about these had much technical detail about the coding but it would be interesting to see if we could replicate their performance. Overall, the only models that performed better than the baseline were training the VAE on less data by undersampling to avoid bias and the CNN Autoencoder.

One possible idea we had to improve the predictions even more was to change the way that we predict the anomalies. In order to do that, we would project our training samples on the latent space and then for every prediction we would estimate a distance for k Nearest Neighbors and if the prediction is above the threshold that we set, then it is classified as an anomaly.

Figure 4. AUC vs pAUC for development data



References

[1] Ritwik Giri, Fangzhou Cheng, Karim Helwani, Srikanth V. Tenneti, Umut Isik, and Arvindh Krishnaswamy. Group masked autoencoder based density estimator for audio anomaly detection. In *Detection and Classification of Acoustic Scenes and Events Workshop 2020*, 2020. 3

[2] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019*. ISCA, 2019. 2

[3] Qiping Zhou. Arcface Based Sound Mobilenets for DCASE 2020 Task 2. Technical report, PFU SHANGHAI Co., 2020. 2

Trial	Mel Count	Latent Dimension	Batch Size	Epochs	Architecture Size	AUC
1	256	20	128	10	Large	0.889
2	256	20	254	20	Large	0.895
3	256	25	254	25	Large	0.890
4	128	30	128	20	Small	0.875
5	128	20	128	10	Small	0.855

Table 1. AUC scores reported from the performance of the CNN on the dev dataset