AML Spring 2022

Challenge 2: Anomalous sound detection

September 19, 2022

Group 8: Harkati Chiraz Rayene, Senane Zineb, Zheng Estelle

Emails: {Chiraz.Harkati, Zineb.Senane, Estelle.Zheng}@eurecom.fr

Professor: Michiardi Pietro

1 Introduction

This challenge aims to detect whether sounds emitted from machines are anomalous or normal by training only on normal samples. In summary, we will do the following steps: Process the sound clips to convert them into data that a neural network (NN) can use (i.e. into images), develop several models with different learning strategies in order to evaluate the ability to detect abnormal sounds in a test set and compare the approaches in terms of results, difficulty of implementation, computation complexity, etc.

2 Data Analysis & Preprocessing

2.1 Data Exploration

We are provided two datasets: a development dataset and an evaluation dataset. Each sample is an approximately 10-sec-long audio emitted from a certain machine id of a slider machine type with its environmental noise. As shown in Fig.1, the development set is composed of 2370 training samples labeled as normal and 1101 normal and anomalous testing samples. The evaluation dataset is also decomposed into a training set (2370 normal samples) and testing samples (834 samples), but with different machine ids and with unlabeled testing samples. There is neither duplicated data nor features missing values. Each dataset contains 3 different machine ids that are the same for train and test, but the ones of the evaluation set are different from the development set. The purpose behind this separation is that we train and optimize our models on the development set and evaluate the performances of our model on the evaluation set. As the development dataset contains only sounds classified as normal, we will use unsupervised learning to build a model that is capable of detecting anomalous sounds.

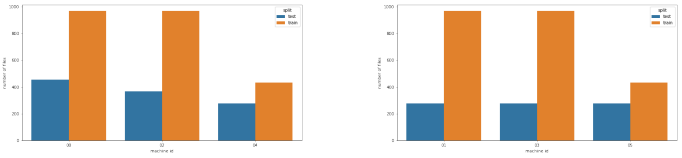

Fig. 1: Distribution of extracts of the slider machine type (left: development, right: evaluation)

2.2 Data Preprocessing

Instead of using the audio in its raw form, we used a common approach to convert the audio to spectrogram images such that a NN architecture can process these images. The waveform spectrum gives an idea of how loud or soft a sound clip is over time, but nothing about frequency. Since the spectrum in the frequency domain is a surrogate representation that allows us to identify the frequencies present at a given point in time and their magnitudes, we used a spectrogram that shows the signal energy by plotting the frequency spectrum over time, putting them together and adjusting color palette. The principle behind these spectrogram generations is to apply a Fourier transform to a WAV file to decompose the signal into its fundamental frequencies.
Mel scaling is an elegant way to capture the fundamental features of audio data as an image, much like the "fingerprint" of a signal. It is more interesting than a normal spectrogram as it expresses the mel scale in decibels rather than frequency and amplitude. This is suitable for humans logarithmic loudness perception, as humans perceive a very small and concentrated range of frequencies and amplitudes. In our setup, we used the librosa feature.melspectrogram function to generate Mel spectrograms with the following parameters: n_mels=128, frames= 5, n_fft=1024, hop_length= 512, and Power = 2.0.
Because our dataset is small, we considered data augmentation. When processing audio data, data augmentation can be performed on the raw audio data or the spectrogram image, or even both. However, we did not use this technique because converting audios to spectrograms generates large images (if we keep good resolution) to pass as input to the autoencoder, which will take a longer time to train.

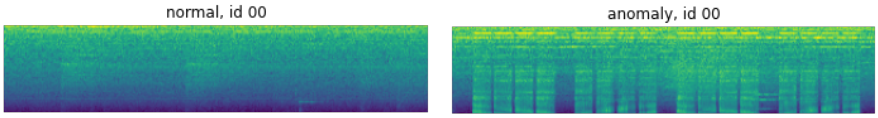

Fig.2: Mel spectrograms of a normal sound (left) and an anomalous sound (right)

2.3 Features Engineering

To extract useful features, we first investigated two main approaches to sound processing: Mel-Frequency Cepstral Coefficients (MFCC) and Mel-Frequency Energy Coefficients (MFEC). MFCC, derived from the Mel cepstrum representation of audio, is one of the most famous and popular audio processing in such classification problems but it has a drawback, it applies a discrete cosine transform (DCT) to the logarithm of the filter output, resulting in a decorrelated MFCC feature. Hence, we will have non-local features that are not really useful and unsuitable for convolutional neural network (CNN) processing. Therefore, we tried another audio signal processing function called MFEC, which is logarithmic energy directly derived from filter bank energy. These are similar to MFCC, but do not include DCT operations. The feature achieves great results in detecting various audio sounds and classifying them.

3 Model Selection

Methods like unsupervised nearest neighbors (KNN) or isolation forests are famous models for anomaly detection. However, KNN is a naive approach and has high time complexity, and isolation forests do not work well for high dimensional data and have high variance. Thus, we will rather use deep learning techniques such as NNs as they are suitable for high dimensional data (large images). In this challenge, we used dense autoencoder (DAE), variational autoencoders (VAE) and convolutional autoencoders (CAE).
The base learner is an autocoder (AE) which is an unsupervised artificial neural network which aims to regenerate the input by learning latent representations. It consists of an encoder that transforms input into a latent representation (hidden representation with a lower dimension), via a nonlinear transform, and a decoder that reconstructs the input. Based on the reconstruction error that we used as an anomaly score, we can detect anomaly sounds. First, we trained the model only with normal samples to minimize the Mean Squared Error (MSE), which is the reconstruction error chosen. Then, we tested with unseen data. The reconstruction of an image from a normal sample (close to what the autoencoder trained on) will result in a smaller reconstruction error, but the reconstruction of an anomalous sample will result in a larger one, hence detect anomalies.
For each of the three AE used, the input and output shape is 640. During training, we used Mini-Batch Gradient Descent in the Adam optimizer for all architectures with a learning rate of 0.001 and the batch size for the DAE, CAE and VAE was set as 512, 64 and 1000 respectively.

3.1 Dense Autoencoder (AE)

The DAE used is a deep fully-connected AE where the encoder and decoder are composed of four fully-connected (FC) layers with 128 hidden units. After each layer, we added a batch normalization layer to deal with the vanishing gradient problem, and we used ReLU as the activation function. The latent space layer is a FC layer with 8 hidden units.

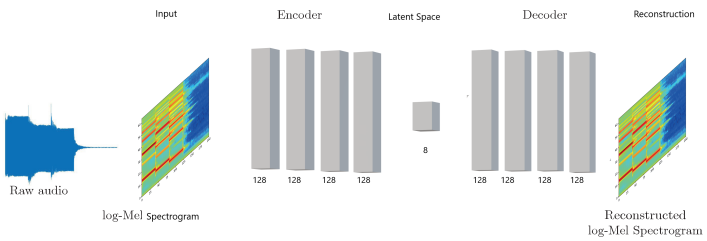

Fig. 3: AutoEncoder Architecture

3.2 Variational Autoencoder (VAE)

VAE is a DAE combined with Variational Inference and is thus a probability-based generative model. The architecture of a VAE is very similar to a regular DAE. The main difference is that the latent space of a VAE has a layer of data means and standard deviations used, thanks to variational inference, to generate a sample with the same shape as the latent representation of the corresponding DAE. VAE assumes that the input is drawn from a probability distribution and it attempts to find its parameters.
The VAE we used is provided by the baseline. The encoder and decoder are composed of two FC layers with 400 hidden units followed by ReLU activation function. The final latent space layer is a FC layer with 20 hidden units.

3.3 Convolutional Autoencoder (CAE)

As FC layers fail in capturing the patterns of images because they cannot hold neighboring data, we tried to use convolutional layers for a good capturing of the mel spectrograms in latent space. That is the motivation behind using CAE. However, during encoding, the image sizes get shrunk by subsampling with pooling which leads to information loss and makes the reconstruction hard while decoding.
CAE is a AE where the underlying network is a CNN. The CAE architecture is presented in Fig. 3. The encoder is composed of 5 hidden layers and convolutional filters of 32, 64, 128, 256 and 512. We set the kernel sizes and strides respectively as the following: 5, (1,2); 5, (1,2); 5, (2,2); 5, (2,2) and 2, (2,2). The latent space is a convolutional layer with 40 filters. The decoder was made of one FC layer to expand the latent representation to the shape of the last layer of the encoder, followed by five transposed convolutional layers that mimic the encoder structure. Each layer is followed by batch normalization and ReLU activation function.

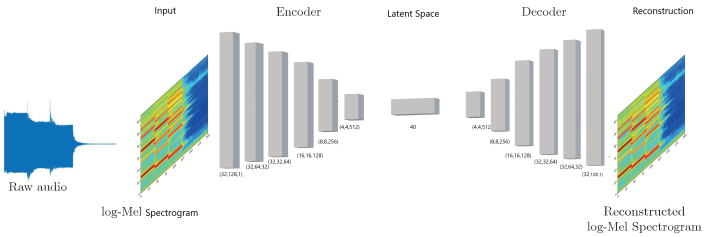

Fig. 4: Convolutional Autoencoder architecture

4 Model Performance

To compare the models, we used the Area Under the Curve (AUC) metric, which is an evaluation metric that summarizes in a single number the ability of a binary classifier to distinguish between classes. It is the area under the ROC curve which is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The higher the AUC, the better the model is at distinguishing classes. We also used the partial AUC (pAUC) metric which is calculated from a portion of the ROC curve.

Machine Id (Development)	VAE (epochs 20)		VAE (epochs 50)		VAE (epochs 100)		DAE		CAE	
	AUC	*pAUC*	*AUC*	pAUC	AUC	pAUC	*AUC*	pAUC	*AUC*	*pAUC*
00	96.3	82.3	95.8	79.7	95.4	77.6	96.3	82.1	98.0	94.6
02	73.3	61.6	76.2	63.1	77.1	63.7	78.3	61.9	83.3	66.9
04	79.5	58.9	86.1	62.9	86.9	61.5	90.6	66.6	97.0	84.6
Average	**83.0**	**67.6**	**86.0**	**68.6**	**86.5**	**67.6**	**88.4**	**70.2**	**93.0**	**82.0**
Evaluation AUC	**71.1**		**75.5**		**75.7**		**84.7**		**90.1**	

Table 1: Comparison between the different models' performance

For VAE, we implemented different models varying the number of epochs (20, 50 and 100). With only 20 epochs, the model worked quite well, achieving an average AUC of 83.0% on the test set of the development dataset, and 71.1% on the test set of the evaluation dataset. However, these performances are still noticeably worse than when we train the VAE with more epochs. With both 50 or 100 epochs, we got a test AUC of $\tilde{8}6\%$ on the development dataset and $\tilde{7}5$-76% on the evaluation dataset. This shows that to a certain point, increasing the number of epochs won't improve the model as we get to the limit of what the model can perform. Indeed, we get more than 10% points less on the AUC of evaluation compared to the one on development which is a sign that the model does not generalize greatly. Adding more hidden layers to encoder and decoder (only 2 for each in our model) might have improved a bit the model. However, this gap might also be due to the fact that the assumed distribution over prior is not suitable in this problem. Morover, VAE are usually used for generating new data that are related to the original images. As it is a generative model, we don't expect to replicate the same input image identically. But for anomaly detection, we want to replicate the input image as close as possible to minimize the reconstruction error, and thus, we usually use DAE or CAE. This is also a reason that can explain the slightly worse performances we got compared to other models.
For DAE, we trained 50 epochs which we considered sufficient as the model architecture was similar as VAE and training with more than 50 epochs did not considerably improve performances. During the training, the validation loss was decreasing overall, as shown in Fig. 5, which is a good sign to reduce overfitting. There are two spikes in the loss function at epoch 15 and 31 but it is not critical because since we are using Mini-Batch Gradient Descent in Adam, spikes are an unavoidable consequence of this as some mini-batches have by chance unlucky data for the optimization, inducing those spikes in the cost function using Adam. We obtained an AUC on the development and evaluation test set of 88.4% and 84.7% respectively, which is a consequent improvement with respect to VAE.
CAE achieved the best result with respect to VAE and DAE. We obtained an AUC on the development and evaluation test set of 93.0% and 90.1% respectively. The validation loss was initially low at epoch 1 (0.39) and decreased rapidly for a few epochs and remained almost constant afterwards, as shown in Fig. 5. This decreasing path for the validation loss is a good sign to reduce overfitting. Especially, values taken by the loss are much lower for CAE than for DAE. This result is expected as the main advantage of CNN compared to DNN is that it automatically detects the meaningful features without any human supervision and often outperforms traditional DNN. We set a maximum number of epochs 100 and set an early stopping criteria to monitor if the validation loss did not decrease for more than 10 epochs. With this early stopping, the model trained 21 epochs, which is a lot less than what we needed for VAE and DAE. In terms of computation and time complexity, CAE has a higher cost per epoch but as we need less epochs to obtain a good model, it is a good compromise.

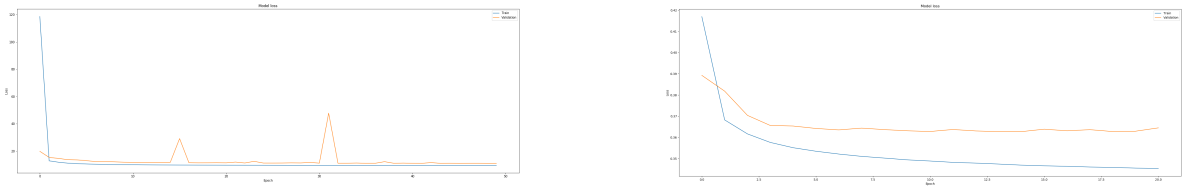

Fig. 5: Training loss and validation loss for AE (left) and CAE (right)

5 Conclusion & Further Improvements

In this challenge we worked on raw audios, processed them and converted them to spectrograms such that they can be directly passed to AE. The choice of AE was suitable as we are working with images in an unsupervised learning context. Given the two datasets, the models (DAE, VAE and CAE) were trained on the energy features extracted from the mel spectrograms of the datasets' training audios. Overall, the best performance has been achieved by the CAE with an AUC on evaluation test set of 90.1%.
To improve the performance of the DAE, we aim to use an adapted data augmentation technique such as time-shifting on a small part of our datasets. For instance, this will ameliorate the training AUC values and also spend a moderate time while training AE. As a future work, other types of deep learning architectures can be explored and used for ASD such as GAN.