Anomaly Detection - Challenge 2 AML

Alex Argese
Student
EURECOM
Biot, France
alex.argese@eurecom.fr

Cristian Degni
Student
EURECOM
Biot, France
cristian.degni@eurecom.fr

Miriam Lamari
Student
EURECOM
Biot, France
miriam.lamari@eurecom.fr

Enrico Sbuttoni
Student
EURECOM
Biot, France
enrico.sbuttoni@eurecom.fr

***Abstract*—This report presents a comparative analysis of unsupervised machine learning techniques for detecting anomalous sounds in industrial equipment, specifically focusing on the slider machine type. The primary challenge addressed is identifying unknown anomalous sounds in test samples, given only normal sound recordings during training. We describe the dataset, which includes audio recordings from the ToyADMOS and MIMII collections, and detail the preprocessing steps such as audio normalization, feature extraction using log-mel spectrograms. Various models were evaluated, including standard Autoencoders, Variational Autoencoders (VAEs), Convolutional Autoencoders (CAEs), and the PANNs (Pretrained Audio Neural Networks) inference framework. The models were assessed using Area Under the Curve (AUC) as the primary metric to reflect their ability to rank samples by anomaly likelihood. Results highlight the strengths and limitations of reconstruction-based methods versus embedding-based representations, with PANNs showing promising generalization across machine IDs. This work underscores the potential of deep unsupervised methods for robust anomaly detection in real-world, noise-contaminated industrial environments.**

***Index Terms*—Machine Learning, Anomaly Detection, Sound processing, Autoencoder, Pannes Inference**

I. Introduction

Detection of anomalies in industrial equipment is a critical task for ensuring operational efficiency and preventing costly downtimes. In this challenge, we focus on detecting anomalous sounds from a specific type of machine, the slider, using unsupervised machine learning techniques. The primary goal is to identify unknown anomalous sounds in test samples, given only normal sound recordings during training. The dataset used for this challenge includes audio recordings from MIMII dataset.

II. Dataset

The dataset consists of audio recordings in **.wav** format, collected with eight microphones placed around a slider machine. all recordings are regarded as a single-channel recording as a fixed microphone. Each recording is a single-channel (approximately) 10-sec length audio that includes both a slider machine's operating sound and environmental noise. The sampling rate of all signals has been downsampled to 16 kHz.

The training set contains normal sound recordings, while the test set includes both normal and anomalous sounds.

The dataset is structured as follows:

- **dev/ folder**: contains around 1,000 samples of normal sounds for training and 100-200 samples each of normal and anomalous sounds for testing.
- **eval/ folder**: contains around 400 unlabeled test samples for evaluation. Inside this forlder, there are also other normal samples that can be used for training.

The distribution of classes within the test set is unbalanced:

- **Class Anomalous**: 801 sounds
- **Class Normal**: 300 sounds

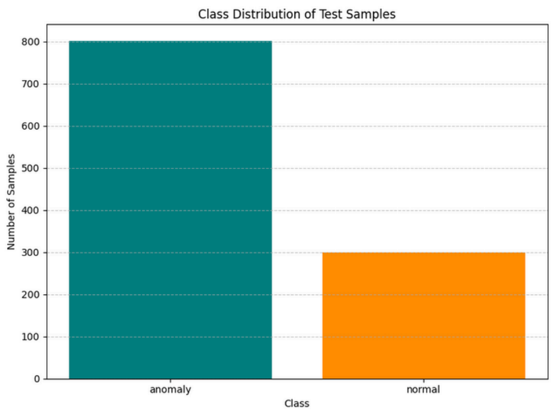

Fig. 1: Test samples class distribution.

III. Preprocessing

A. Mel-spectrogram transformation

Coversion of waveform into a mel-spectrogram representation is a common practise used with time-frequency feature in audio analysis, particularly suitable for machine listening tasks like anomaly detection.

The preprocessing involved first loading the **.wav** sounds using **Librosa** Python library and then extract mel-spectrograms using the following parameters:

- number of FFT components, number of samples used in each Fourier Transform window;
- hop length, determines how far apart successive analysis frames are;
- number of mel bands, sets how many mel frequency bins to use in the final spectrogram.

The mel-spectrograms were converted to decibel scale and reshaped into flattened vectors for efficient storage and model input compatibility. Extracted features were organised into separate folders for training and testing samples.

A mel-spectrogram looks like this:

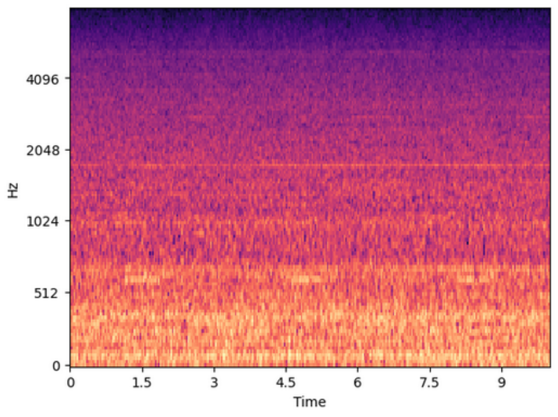

Fig. 2: Example of a mel-spectrogram.

B. Feature Normalization

After converting the audio signals to mel-spectrogram representations and flattening them into 1D feature vectors, we applied feature normalization to ensure numerical stability and improve model performance.

We used z-score standardization, which transforms each feature to have a mean of 0 and a standard deviation of 1. This is important for models that are sensitive to feature magnitudes, such as support vector machines and neural networks. Without normalization, features with larger numerical ranges could dominate the learning process or slow convergence.

The normalization parameters (mean and standard deviation) were computed from the entire set of normal training samples: this aligns with the anomaly detection paradigm, where the assumption is that only normal examples are available during training.

IV. Models Evaluated

A. Logistic Regression

Logistic Regression was used as a **baseline model** due to its **simplicity** and **interpretability**. The model was implemented using the scikit-learn library. Since this algorithm does not exploit **spatial structure** in images, each **32×32 RGB image** was **flattened** into a **3072-dimensional vector**.

To improve **performance** and **stability**, we conducted a small **grid search** on key **hyperparameters**, tuning the **regularization strength** (C), the **solver**, and the **maximum number of iterations**. The tuning process was performed via **5-fold cross-validation** on the training set using **F1 score** as the evaluation metric.

TABLE I: Hyperparameter tuning for Logistic Regression

Hyperparameter	Ranges	Best Parameter
C	0.1, 1, 10	10
solver	lbfgs	lbfgs
penalty	l2	l2
max_iter	1500, 3000	1500

While the model is limited by its **inability to learn spatial features**, it remains a **valuable baseline** that performs surprisingly well in this context, particularly when combined with proper **preprocessing** and **class balancing**.

B. Support Vector Machine

Support Vector Machines (SVM) are well-suited for **binary classification** tasks with **high-dimensional** input spaces. In our setup, the images were first **flattened** into **3072-dimensional vectors** and then **standardized** using **z-score normalization**.

We performed a **grid search** on key **hyperparameters** using **5-fold cross-validation** to optimize **model performance**. The tested hyperparameters included different **kernel types** (rbf, sigmoid, poly), values for **regularization parameter** C, and **kernel coefficient** gamma.

TABLE II: Hyperparameter tuning for SVM

Hyperparameter	Ranges	Best Parameter
C	0.1, 1, 10	10
kernels	‘rbf’, ‘sigmoid’, ‘poly’	rbf
gammas	‘scale’, ‘auto’	scale

The **optimal configuration** consisted of an **RBF kernel** with C=10 and gamma='scale'. Although **computationally more expensive** than logistic regression, SVM provided improved **generalization capabilities** by modeling **non-linear boundaries** in the data space.

C. Convolutional Neural Network (CNN)

To better capture **spatial structures** in the images, we implemented a **custom Convolutional Neural Network (CNN)** using **PyTorch**. The architecture was designed with **simplicity** in mind to ensure **interpretability** and **fast training**, yet flexible enough to benefit from **data augmentation**.

The model consists of two **convolutional layers** followed by **max-pooling**, **batch normalization**, **ReLU activation**, and **dropout**. The final layers are **fully connected** with a **sigmoid output unit** for **binary classification**.

We performed a **grid search** on four key **hyperparameters** to **fine-tune** the model.

TABLE III: Hyperparameter tuning for general CNN

Hyperparameter	Ranges	Best Parameter
learning rate	0.001, 0.005, 0.01	0.001
drop_out	0.25, 0.4, 0.5	0.25
initial_filters	8, 16, 32	32
fc layers	64, 100, 128	64

The model was trained for **10 epochs** using the **Adam optimizer** and **binary cross-entropy loss**. **Data augmentation** included **random horizontal/vertical flips** and **slight rotations**, applied **on the fly** during training. Thanks to this setup, the CNN was able to learn **meaningful spatial patterns**, achieving **substantial improvements** over **non-convolutional models**.

D. ResNet18

To leverage **deep feature representations** learned from **large-scale image datasets**, we adopted a **transfer learning** approach using **PyTorch's pretrained ResNet18 model**. The model, originally trained on **ImageNet**, was adapted to our **binary classification** task by replacing the final **fully connected layer** with a **custom head**: a **single neuron** followed by a **sigmoid activation function**.

We **froze** the pretrained layers and **fine-tuned** only the **final block** and **classification head**. Images were resized to **224×224** to match the **input size** expected by ResNet18.

A small **grid search** was conducted to tune the **learning rate** and **optimizer**.

TABLE IV: Hyperparameter tuning for general ResNet18

Hyperparameter	Ranges	Best Parameter
learning rate	0.001, 0.0001	0.001
optimizer	'adam', 'sgd'	'adam'

The model was trained using **binary cross-entropy loss** and the **Adam optimizer**. We applied **moderate data augmentation** (**horizontal/vertical flips** and **rotations**) and **early stopping** to avoid **overfitting**. The results demonstrated **superior performance** compared to all other models evaluated.

V. Metric Justification

Given the **class imbalance**, we used the **F1 score** as the main metric, as it balances **precision** and **recall** better than accuracy. This is especially important in **ecological monitoring**, where missing a cactus (**false negative**) is more critical than a false detection. F1 was therefore used for both **model selection** and **hyperparameter tuning**.

VI. Results Summary

A. Detailed Results – Logistic Regression

The best **logistic regression** model, trained with the selected **hyperparameters**, was evaluated on the **held-out test set**. The table below summarizes the **classification performance per class**:

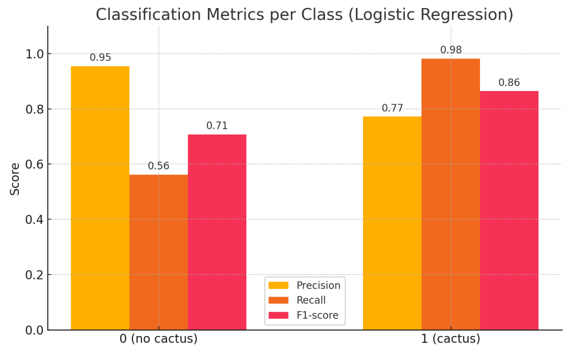

Fig. 3: Metrics for Logistic Regression

The model achieved a final **test accuracy** of **81.01%**. It showed very high **precision** for class 0 (**no cactus**), but the relatively low **recall** (**0.55**) indicates many **false negatives**. For class 1 (**cactus**), the model achieved **strong performance** across all metrics, reflecting its **bias toward the majority class**. This suggests that while the **logistic regression model** benefits from the **augmentation strategy**, it still struggles with **class imbalance** and lacks the capacity to model **complex spatial patterns**.

B. Detailed Results – Support Vector Machine

The best-performing **SVM** model was evaluated on the **test set**. The performance breakdown per class is shown below:

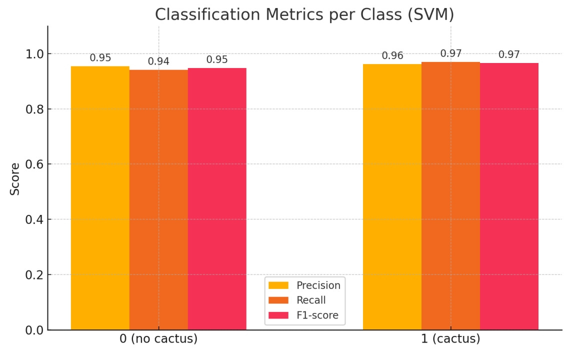

Fig. 4: Metrics for SVM

The overall **test accuracy** was **95.8%**, with a **weighted F1 score** of **0.9582**. The SVM showed higher **recall** for the **cactus class** (class 1), and moderately improved **recall** on the **minority class** (class 0) compared to **logistic regression**. This confirms the model's ability to capture **non-linear decision boundaries** through the **RBF kernel**, although it still struggled with some **overlap in feature space**.

C. Detailed Results – Convolutional Neural Network

The best **CNN** model was evaluated on the **held-out test set**. The performance results per class are as follows:

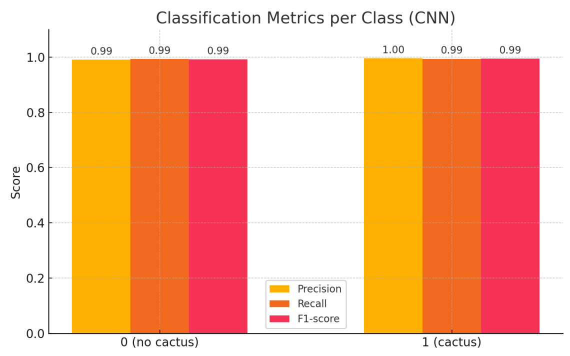

Fig. 5: Metrics for SVM

The final **test accuracy** was **99.0%**, with a **weighted average F1 score** of **0.9896**. The **CNN** outperformed both **logistic regression** and **SVM** by a wide margin, particularly in identifying **minority class** (class 0) samples, thanks to its ability to learn **local spatial features** and **generalize well** from **augmented examples**.

D. Detailed Results – ResNet18

The **fine-tuned ResNet18** model achieved the **best performance** on the **test set**. Below is the detailed **classification report**:

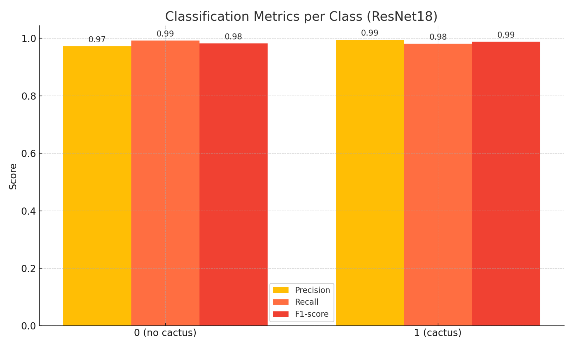

Fig. 6: Metrics for SVM

ResNet18 reached a final **test accuracy** of 98.6%, with a **weighted F1 score** of 0.9857. It demonstrated **excellent generalization** and **balance between precision and recall** across both classes, validating the power of **transfer learning** even in **low-resolution**, **small-format image classification** tasks.

TABLE V: Performance Comparison of Models on the Validation Set

Model	F1 Score	Accuracy
Logistic Regression	0.8098	81.01%
SVM	0.9582	95.82%
CNN	0.9896	98.96%
ResNet18	0.9857	98.57%

VII. Model selected

Although the **custom CNN** achieved a slightly higher **validation F1-score** (**0.9896**) than **ResNet18** (**0.9857**), we selected **ResNet18** as the **final model** due to its superior **robustness** and **generalization capabilities**. Its **pretrained layers** from **ImageNet** allow it to leverage **learned features** even on **small**, **low-resolution inputs** like our **32×32 aerial images**. This makes it more **reliable for deployment** on **unseen data**, where the **custom CNN** might be more prone to **overfitting**.

VIII. Inference on Unlabeled Test Set

After **model selection**, we applied the four **best-performing models** — **Logistic Regression**, **SVM**, **CNN**, and **ResNet18** — to the **4000 unlabeled images** from the **test set**. Each model was used to generate a **prediction (0 or 1)** for every image, maintaining the **order** of the images as read by the **DataLoader**.

To combine these predictions, we employed a **weighted majority voting** strategy, assigning a **normalized weight** to each model based on its **accuracy** on the **internal test set**. The final class for each image was assigned as **1 (cactus)** if the **weighted sum** of predictions exceeded **0.5**, and **0** otherwise.

We opted for **weighted majority voting** instead of **simple majority voting** to give greater influence to **more accurate models**. This choice reflects our aim to **prioritize** the decisions of models that demonstrated **higher reliability** during **validation** and **internal test**, thus improving the **ensemble's overall robustness** and **predictive power**.

Finally, the resulting **predictions** were saved in a **CSV file** (ensemble_predictions.csv) with two columns: **id** (image filename) and **label** (predicted class).

IX. Conclusion and Next Steps

Despite its **simplicity**, **Logistic Regression** achieved a **strong baseline performance**, demonstrating that even **linear models** can be effective when supported by appropriate **preprocessing** and **balancing techniques**.

The **SVM** outperformed logistic regression, especially in terms of **class 1 recall**, but was still constrained by the lack of **spatial awareness** in **flattened input representations**.

The **custom CNN** significantly improved **classification accuracy** and **balance across classes**, confirming the advantage of **convolutional architectures** in **image-based ecological tasks**.

Among all tested models, **ResNet18** stood out with **outstanding precision**, **recall**, and **F1 scores**, confirming the effectiveness of **transfer learning** even when applied to **small aerial images** of **ecological relevance**.

X. References

This report is inspired by the VIGIA project as described in:

Efren López-Jiménez et al., **Columnar Cactus Recognition in Aerial Images using a Deep Learning Approach**, Ecological Informatics, 2019.