Introduction to Machine Learning - the Voice Classifier project

Names: Kuba Drażan, Filip Filipkowski, Inaki Gutierrez-Mantilla Lopez, Hector Rodon Llaberia, Filip Sabbo-Gołebiowski

Date of submission: January 2026

Abstract

In this report, we present our project about training a neural network to solve the binary classification problem for voice recordings. The recordings are sorted into two classes, depending on the voice of the speaker. Speakers are divided into two distinct sets: accepted and rejected.
This report contains a summary of the whole process, which includes data collection, data exploration, preprocessing, training, experimentation and evaluation.

Contents

List of Figures

1 Introduction

1.1 Problem setting and task

We consider the following scenario: we are a security team in a building with zones of restricted access, with doors being the only way in. Each door is equipped with a device which is able to record sounds. We are to construct a software which allows authorised personnel to open the door using their voice, but restrict the access of unauthorised people.

In other words, we construct and train a neural network to sort voices into two classes: *Allowed* and *Not allowed*. This is done without the use of a spoken password; we only use the general sound of the voice of the person. The task is therefore a binary classification problem on short voice recordings, where the spoken content is irrelevant and the identity/voice characteristics are decisive.

The process of creating the software includes collecting audio data, preprocessing the data, designing and training a neural network and evaluating the results. The purpose of this report is to showcase our method of solving these tasks as well as the results we have got.

1.2 Audio representation

As we handle audio data, we require a way to represent it in a form suitable for a neural network. To this end, we use mel-spectrograms. A mel-spectrogram is a time–frequency representation that shows how energy is distributed across frequencies over time. It can be created by applying a Fourier transform to obtain a spectrogram, and then mapping frequencies to the mel scale, which is closer to human perception. This representation is widely used in speech-related machine learning tasks and works well with convolutional architectures.

1.3 Evaluation criteria and goals

For evaluation, a natural metric is the accuracy of the model on the test set, as this is a common criterion to judge a neural networks effectiveness. Additionally, we examine the amount of false positives. This stems from the fact that false positives describe the case where an unauthorised person is granted access, which is a tremendous safety violation. By contrast, false negatives are not as detrimental to safety, since these describe the case of an authorised person being denied access. Of course, we try to avoid both types of mistakes.

In terms of the goals we set for this project, we aimed to achieve an accuracy level of 95%. We decided to prioritize the reduction of the amount of false positives if the accuracy is (almost) not negatively impacted by these changes.

2 Data obtaining and preprocessing

Most of our data is from the DR-VCTK dataset. It is in a device-recorded form, which makes it suitable for our scenario: the recordings contain realistic channel effects and background noise typical for consumer microphones. In addition, we include a small set of our own manual voice recordings. This serves two purposes: to check how the trained model behaves on data recorded outside the original corpus, and to later verify the end-to-end real-time prototype that uses a microphone as input.

We download the dataset as a zip archive, extract it locally, and then construct our working dataset by selecting a fixed set of speakers and assigning them to the two target classes: *Allowed* (label 1) and *Not allowed* (label 0). We cap the amount of audio taken per speaker to obtain around 15 minutes of recordings per each accepted one and around 5 minutes per rejected. Final dataset has 6 accepted speakers and 20 rejected ones.

We then apply a preprocessing pipeline to each recording. We perform simple silence removal using an energy-based rule: samples below a low-

energy threshold (computed per recording) are removed to reduce long quiet fragments. Finally, the remaining signal is split into fixed-length segments of equal duration.

After processing both the corpus subset and the manual recordings in the same way, we generate a manifest file listing all segments together with their speaker identity and class label. This manifest is used later for data splitting and training.

3 Exploratory Data Analysis

In this section, we present the analysis performed on the collected audio dataset to understand its structure, quality, and class separability. The analysis was conducted using the `exploratory_data_analysis` and `error_analysis` notebooks.

3.1 Dataset Overview and Class Balance

The dataset consists of a total of 11,225 sample audio recordings. We verified the class balance between the two target classes: *Not allowed* (labeled as 0) and *Allowed* (labeled as 1). The dataset is nearly balanced, containing 6005 samples for the *Not allowed* class and 5220 samples for the *Allowed* class. This balance suggests that the model should not be biased toward a majority class during training.

3.2 Speaker Distribution

We analyzed the distribution of recordings across speakers to identify potential biases. The dataset contains a total of 26 unique speakers. The distribution of files per speaker is not uniform; specifically, we observed two distinct groups 20 speakers from the "not allowed" category, each contributing approximately 75 files, and a group of 6 speakers (the "allowed" category), each approximately 225 files.

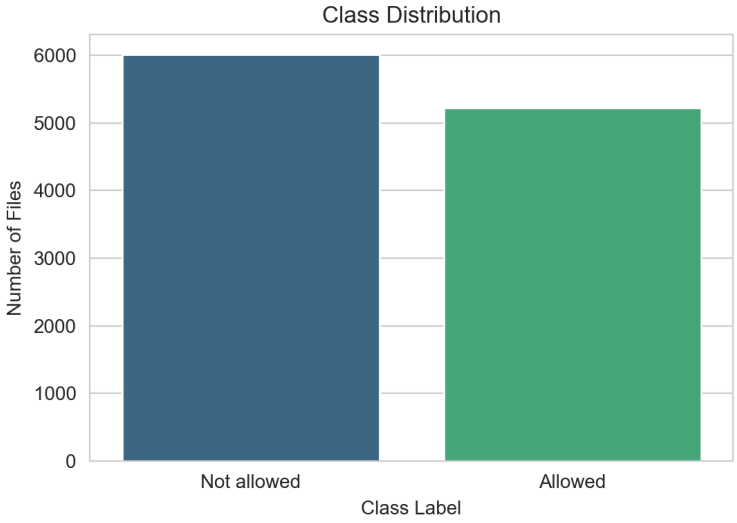

Figure 1: Class distribution of the dataset, showing a balanced ratio between "Not allowed" and "Allowed" recordings.

This distribution aligns with the project's data collection strategy regarding allowed and not allowed personnel.

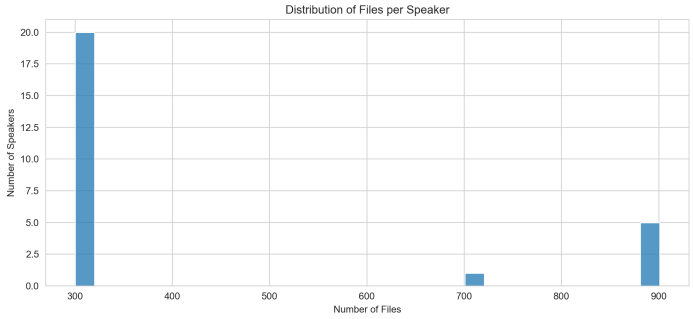

Figure 2: Distribution of audio files per speaker. Note the higher volume of recordings for the 6 "allowed" speakers compared to the 20 "not allowed" speakers.

3.3 Audio Signal Properties

To ensure data consistency, we examined the duration and energy of the recordings.

- **Duration:** An analysis of the recording duration distributions confirmed that all samples in the dataset are exactly 2 seconds long. This uniformity eliminates the need for duration-based padding or truncation during preprocessing.
- **Total Recording Time:** The total aggregated recording time is almost equal for both classes, further reinforcing the dataset's balance.
- **Anomaly Detection:** We analyzed the energy distribution of the samples to detect anomalies (e.g., files that are too quiet). The energy features were found to resemble a normal distribution, indicating consistent recording conditions without significant outliers.

3.4 Feature Analysis and Visualization

We performed elementary statistical analysis and visual inspection to assess the difficulty of the classification task.

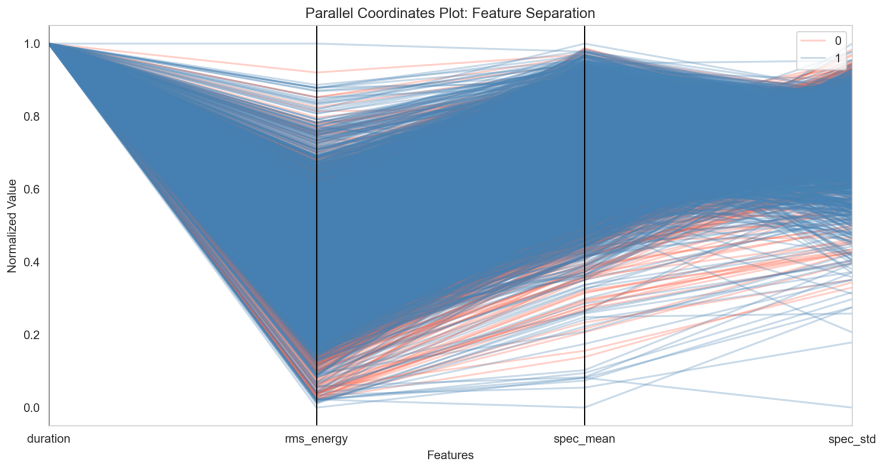

Figure 3: Multidimensional analysis. Parallel Coordinates.

As we can see on plot 3 the classification task is difficult. There are no significant differences in audio features.

3.4.1 Spectrogram Inspection

Visual inspection of the spectrograms showed high data quality, with clear frequency structures visible for both classes.

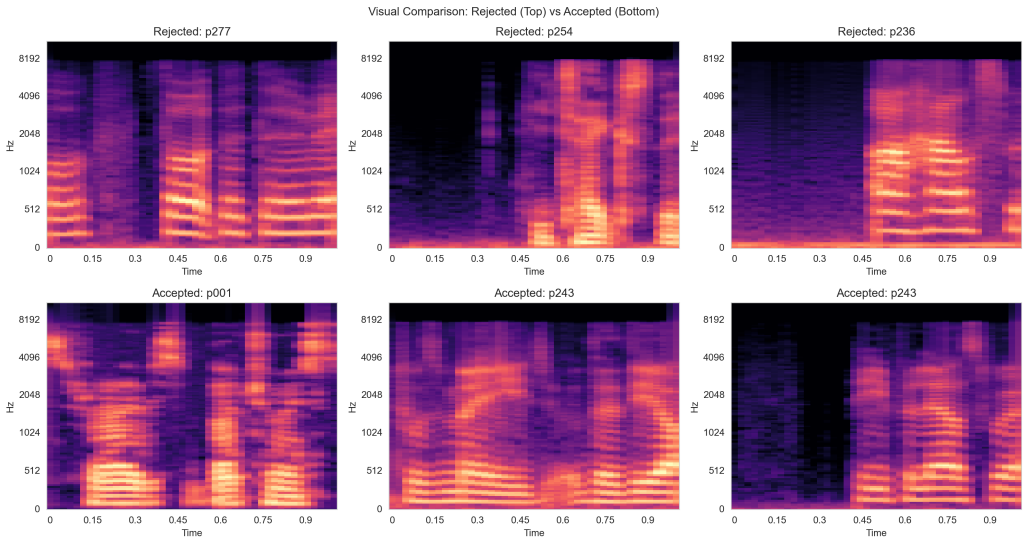

Figure 4: Visual comparison of spectrograms. Top row: Rejected samples; Bottom row: Accepted samples.

3.4.2 Multimensional Analysis (t-SNE)

Box plots and elementary statistics revealed that both classes share very similar characteristics in univariate analysis, suggesting the problem is non-trivial. To understand the separability of the data in a higher-dimensional space, we employed t-SNE projections on the model embeddings.

The analysis indicated that while there are no simple linear boundaries, there is a fairly good separation when points are colored by their true labels, see figure 5.

Notably, the manual recordings of speaker `p001` form a distinct cluster, significantly separated from others, as seen in the figure 6.

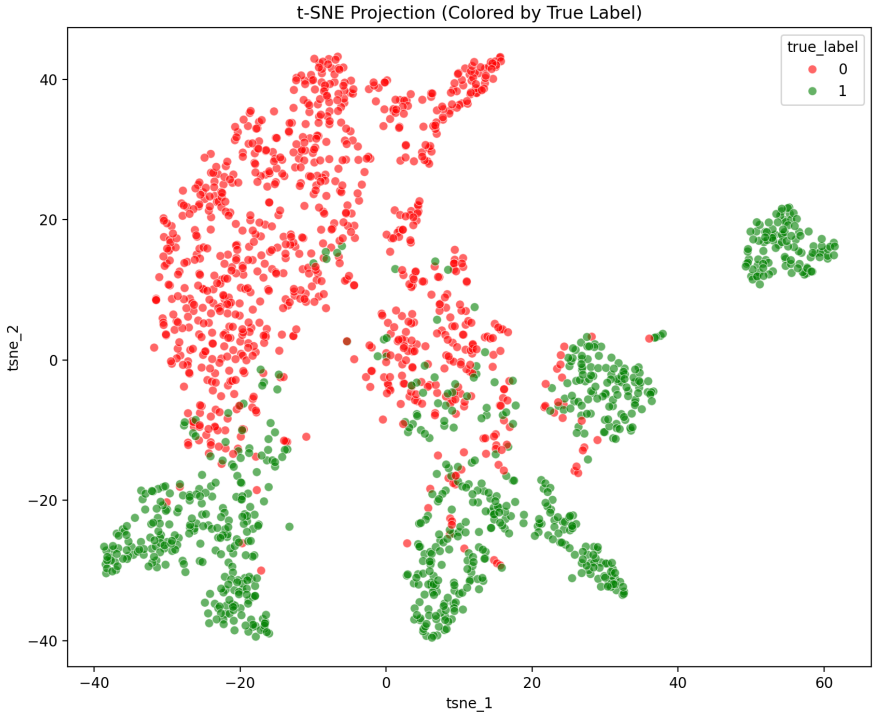

Figure 5: t-SNE projection of the dataset embeddings. The separation of colors indicates that the model can distinguish between the classes in the latent space.

3.5 Conclusion of EDA

The Exploratory Data Analysis confirms that the dataset is well-structured and clean. The classes are balanced, and the audio duration is uniform. However, the lack of significant differences in elementary statistics implies a hard classification problem. The t-SNE results suggest that while the data is separable, it likely requires non-linear feature extraction, validating the choice of a neural network architecture for this task.

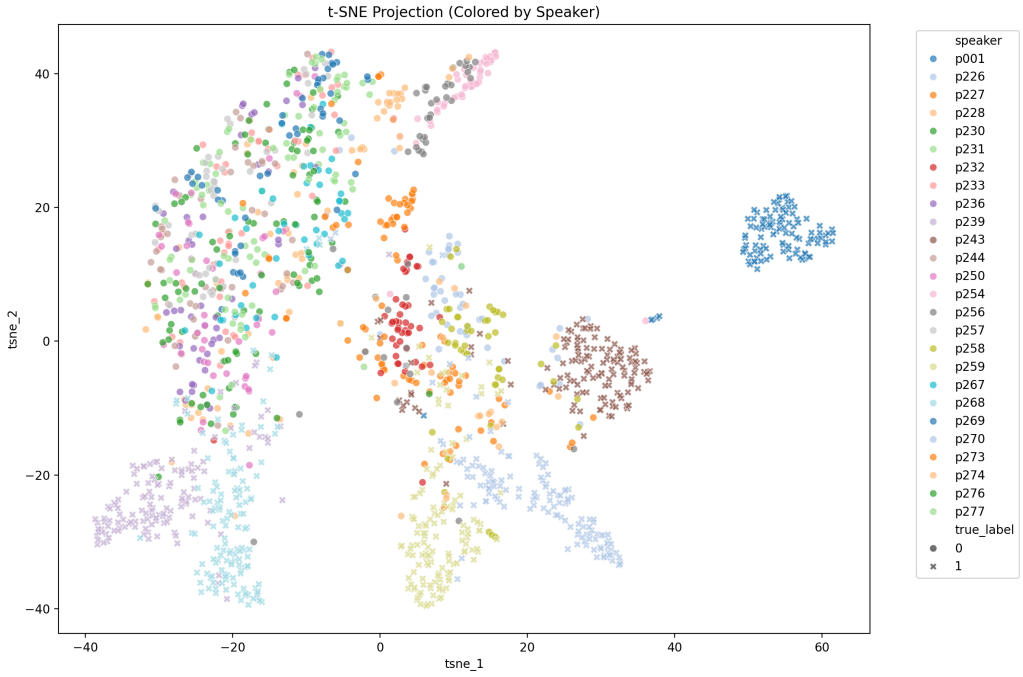

Figure 6: t-SNE projection of the dataset embeddings. The separation of colors indicates that the model can distinguish between the classes in the latent space.

4 Splitting and training

4.1 Three-way split and leakage prevention

The dataset is split into training, validation and test sets using an approximate ratio of 70%/15%/15%. Since our samples are obtained by segmenting longer recordings, a purely random split would lead to information leakage (very similar segments from the same original recording could end up in both training and test). To prevent this, all segments originating from the same original recording session are assigned to the same split. This makes the reported validation and test performance more realistic.

4.2 Baseline model and training setup

As a baseline, we train a compact convolutional neural network on mel-spectrogram representations of audio. Intuitively, the model works like this: early layers learn local time–frequency patterns in the spectrogram (e.g., formant-like structures), pooling layers progressively compress the representation, and final dense layers combine the extracted patterns to output a single probability of the *Allowed* class.

The baseline is trained as a binary classifier with binary cross-entropy loss. We use mini-batch training (batch size 32) for up to 15 epochs, monitoring validation loss and applying early stopping (patience 5) to reduce overfitting and avoid unnecessary training.

4.3 Experimentation and results

Starting from the same baseline CNN and leakage-safe split, we tested optimizer choice (SGD vs Adam), learning-rate scheduling, weight decay (AdamW), dropout, and some set of architectural variants, including changing depth/width and adding batch normalization. Below is a summary of these experiments and some results.

Our baseline used Adam with a constant learning rate of 10^{-3} and no dropout. This configuration reached **94.48%** test accuracy. Replacing Adam

with SGD (with momentum) and using a higher learning rate of 10^{-2} improved the result to **95.49%**, suggesting slightly better generalization for this task. Next, we added moderate dropout (0.2) as a simple regularizer. With Adam and the original learning rate (10^{-3}), dropout increased test accuracy further to **95.96%**, indicating that the baseline network benefits from mild regularization and otherwise tends to overfit slightly.

We then introduced learning-rate reduction when validation loss stops improving. This did not dramatically change the results, but it made training more stable and consistently competitive. Finally, we tested AdamW, i.e., Adam with decoupled weight decay, as an additional form of regularization. With dropout and a cosine schedule the result was **94.66%** (not an improvement), but combining AdamW (weight decay 10^{-4}) with dropout and plateau-based learning-rate reduction produced the best test accuracy of **96.44%**. Overall, the most reliable gains came from regularization (dropout + weight decay) and conservative learning-rate control near convergence.

In experiments that used dropout, we additionally applied Monte Carlo dropout at inference time to estimate uncertainty by running multiple stochastic forward passes. The mean predictive uncertainty (standard deviation across passes) was low for correct predictions (about **0.065**), but substantially higher for wrong predictions (about **0.236**). This indicates that the model is typically less confident exactly on the samples it misclassifies.

In a separate final training/evaluation run of the same overall approach, we obtained a comparable test accuracy of **96.20%**.

Variant (relative to baseline)	**Test accuracy**
Baseline (Adam, 10^{-3}, no dropout)	94.48%
SGD (momentum, 10^{-2}, no dropout)	95.49%
Adam + dropout 0.2	95.96%
Adam with too small LR (10^{-4})	92.52%
Adam + dropout 0.2 + plateau schedule	95.73%
SGD + plateau schedule	95.85%
AdamW + dropout 0.2 + cosine schedule	94.66%
AdamW + dropout 0.2 + plateau schedule (best)	96.44%

Table 1: Summary of the main training variants tested.