2024 AML Challenge 2 Group 20

Moaad MAAROUFI Barthélémy CHARLIER

EURECOM

Abstract

In the context of the Detection and Classification of Acoustic Scenes and Events challenge (DCASE), this work aims to investigate different approaches. It begins with exploratory data analysis and preprocessing, followed by a selection of machine learning models and techniques deemed suitable for the problem.

1 Introduction

Predicting machinery failures before they occur can save significant amounts of money, time, and even human lives. Machine learning approaches are particularly useful for handling the vast amount of data for predicting potential anomalies or failures. Once trained, the inference process is generally not resource-intensive.

In our case, the problem is anomaly detection, with only normal data available for training. Therefore, we must choose models that can be trained using a single class.

We set two goals for this study:

- To test the generalizability of some models on this dataset by training them on a single class at a time and then performing zero-shot or fine-tuning on the other classes. However, this will only be done for smaller models, as it is resource-intensive.

- To compare spatial versus temporal dependencies, i.e., to evaluate the performance of a model that converts the audio's time dependency into a spatial one against a model that directly utilizes this "time" dependency.

To begin with, an interpretable baseline model such as the Isolation Forest is a suitable choice. The One-Class SVM is also a good option. After gaining a better understanding of the dataset with these baseline models, we will consider our findings and explore more complex one. These will include the Variational Autoencoder, the Convolutional Variational Autoencoder (which focuses on spatial dependency), and an Encoder-Decoder Transformer with self-supervised autoregressive (which utilizes time dependency).

2 Dataset description

The dataset for this task comprises parts of the MIMII Dataset, consisting of normal and anomalous operating sounds of six types of toy and real machines. Each recording is a single-channel audio file approximately 10 seconds in length, capturing both the operating sound of the target machine and environmental noise.

The MIMII dataset includes recordings of real machines collected with eight microphones. Anomalous sounds were collected by deliberately damaging the target machines. For this task, only the first channel of multi-channel recordings is used, and all recordings are regarded as single-channel recordings from a fixed microphone. The sampling rate of all signals has been **down-sampled to 16 kHz.**

For the AML challenge, we were assigned to work on a single machine type: the Slide rail. This machine type has three IDs (0, 2, and 4) in the development dataset, which includes both test (normal and anomaly data points) and train (only normal data).

3 Data Analysis

In our analysis, we observed an imbalance in the data distribution across different machine IDs. Specifically, the third machine (ID 04) had a lower representation in the training samples compared to the other machines but a higher representation in the testing samples. This imbalance might cause the model to be biased towards the more represented machines, making it challenging to accurately identify anomalies in the less represented machine.

The following figures illustrate the data distribution:

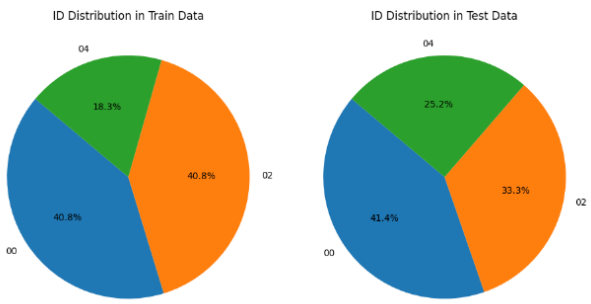

Figure 1: ID Distribution in Train Data

Additionally, we analyzed the proportion of anomalies versus normal signals for each machine ID in the test dataset. Our analysis revealed that the proportion of anomalies in all IDs is approximately the same.

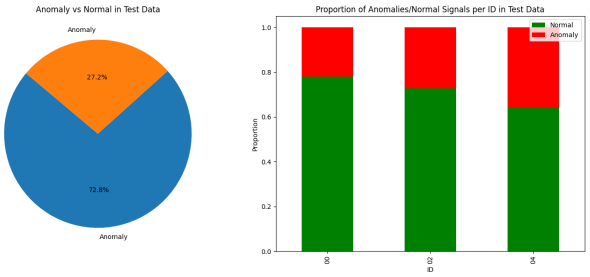

Figure 2: Proportion of Anomalies/Normal Signals per ID in Test Data

4 Data Prepossessing

4.1 Introduction to Mel Scale

When looking at the literature and the example report the professor sent, we see that most approaches utilize the mel scale. After the FFT (Fast Fourier Transform) to extract the frequencies, it spaces frequencies in a way that matches human hearing. Equal steps on the Mel scale correspond to equal perceived changes in pitch, unlike the initial FFT, where it's linear.

4.2 Human Labeling Assumptions

This approach presumes that labeling was done by human ears, which is incorrect in this case. Quote: "Anomalous sounds were collected by deliberately damaging target machines." From the official website of the DCASE Challenge. The labeling, therefore, wasn't based on humans listening to the noise and knowing the machine is deficient, but by deliberately breaking it. Consequently, the human factor doesn't have much to play here. The point we want to make is that by going to the mel scale, we might be missing out on some frequencies that are not human-audible but could convey important information in the anomaly detection process. For instance, a machine might emit high-frequency sounds when it's deficient, which humans cannot hear, and we lose them when we switch to the mel scale.

4.3 Adoption of Mel Scale Approach

Surprisingly, the models that utilize this approach work well. However, due to time constraints, we couldn't risk getting caught up by this problem, as it would take away from the main goal of this report. Therefore, we will use the well-established approach, but we will make note of this limitation.

4.4 Mel Spectrogram Hyperparameters

The mel spectrogram provides several parameters such as n_fft, which is the number of samples used by the FFT, the number of mel bins, and the hop length. We used the same parameters from this paper as they were proven to work well for most models. However, for the transformer, where we want to conserve the notion of time as much as possible, we used a smaller hop length to maintain the "temporal" dependency.

4.5 Data Augmentation Techniques

For the CNN-VAE, we augmented the images using techniques such as cutting out holes and interpolating with nearby pixels, as well as translating, scaling, and rotating the input using the Albumentations library. This approach allowed us to compose transformations and resulted in a more streamlined process, in hopes of making the model more robust and encouraging it to generalize well.

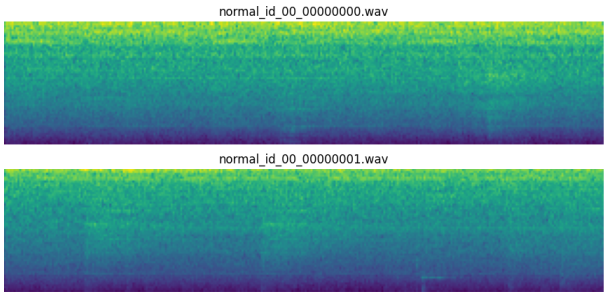

Figure 3: Mel spectrogram of normal data

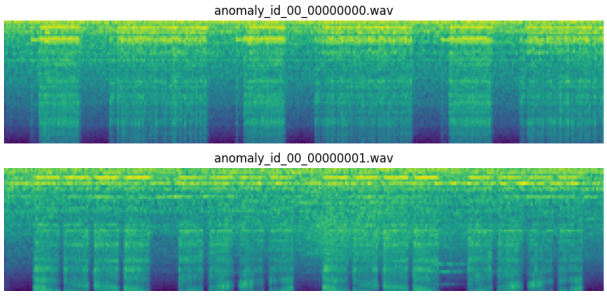

Figure 4: Mel spectrogram of anomalies

4.6 Model Specific Approaches

For all models other than the CNN-VAE and the Transformer, the mel spectrogram was flattened. The Transformer model, however, requires sequences of vectors. Since there are num_bins vectors in the spectrogram, which can too high, we concatenate some vectors to reduce the overall number of elements in each sequence while increasing the dimensionality of the individual vectors. This is why the Transformer model will use a linear layer as a projection into a smaller space.

4.7 Data Splitting Strategy

The test data was split into 60% for testing and 40% for validation. As we cannot tune the models without access to anomalies, we proceeded with this approach instead of solely relying on the reconstruction error of the normal samples, which may not reflect the model's true performance. We acknowledge the risk of data leakage, but for the purpose of our study, we are willing to accept it. We used the entire train data to train the models.

5 Model selection

6 AUC score

To measure the performance of our models, we will use the Area under the Curve (AUC). We will input the

anomaly scores from our models, and the AUC will test all thresholds and report the model's average performance. We will conduct this process for each machine ID to better understand the models' generalization.

6.1 Isolation forest

Isolation forest is a model designed to isolate observations by randomly splitting on features. It is composed of many estimators/trees which perform this random recursive splitting to isolate the data points. The number of splits until the leaf is the anomaly score. Anomalies will tend to be isolated very early on during the split. This anomaly score is averaged all the trees to give the final prediction.

This baseline will help us understand which machine IDs are most difficult to learn. We will also determine if the underrepresented class is problematic or if the models can generalize based on other classes.

We will conduct five experiments. In the first, we'll train a single model on all machine IDs. In the following three, we'll train a model for each class but test it on all classes. In the fifth, we'll downsample the larger classes to match the size of the smaller class, creating a balanced dataset.

The hyperparameters we tried to tune were n_estimators max_samples (max samples to sample from the dataset for each estimator) contamination(a prior that we give to model which hints on the percentage of contamination in the dataset) bootstrap(sample with replacement). We fine-tuned the parameters on the validation set but did not observe any significant performance improvement compared to the default parameters. We couldn't perform grid search or cross-validation due to the size of the dataset.

Data	Id0	Id2	Id4	Average
Original	0.87	0.65	0.63	0.72
MachineId 0	**0.90**	0.67	0.57	0.72
MachineId 2	0.88	**0.69**	0.56	0.72
MachineId 4	0.82	0.51	**0.64**	0.66
Downsample 2&4	0.85	0.62	0.62	0.70

The results are quite insightful. The first experiment, using the original data distribution, confirms that the underrepresented machine ID 4 has the poorest performance among the three. Additionally, ID 0 and 2 have the same number of samples, but the difference in performance indicates that anomalies in ID 0 are much easier to learn than in ID 2.

The following three experiments, where we train a model for each class and test on all classes, show an expected improvement in performance for each class. They also demonstrate the model's generalization capabilities. However, the model struggles to generalize well on the underrepresented class.

The fifth experiment with downsampling leads to slightly worse results than the training with imbalanced classes, so it's probably not a good idea to pursue this route with future models.

7 SGD One Class SVM

Another baseline model we wanted to try is a one-class SVM with an RBF kernel. This is a special case of a support vector machine that aims to find a hyperplane enclosing the majority of data points. Points falling on the other side are classified as anomalies.

The training was taking too long (over half an hour and still not done) and we couldn't afford it as we wanted to conduct multiple experiments and fine-tuning. SVM solves a quadratic programming problem, meaning the training time scales quadratically with the number of samples, or worse. By reformulating the quadratic optimization problem into an unconstrained one using hinge loss, as seen in the MALIS course, we can update the parameters using gradient descent, which scales well with large datasets.

$$\underset{w,w_0}{\arg\min} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{12-\text{ regularizer}} + C\sum_{i=1}^{N} \underbrace{\max\left(1 - y_i\left(\mathbf{w}^T\mathbf{x}_i + w_0\right), 0\right)}_{\text{hinge loss}}$$

SGDOneClassSVM by Sklearn does exactly that, so we will use it instead. However, we lose the RBF kernel with this approach, but it can be simulated using the Nystroem Method for Kernel Approximation. However, we discovered that it's not a good idea to do that as, after the transformation, we were getting a single prediction for all data points (which is probably the mean) all the time because of the curse of dimensionality. The dimensionality of the input vectors is already very large, so there is no need to push it further.

We performed the same experiments as Isolation forests. and The results were the following:

Data	Id0	Id2	Id4	Average
Original	0.94	0.75	0.52	0.74
MachineId 0	0.94	0.75	0.52	0.74
MachineId 2	0.94	0.75	0.52	0.74
MachineId 4	0.93	0.75	0.52	0.74
Downsample 2&4	70.94	0.75	0.52	0.74

The results are quite unusual. The model's scores are consistent, no matter the training, which suggests that it is excellent at generalizing. It recognizes general features but fails to capture the specific nuances of the machine IDs (probably because it's a linear model). As a result, the performance does not improve when the model is trained exclusively on a particular machine ID, unlike the IF. The fact that the model manages to generalize suggests that there might be some common patterns across the different machine IDs.

7.1 Variational Autoencoder

This is the first deep model to come in mind when it comes to anomaly dectetion, it is a more complex one compared to the baselines, so we hope that it would be able to learn the specilized fetaures of each machineId.

We use a slightly modfied version of the VAE in the official pytorch repo examples as a starting point. It consists of an encoder(2 relu activated layers) and a decoder (also 2 layers) The encoder compresses the data into a

low dimensional latent space and the decoder maps it back to the original space. The model uses the reparameterization trick to enable backpropagation through the sampling process in the latent space. The loss fucntion is composed of both the reconstruction error and the KL divergence which is kind of a regulizer keeping the latent distribution close to the prior (which is in gaussian in our case).

We performed the same experiments as before (except for downsampling):

- Trainable params: 858,280
- Estimated Total Size (MB): 3.29
- Total training time for each experiment : 15min, 50 epochs, T4 GPU.

Out of curiosity, we added another experiment where we fine-tuned a pre-trained model from the first experiment on the machineId2 data, as it had the lowest AUC score.

Data	Id0	Id2	Id4	Average
Original	0.95	0.75	0.88	0.86
Fine Tune on Id2	0.94	0.77	0.57	0.76
MachineId 0	0.94	0.66	0.54	0.71
MachineId 2	0.91	0.75	0.54	0.73
MachineId 4	0.90	0.63	0.75	0.76

Training on the original data yields significantly better results for machineId 0 and 4, while the performance for Id 2 is similar to that of SVM. In the fine-tuning experiment, the score is slightly improved for Id2, but the model forgets the training on other classes. As a general conclusion, we found that it is better to train the model on the whole dataset, as it can learn from some classes and apply that knowledge to others. Relying solely on the model's generalizability is not sufficient in this case. This approach will be used for the next models, as they are much more computationally intensive, and performing all these kinds of experiments is not feasible.

7.2 Convolutional Variational Autoencoder

As part of our study, we wanted to compare spatial vs. temporal dependencies. The mel spectrogram can be viewed as an image, as we saw in the data processing part. Upon visual investigation, we noticed that anomalies have some kind of bands (not present in normal audios), which CNNs should be able to recognize in theory. Since we had already worked with this type of autoencoder in the previous challenge, we tweaked the previous architectures to suit our new images and added more layers because the mel spectrogram images are much larger.

The results were not as expected, with the model performing even worse than the baselines. We only trained the model for 10 epochs, as it is extremely resource-intensive. It's possible that training it for longer and tweaking the hyperparameters further could yield better results. In this paper, where they also explore some models for this DECASE challenge, they used a CVAE it perfromed best on the slider machine, but in our case, it did not. This could be due to differences in architecture, data augmentation, or other factors. Due to the resource-intensive and longer training times, it's very difficult to perform cross-validation or tune the hyperparameters and retrain the model. The model has 1 million trainable parameters and took apporximately 30 minutes to train on a T4 GPU.

Data	Id0	Id2	Id4	Average
Original	0.93	0.75	0.52	0.73

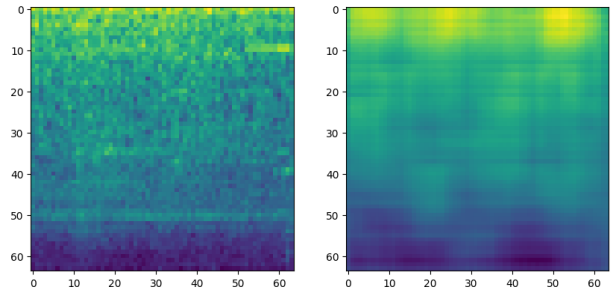

Figure 5: Reconstructed Mel spectrogram of a normal audio

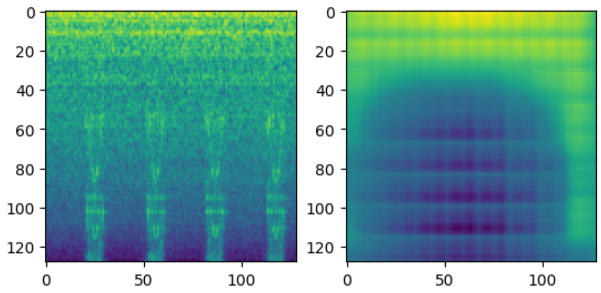

Figure 6: Reconstructed Mel spectrogram of an anomaly audio

7.3 Self supervised Transformer

7.3.1 Introduction

The approaches we've used so far, some didn't even use time dependency in the data, while others did, like the CVAE, but transformed it into a spatial dependency problem. We wondered if we could use the time dependency directly, not spatially. For this, we need models with memory that perform anomaly detection. There aren't any traditional models that do this. However, there's a method that uses the time series nature of the data for anomaly detection. The idea is to train a memory model in an autoregressive, self-supervised way. We'll use the mel spectrogram-transformed audio again, as we have the pipeline ready and, due to time constraints, couldn't redesign it for the raw signal. So, we must admit, our approach doesn't fully use time dependencies within vectors, but it does between vectors.

7.3.2 Transformer Architecture

We chose the transformer architecture because it includes an encoder by default, unlike LSTMs and GRUs. Let's say we have a sequence of vectors representing our audio, $S_{initial} = [v1, v2, v3, v4, v5]$. The sequence $S_{encoder} = [v1, v2, v3, v4]$ is given to the encoder, which produces an encoding, $E = [e1, e2, e3, e4]$. The decoder is supposed to generate $S_{decoder} = [v2, v3, v4, v5]$. It will be given the encoding E and each time $T = [v1, v2, v3, v4]$, as part of teacher forcing. We'll use masked multihead

attention so that the decoder doesn't attend to future tokens. Generally such an approach is used to genreate a diffrent sequence in ouput for example in the original transformer paper, it's used for translation. In our case, we'll use it to regenerate the input, which may seem counterintuitive. We hope that the generated output will be worse for anomalies, allowing us to classify them. Note that during inefrence there is no techer forcing and the generation is autoregressive.

7.3.3 Attention Mechanism and Positional Encoding

The attention mechanism is a set model, so we have to add positional encoding using sine and cosine functions of different frequencies. Our sequences have the following dimensions: 283 vectors with 3840 dimensions. We project these using a linear layer at the input into a 512-dimensional space, which is the model's input. We use 4 layers for both the encoder and decoder, and 8 attention heads.

7.3.4 Training Details and Results

The model was trained on an L4 gpu 24Gb for 1 hour approximately. 30 epochs with a bacth size 16.

Th results are enclosed int the following table:

Data	Id0	Id2	Id4	Average	Overall AUC
Original	0.95	0.77	0.44	0.72	0.78

7.3.5 Performance Analysis

Let's analyze the performance, referencing a recent paper, "DECODER-ONLY FOUNDATION MODEL FOR TIME-SERIES FORECASTING" that the professor recommended. In the paper, they confirm that transformers can be used for time series forecasting to create a foundational model that works with zero-shot learning. They use a decoder-style model with 200 million parameters, compared to our model's 16 million. They use a mix of real-world and synthetic data and the raw signal time series. Unlike our approach, which made the problem an inter-vector time series but not intra-vector, they have a better perspective. Additionally, the mel spectrogram transformation might be lossy in terms of time series and not very expressive especially for machine 4. Instead, we should have used the raw signal, but due to time constraints, it wasn't feasible. Using synthetic data for machine 4 could have been helpful, and training for a longer duration as well as the authors confirm in the paper.

8 Some problems with the cloud provider and model training

8.1 Issues with Google Colab

We used Google Colab for our experiments and Lightning AI only for data processing and training smaller models, as the GPU credit is consumed quickly. The bigger models were trained on Colab.

Google shuts down the machine every time we close the browser or when the computer goes to sleep. Even though we use checkpointing, it's impractical to reload the environment each time to continue training and have to leave the computer on for the entire duration of training.

8.2 Using Google Cloud to Counter Shutdowns

To counter this problem, we used free credit in Google Cloud (which can't be used to allocate GPUs, just VMs) to create a virtual machine, install a VNC viewer in Docker container, and use a minimalistic Debian distribution with a Chromium browser. This way, we could SSH, open Google Colab, and leave it running all day without it shutting down.

8.3 Data Handling Issues for Transformer Model

Another problem we faced was constructing the sequences and labels for the transformer model. Initially, we were careless and placed the input sequence and its target in the PyTorch dataset, causing it to explode in size, exceeding 60GB. This was due to the Mel spectrogram and the conversion to PyTorch tensor, as we forced the dataset object out of lazy execution(all the preprocessing was not done during training, but beforehand). This resulted in us being unable to load it entirely into the CPU memory, and we had to read from the disk every time, creating a bottleneck for training.

8.4 Dynamic Sequence Construction Solution

To resolve this, we stored only the original sequence $S_{\text{initial}} = [v1, v2, v3, v4, v5]$ in the PyTorch dataset object, and constructed the input ($S_{\text{input}} = [v1, v2, v3, v4]$) and target ($S_{\text{target}} = [v2, v3, v4, v5]$) dynamically during runtime. This reduced the dataset object to 30 GB, allowing it to fit into memory (51GB high RAM VM in Colab). We then pickled it and stored it in Google Drive, and when we wanted to train, we loaded it entirely into memory.

9 Conclusion

n this study, we experimented with various machine learning models, explaining their suitability and limitations. The VAE (Variational Autoencoder) performed the best. We observed that the SGD One-Class SVM has very good generalizability, while the VAE does not. We compared both approaches of leveraging time and spatial dependencies. However, the results were not conclusive enough to determine which is better.

10 Future Work

If we had more time and compute power, we could have investigated further to understand why the CNN (Convolutional Neural Network) model didn't work well and attempted to improve it by tweaking the architectures and tuning the hyperparameters. We could also retrain the

transformer for longer epochs, add more layers, and use synthetic data (as mentioned in the paper) with the current approach and try to implement the other approach using the raw signal to fully utilize the time dependencies. Additionally, we could experiment with zero-shot learning using the foundational model proposed in the paper with our data.

11 References

1. DECODER-ONLY FOUNDATION MODEL FOR TIME-SERIES FORECASTING.
2. Attention is all you need.
3. DCASE2020_Pilastri_86_t2
4. Example models from pytorch offical github repository.
5. EURECOM MALIS course slides: Soft SVM.
6. EURECOM Deep learning course: Sequence modeling.