EURECOM
Sophia Antipolis

Challenge 2 Report

Anomalous Sound Detection

Federico Germinario, Andrea Ghiglione, Gabriele Gioetto, Nour Jamoussi

24-05-2022

Abstract

In this report we present an approach to an unsupervised anomaly sound detection (ASD) problem. Despite having at our disposal only normal sounds for training, we created a setting of **classification-based ASD** where the main task for the model is to solve the machine ID identification. The approach is based on data analysis, converting audios to spectrograms, implementing a **Siamese Neural Network** as a machine ID classifier and then using the network as a features extractor for **KNN anomaly detection**, which addresses the original problem. Following this method we evaluate the anomaly score of samples of the testing data. The proposed approach provided satisfactory results overcoming the baseline with an AUC score of 0.94.

1 Problem Overview

1.1 Introduction

The challenge is about the definition of a model to detect an anomalous sound from machines. The original training data consists of sounds of 4 machines and 2 toys, but for this challenge we worked on *slider* machine only. In particular, the problem is unsupervised since every sound in the training set is normal. This is also a realistic scenario because there are many different sounds that can be anomalous. So in general, these problems are solved by letting the model strongly understand the concept of *normal* sound, in order to detect as *anomalous* everything which is not normal. Common approaches consist of Autoencoders with reconstruction error, however we adopted a different strategy, splitting the problem into two stages; one supervised stage with a siamese network that worked both as a machine ID classifier and as a feature extractor (its encoder), and a second stage of exploiting extracted features feeding them to a KNN outlier detector from PyOD library. Our aim is to obtain a model which performs well, evaluated in terms of Area Under the Curve (AUC).

1.2 Data Exploration

The training set is divided into a development set and an additional training set (also called *train evaluation*). In each of them there are sounds of the same machine types, but from 3 different machines each. So, the total training set is composed by 6 machines of type *slider*. The development set contains also a test part, which is used as a local reference in terms of AUC score, and it consists of only 3 slider machines. On the other hand, the evaluation test set consists of the other 3 slider machines. The audios are divided as follow:

- *Train development*: 2370 audios of slider machines with IDs 0, 2 and 4
- *Train evaluation*: 2370 audios of slider machines with IDs 1, 3 and 5
- *Test development*: 1101 audios of slider machines with IDs 0, 2 and 4
- *Test evaluation*: 834 audios of slider machines with IDs 1, 3 and 5

All the audios have a sample rate of 16kHz and a length of 10 seconds as shown in Figure 1. In addition, the training machine IDs are unbalanced; as we can see in Figure 2 slider machines 4 and 5 are the half of the other slider machines.

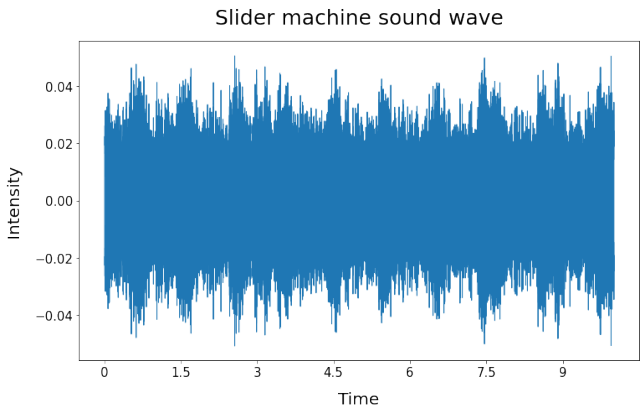

Figure 1: Typical slider machine sound wave.

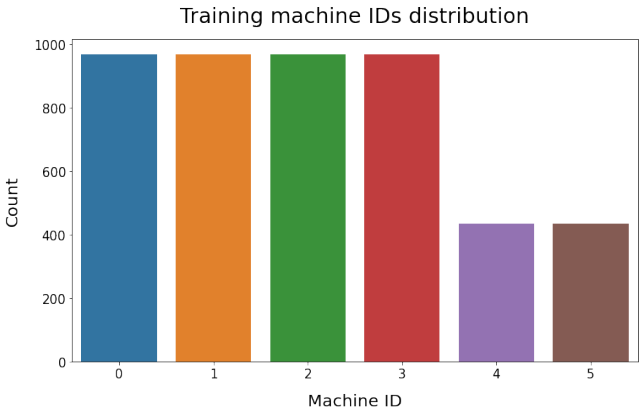

Figure 2: Training machine IDs distribution

2 Proposed Approach

2.1 Data processing: Audio Transformation

Since we worked with Convolutional Neural Network we had to provide input images rather than audios. For this reason we converted sound waves into spectrograms, in order to capture information about the amplitude of a particular frequency at a particular time of an audio signal in an image. Due to the limited computational resources we converted each sound wave into a (384,384) mel spectrogram and then we resized it as (128,128) which is our network input shape. We normalized the data in order to have zero mean and unit variance in all the input spectrograms. Even if the problem was unsupervised, we used machine IDs as labels for the siamese network, and we splitted the training data to get a validation set (20% of the training data) stratifying on the labels. In this way we ensured to have a good reference with a well-balanced validation set, avoiding overfitting.

2.2 Model Implementation

2.2.1 Siamese Network Classifier

The first block of our model is the Siamese Network Classifier, which performs an additional supervised task. In particular, siamese networks take two inputs which are processed in parallel by two twin networks and through an ad-hoc loss function the network is able to predict if the two inputs belong to the same class or not, and quantify such information with a probability. The parameters between the two networks are shared to ensure that two similar images get mapped in the same location in the feature space. So, instead of having as goal just to classify the inputs, a siamese network learns also how to differentiate between inputs by learning their similarities. We used the contrastive loss as a loss function which minimizes the embedding inputs distance when they are from the same class (same machine ID here) and maximizes the distance if they are from different classes. In particular the loss function we used for a single sample is reported in Formula 1 where the distance D is euclidean distance between the true label and the predicted one and M is a margin value of 1. The siamese network structure is reported in Figure 3. This network allowed us to get an encoder trained on a supervised task; we were able to get a feature extractor for an unsupervised problem by solving a secondary problem of supervised classification. The feature extractor is the encoder of the siamese network (after removing the final dense layer) and it is represented in Figure 4. After each convolutional layer we applied tanh activation function and the encoder extracts 64 features per input. Pairs to feed to the siamese network were picked randomly, following only a simple criteria: for every image we had to pick the same amount of positive and negative images, i.e. we had to create the same numbers of pairs with that specific image that are of the same class (positive) and different classes (negative). Also, since machine IDs 4 and 5 have half of the records of other IDs we generated the double of pairs with respect to other machine IDs; in this way we had a balanced input in terms of pairs and classes.

$$(1 - y_{true}) \cdot D^2 + y_{true} \cdot (max(0, M - D))^2 \tag{1}$$

As optimizer, we used Stochastic Gradient Descent with Momentum. After trying different parameters, we selected a learning rate equal to 0.05 and a momentum of 0.7.

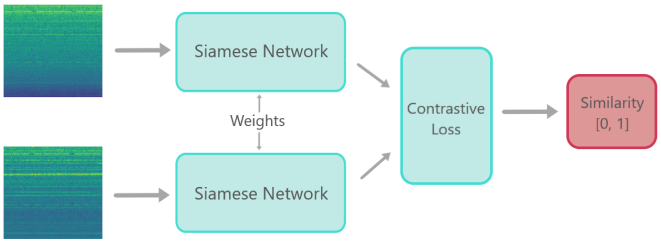

Figure 3: Siamese Network structure

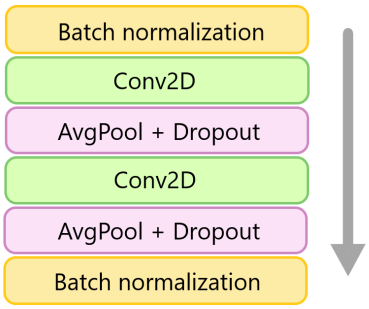

Figure 4: Encoder structure,which extracts 64 features

2.2.2 KNN Anomaly Detector

Once we produced 64 features per input sound, we can pass them to the last block of our model, which is an outlier detector. In particular, we used a K-Nearest-Neighbors detector. This model was extremely useful since it is strongly able to detect outliers thanks to the features extracted at the previous block. To be specific, KNN for outlier detection identifies the anomalies using the distance from each point to its neighbors. In aim to differentiate between a normal sound spectrogram and an anomalous one (an example is shown in Figure 5 and Figure 6), the KNN Neighbors Detector provided by PyOD can use three different methods to calculate the anomaly score :

- *largest*: use the distance to the Kth neighbor as the outlier score
- *mean*: use the average of all K neighbors as the outlier score
- *median*: use the median of the distance to K neighbors as the outlier score

After choosing the method, we had to set the *contamination*, which is used to calculate the threshold on the decision function. The last hyperparameter used is the *n_neighbors*, which determines the number of neighbors used for the anomaly score.

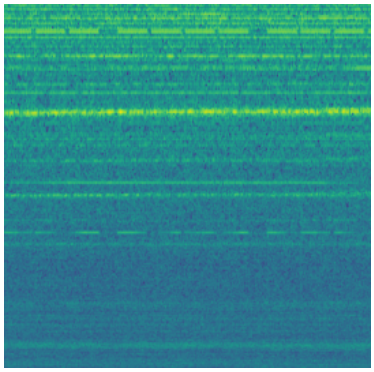

Figure 5: Spectogram of a normal sound

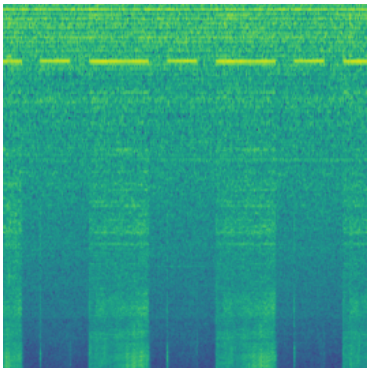

Figure 6: Spectogram of an anomalous sound

3 Results

3.1 Numerical results

After having tried different setting, we found out that a shallow network works better. This may be due to the fact that very high level features are not strictly necessary to distinguish between normal and anomalous sounds. Also, by using a shallower network, the training was much faster. In particular, two convolutional layers with 4 filters each of size (5,5) were enough to achieve an accuracy of around 90% on the validation set in the siamese network. As KNN outlier detector parameters we used *largest* method, *radius 1.0*, 5 *n_neighbors* and *leaf size* 30, after having applied PCA. We also tried Gaussian Mixture Models (GMM) and Local Outlier Factor (LOF). Local AUC score on sound anomalies was roughly 0.93. We also tried different input pair combinations and different spectrogram shapes; summary of our trials is reported in Table 1.

Spectrogram size	(32, 32)	(64, 64)	(128, 128)	(192, 192)	(384, 384)	(1025, 313)
Conv2D layers	1		2		3	
Pairs generated	3 - 6		4 - 8		5 - 10	
Outlier detector	KNN		GMM		LOF	

Table 1: Overview of the main parameters and combination we tried. Pairs generated X - Y correspond to number of positive pairs generate per class; if the class is machine ID 4 or 5 we generate Y pairs, otherwise we generate X pairs. Same reasoning is applied for negative pairs.

3.2 Limits and merits of the model

Of course one of the biggest limits of this model, but in general of deep learning, is explainability. We tried different parameters combinations and different network architectures, but it is very difficult to understand precisely the process that led to final results when working with deep learning.
On the other hand, this model has the big merit to be composed of two blocks, which one of them is supervised. Being in an unsupervised scenario and having this model that solves an additional task allowed us to reach very satisfactory results in the leaderboard and at the same time we were able to experience with a different approach a problem of unsupervised anomaly sound detection. In addition, our model is very light and very flexible. We also tried to implement other methods without using deep learning, i.e. exploiting spectrogram features only. However, this model based on CNNs overcame it since it is able to capture much more information which is then conveyed to the outlier detector that will complete our task.
It would be interesting to extend this work having all the machines types at our disposal; we could use the Siamese Network to solve first the problem of machine types classification, and then fine tuning it on each different specific machine (i.e. the one we solved for slider machine). This should provide more robust features and it would be another very challenging scenario, since we would have to deal with different machine types and different normal sounds.