Technical Report Challenge 2 - Anomalous Sound Detection

Algorithmic Machine Learning Course (AML), Spring 2023, EURECOM

I. Introduction

Anomalous Sound Detection (ASD) is the task to identify whether the sound emitted from a target machine is normal or anomalous [1]. This task tackles prompt detection of machine anomaly by observing its sounds, which results to be crucial for machine condition monitoring. The ultimate challenge lies in identifying anomalous sounds, under the hypothesis that only normal sound samples have been provided as training data. Indeed, in real-world factories, actual anomalous sounds rarely occur and are highly diverse, hence exhaustive patterns of anomalous sounds are impossible to deliberately make and/or collect. This leads to the detection of unknown anomalous sounds that were not observed in the provided training data. Given normal sound samples of some machine IDs, the objective is to build a model which can be used to predict whether an unknown sound, which belongs to one of the given machine IDs in the training dataset, is normal or anomalous. Such a scenario, in which labeled anomaly data is unavailable, requires generative models rather than discriminative or clustering methods. This is due to their capability to infer by learning the underlying probability distribution of the training data, hence allowing to capture the patterns and structures of the data. This succeeds in differentiating between normal and anomalous instances, rather than grouping similar instances together without explicitly modeling the data distribution as in clustering, or focusing on learning decision boundaries as in discriminative approaches.

II. Dataset Structure and Exploration

The data employed for this task comprises parts of Toy-ADMOS and the MIMII Dataset consisting of the normal/anomalous operating sounds of six types of toy/real machines. Each recording is a single-channel (approximately) 10-sec length audio that includes both a target machine's operating sound and environmental noise [1]. The task will concern only one machine type though, namely the Slide rail, alongside with unique Machine IDs charged to identify each individual of the same type of machine in the training dataset, these can be of three or four.

The dataset is comprised of two parts, namely a development dataset, for training and hyper-parameters tuning, and an evaluation one, designed for assessing the performance of the trained model. More specifically, the development dataset consists of 2370 samples of normal sounds for training, and 1101 of normal and anomalous sounds for the test. The second segment, i.e. the evaluation dataset, features 2370 samples for training and an additional 834 samples for testing. The samples originate from six distinct machines, and the development and evaluation datasets contain samples drawn from different machines, as shown in Figures 1 and 2.

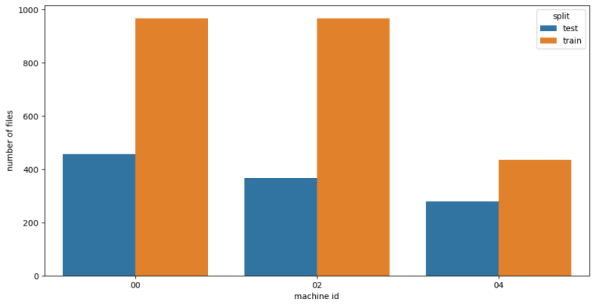

Fig. 1. Bar chart of the number of samples in the development dataset separated by machine ID

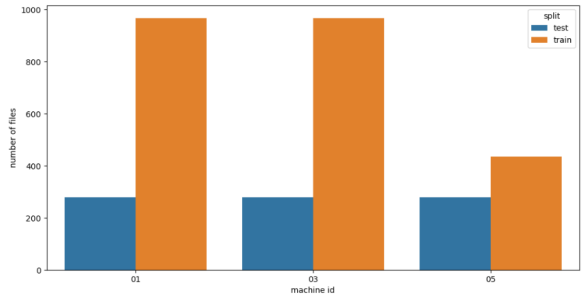

Fig. 2. Bar chart of the number of samples in the evaluation dataset separated by machine ID

In the data preparation phase, the raw audio files, which are time-series data representing sound amplitude over time, are converted into ***Mel spectrograms***, i.e. spectrograms with the frequency axis expressed using the Mel scale (Figure 3). Mel spectrograms, with their human-like non-linear perception of pitch, offer an enhanced visual representation of audio waves that allows to see how the frequency distribution of the audio signal evolves over time. Moreover, being images, Mel spectrograms can be used as input in CNN-based models.

This conversion is carried out using Librosa, a Python library [4].

Noise reduction was avoided. This choice is due to the environmental noise used for training not being representative of the anomalies of test data. Indeed, with noise reduction, the model may overfit to the noise and perform poorly in detecting true anomalies. This is due to the capability of denoising techniques of capturing the underlying patterns and structures of normal variation in the data. However, anomalies often exhibit more complex and diverse patterns compared to the normal data. The denoising process may indeed not effectively capture or reconstruct these anomalous patterns, i.e. lack of generalization, leading to sub-optimal anomaly detection performance. Ideally, the noise should capture the types of perturbations or corruptions that occur in real anomalies.

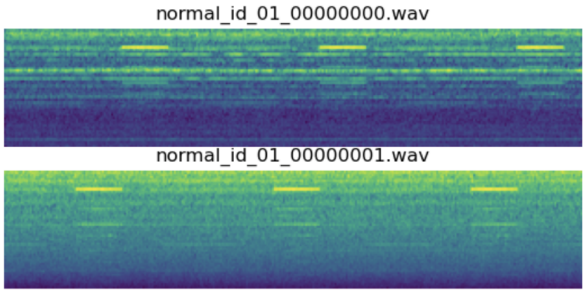

Fig. 3. Audio files (.wav) converted to Mel spectrograms

III. Models

A. Convolutional Autoencoder

As for the models, the curse of dimensionality is tackled via data compression, which is crucial in training a network. The intuition is to compress the data so that the same amount of information can be represented by fewer bits. This is where Autoencoders (AE) and Variational Autoencoders (VAE) come into play, given the high dimensionality that affects the task at hand. Indeed, a linear autoencoder (AE) with input space dimension n and latent space dimensions set to $m < n$ result will span the same vector space as spanned by the first m eigenvectors of PCA [3]. PCA is outperformed by autoencoders once non-linearity is integrated via non-linear activation functions and more hidden layers, making these capable to learn rather powerful representations of the input data in lower dimensions with much less information loss. Convolutional Autoencoder (CAE) has been favored over VAE due to its capability to learn and reconstruct the underlying structure of the input data, especially when dealing with high-dimensional data such as audio spectrograms. Additionally, unlike VAEs which incorporate randomness in their encodings, CAEs have a deterministic latent space representation, providing more stability in the anomaly scores retrieved from the reconstruction errors. The CAE architecture has been designed to receive as input a random crop of the retrieved log Mel spectogram and to be comprised of 4 layers. Each of those consists of a Convolutional Layer alongside with Dropout, to which non-linearity is introduced via a ReLu function. Due to multiple Convolutional Layers, autoencoders suffer in performance when reconstructing images as a consequence of information loss. Residual networks comprising of skip connections are a known solution to this problem. Hence, for the sake of performance improvement, such 'skip connections' can be added from the encoder to the decoder, i.e. across the bottleneck. These additional connections directly send the feature maps from the an earlier layer of the encoder to a later layer of the decoder. This helps the decoder form more clearly defined decompressions of the input image [2]. Therefore, a CAE with skip connections has been employed by stacking with the skipped connection the output of the Convolutional Layer.

B. AnoGAN

Generative Adversarial Representation Learning for Anomaly Detection As an alternative deep generative method for the anomaly detection task, we also investigated the approach based on Generative Adversarial Networks (GANs).

In a GAN, there are two neural networks, the ***discriminator*** and the ***generator***, competing against each other. The generator aims to learn how to transform random low-dimensional samples into high-dimensional data resembling the target dataset. If the generator effectively captures the distribution of the training data, it suggests that there is a latent space point in the GAN where, when passed through the generator, it can produce a sample that closely resembles a real data point. This correspondence is utilized for anomaly detection using GANs.

GAN Training A GAN consists of two adversarial modules, namely the generator G and the discriminator D. The generator G learns to create a distribution p_g over data x by transforming input noise samples z, which are 1D vectors uniformly distributed in the latent space Z, into 2D images in the image space manifold X. This manifold is populated by non-anomalous examples. The architecture of the generator G resembles a convolutional decoder with stacked strided convolutions. On the other hand, the discriminator D is a standard CNN that takes a 2D image as input and produces a single scalar value $D(.)$. This output, $D(.)$, can be interpreted as the probability that the input to the discriminator D is a real image x sampled from the training data X or a generated image $G(z)$ produced by the generator G. The optimization of both D and G occurs simultaneously in a two-player min-max game with a value function $V(G, D)$. During adversarial training, the generator improves its ability to generate realistic images, while the discriminator enhances its capability to accurately classify real and generated images.

Mapping new images to the Latent Space Once the adversarial training is concluded, the generator has learned the mapping from latent space representations z to realistic images x, namely $G(z) = z \mapsto x$. Nevertheless, the inverse mapping $\mu(x) = x \mapsto z$ is affected by the smoothness of transitions in the latent space which, under the condition of sampling from two points close in the latent space, leads to

the generation of two visually similar images. This translates into the need for the z in the latent space whose corresponding image $G(z)$ in the manifold χ resembles best the query image x. The degree of similarity of x and $G(z)$ is related to which extent the query image follows the data distribution p_g used for the generator training. In order to find the best z, a sample z_1 is randomly sampled from the latent space distribution Z and subsequently fed into the trained generator to get its corresponding generated image. A loss function yielding to an updated position in the latent space z_2 is defined according to the retrieved $G(z_1)$. The location of z in the latent space is iteratively optimized across γ backpropagation steps. The over cited loss function is comprised of two components, namely a residual loss $L_R(z_\gamma)$ and a discrimination loss $L_D(z_\gamma)$. The former consists in the L1 distance of the visual similarity between query image x and generated image $G(z_\gamma)$ in the image space, whilst the latter measures the L1 distance of the feature similarity between query image and generated image in the feature representation of the discriminator, $F(x)$ and $F(G(z_\gamma))$ respectively. The discrimination loss is charged of enforcing the generated image to lie on the manifold by using the trained discriminator as a feature extractor, rather than a classifier. The mentioned feature extractor $F(.)$ is retrieved by getting the output of an intermediate layer of the Discriminator, more specifically of the convolutional ones, neglecting the final fully connected block. Hence, the adaptation of the coordinates of z wont rely only on a hard decision of the trained Discriminator, but takes furtherly into account the information of the feature representation, learnt by the Discriminator during adversarial training.

$$L_R(z_\gamma) = \sum |x - G(z_\gamma)|$$

$$L_D(z_\gamma) = \sum_{F(.)} |F(x) - F(G(z_\gamma))|$$

Lastly, for the mapping to the latent space, the overall loss is defined as weighted sum of these both components:

$$L(z_\gamma) = (1-\lambda) \cdot L_R(z_\gamma) + \lambda \cdot L_D(z_\gamma)$$

Whilst the trained parameters of the Generator and Discriminator are kept fixed, the coefficients of z are adapted via backpropagation.

Anomaly detection The overlying loss function evaluates in every update iteration γ the compatibility of generated images $G(z_\gamma)$ with images seen during adversarial training. Therefore, defying the residual score $R(x)$ as the residual loss $L_R(z_\Gamma)$ and the discrimination score $D(x)$ as the discrimination loss $L_D(z_\Gamma)$ both computed at the last Γ^{th} update iteration of the mapping procedure, the anomaly score is expressed as the following:

$$A(x) = (1-\lambda) \cdot (x) + \lambda \cdot D(x)$$

Therefore, the model yields a large anomaly score $A(x)$ for anomalous images, whereas a small anomaly score points out that a very similar image was already seen during training.

IV. Results

A. Convolutional Autoencoder

With Convolutional Autoencoder different hyperparameters were used for training. The scope of testing encompassed numerous hyperparameters including learning rate, batch size, normalization, and applying skip connections, as shown in Table I.

learning rate	batch size	normalizing	skip connections	AUC	pAUC
0.001	256	True	False	0.904482	0.766345
0.001	512	True	False	0.904482	0.766345
0.001	64	True	False	0.904482	0.766345
0.0007	256	True	False	0.904482	0766345
0.001	256	True	True	0.831355	0.722830
0.001	256	False	False	0.741876	0.643735
0.001	32	True	False	0.904482	0.766345

TABLE I
AUC and pAUC scores obtained with different parameter combinations on the development dataset

The best AUC and pAUC scores obtained were 0.904482 and 0.766345 respectively, and were obtained by several combinations of hyper parameters as seen in the table. On validation, the best result was a score of 0.78215, obtained while using a learning rate of 0.001, 256 batch size, normalization activated, no skip connections, and 100 epochs. The number of epochs was kept stable for all tests.

B. AnoGAN

The AUC and pAUC scores obtained on the development test set are the following, with the associated hyperparameters that were used:

TRAINING	
epochs :	300
learning rate :	0.01
batch size :	500
validation split :	0.1
b1 (Adam optimizer) :	0.9
b2 (Adam optimizer) :	0.999
TEST	
test epochs :	1000
test learning rate :	0.001
latent dimension :	100
lambda (Anomaly score) :	0.1

Slider ID	AUC	$pAUC$
00	0.923146	0.685245
02	0.679326	0.507589
04	0.552135	0.535482
Average	0.718202	0.576106

V. Conclusions

In conclusion, the best result was obtained by the CAE model with no skip-connections. This might be motivated by the loss of detail. Indeed, skip-connections directly add or concatenate the output from an earlier layer to a later layer, carrying forward more detail from the input. This might result into a bad practice in the context of anomaly detection, since details carried forward by the skip-connections could not be relevant for the task w.r.t. the features of the latent space, and could introduce noise that entangles the process of identifying the important patterns. The bulk of time spent on the CAE model was spent tuning hyperparameters.

In an attempt to work on something different and potentially more interesting, we spent a considerable amount of time working on the AnoGAN model. The results obtained with AnoGAN were worse than what was obtained with the Convolutional AE method, despite what was reported in the original paper (See [5]). However, hyperparameters tuning was not performed for our AnoGAN implementation because of lack of time: most of it was used to produce a working implementation.

It is worth noting that AnoGAN is not the only implementation of Anomaly Detection with GANs: the most recent ones (e.g. f-anoGAN, see [6]) solve what is the biggest drawback of AnoGAN, that is its execution speed during inference. Indeed, the backpropagation process performed to optimize z is time-consuming and was replaced in later works by other direct mapping techniques.

Future work may focus on testing several combinations of hyperparameters for AnoGAN, both for the training and testing phases, to achieve better results.

References

[1] https://dcase.community/challenge2020/task-unsupervised-detection-of-anomalous-sounds

[2] https://towardsdatascience.com/using-skip-connections-to-enhance-denoising-autoencoder-algorithms-849e049c0ac9

[3] https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

[4] https://librosa.org/doc/latest/index.html

[5] https://arxiv.org/pdf/1703.05921.pdf

[6] https://www.sciencedirect.com/science/article/abs/pii/S1361841518302640