Classification Experiments Report 2025

Team TOOTHPASTE

Aral Cimcim (k11720457) **Sandro Müller (k52010874)** **Linus Madlener (k12310088)**

Kevin Eberl (k12322451)

Contributions

Aral: Labeling function, Evaluation, Experiments, RFC with MLC
Sandro: Formatted & organized the latex report
Linus: Early classification experiments, created slides
Kevin: Experimented with classifiers & analyzed plots

1 Labeling Function

We have selected the following files with multiple sound events to assess if the labeling functions could capture the intended classes:

677254.mp3: This file was annotated three times with each annotation describing it as "A musical instrument similar to a piano is playing." The length of the annotated onset and offset varies for each annotation between 0.02 - 0.65, 9.43 - 26.51, and 2.18 - 4.15 seconds. The recording consists of two people playing saxophone and piano simultaneously. The labeling functions did capture the intended classes in the recording. The free text annotations do mention "piano" but the word "saxophone" is missing. However, it is worth mentioning that the original caption for this file uses the description "piano" (for a detailed graph see Figure 1). The labeled events were clearly audible in the recording.

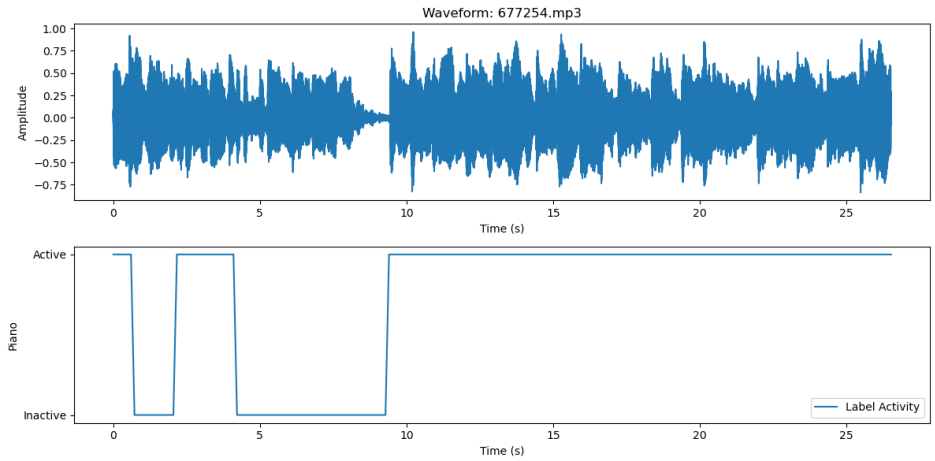

Figure 1: Analysis of the mapped class using the labeling function.

231710.mp3: This file was annotated five times with the descriptions "rhythmic drumming", "a man is talking nearby". The beginning of the file includes a drum part for 5 seconds and then a two men begin to have a conversation. The original captions match the free-text annotations. The labeling function has correctly identified the partitions of the file as "Speech" and "Drums" (for a detailed graph Figure 2). All sound events in the file are clearly audible.

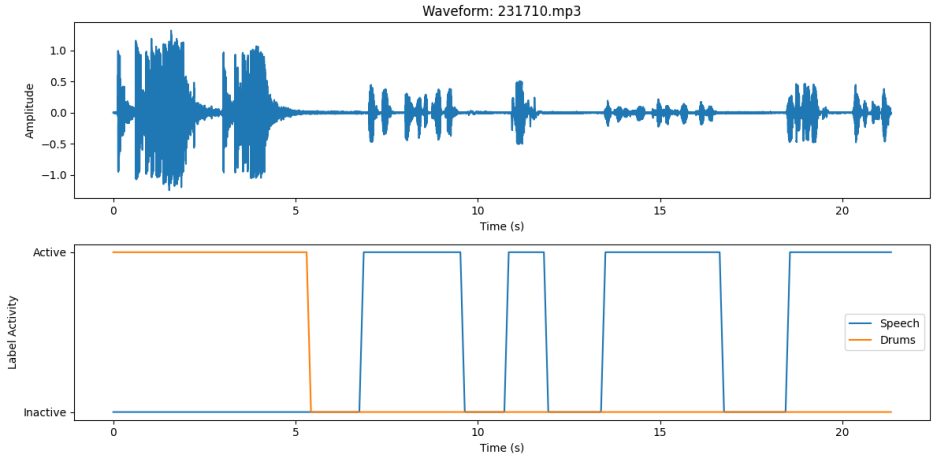

Figure 2: Using the waveform to assess if the labels were correctly identified.

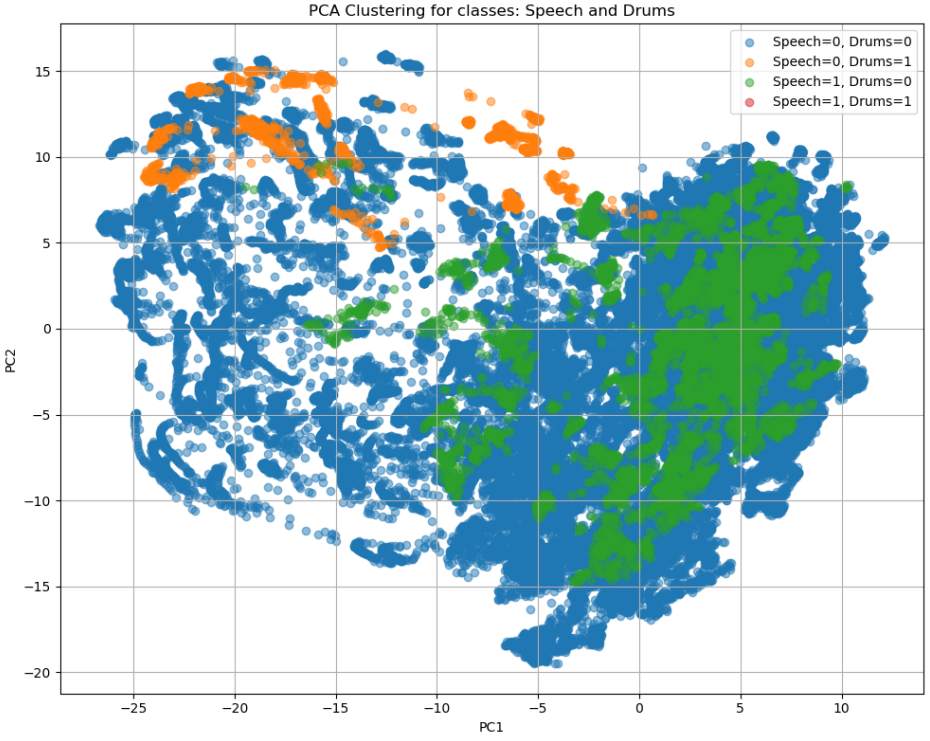

Figure 3: PCA clustering for the classes of interest displaying a small amount of separability.

2 Data Split

We have decided to use 80 % of the data for training (so that the model has enough examples to generalize) with 10% for validation (to reduce overfitting) and 10% for testing (to estimate the model's unbiased performance).

The training set was used for cross-validation and the validation set was used for fine-tuning the hyperparameters. The remaining 10% was used to evaluate the model's performance.

Yes, there are potential risks for information leakage, namely, if the features in the training set are highly correlated with the labels of the test set, this could cause feature leakage. Avoiding to use features from the test set could prevent this type of leakage.

There could also be issues with multiple recordings of the same subject such as "wind" where the model could rely heavily on these subject-specific aspects of the data. One way to address this would be to split the data in such a way that the samples containing information about the same subject are either only in the test set or the training set.

Semantic similarity of class labels could be another issue where the annotation embeddings could have high similarity for certain classes and text embeddings.

3 Audio Features

During the initial phase of the project, we performed exploratory data analysis to identify some important features. Building on those insights, we applied Principal Component Analysis (PCA) to our 768-dimensional feature embeddings, reducing them to just 100 principal components that together explain approximately 90% of the total variance. This reduces the computational costs significantly, while maintaining the bulk of the information needed to train our models.

4 Evaluation

To compare hyperparameter settings and evaluate our models, we use the balanced accuracy score. Since we are dealing with imbalanced data (not all classes are equally distributed), it prevents a model from getting a good score simply by favoring the majority class.

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP}+\text{FN}} + \frac{\text{TN}}{\text{TN}+\text{FP}}\right) \tag{1}$$

We implemented a simple baseline classifier that always predicts the majority class by default (see Table 1). For example, when asking if 'drums' occur in '231710.mp3' (see Figure 2), the baseline predicts 0 because the label is inactive most of the time. Using plain accuracy this already achieves about 75%, but this simply captures the imbalance between "no-drums" and "drums". In contrast, balanced accuracy gives equal weight to the true-positive rate and the true-negative rate. This prevents a classifier from appearing strong merely by favoring the dominant class, getting a better evaluation when one class is less frequent.

Metric	**Drums**	**Speech**
Accuracy	0.747	0.534
Weighted Accuracy	0.500	0.500
Precision	0.000	0.000
Recall	0.000	0.000
F1 Score	0.000	0.000
ROC AUC	0.500	0.500
PR AUC	0.253	0.466

Table 1: Performance Metrics for labels "Drums" and "Speech"

5 Experiments

To reduce computation times, we only used a subset of samples during hyperparameter tuning. This allows us to vary the values of parameters and compare the results to find a good combination, but keep the training time relatively low. For Classifiers with multiple parameters we perform a grid search using all possible combinations.

5.1 K-Nearest Neighbors Classifier

Because each sound event is either present or absent, we treat each label as its own binary classification task. We apply a K-nearest Neighbors (KNN) classifier independently to each label and run repeated evaluations to identify the best hyperparameters. In this setup, the number of k neighbors is crucial: choosing k too small makes the model too complex and prone to overfitting, while a very large k leads to underfitting.

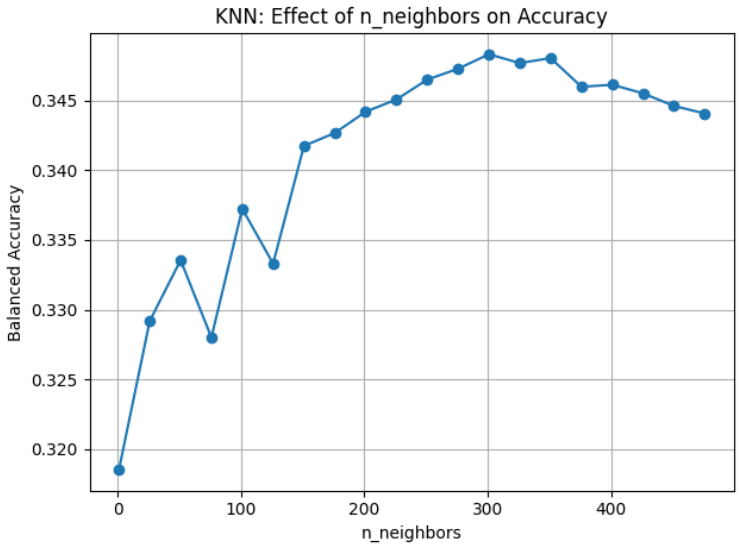

Figure 4: KNN accuracy for different k values

5.2 Decision Tree Classifier (DTC)

A Decision Tree is another straightforward approach for our binary classification tasks. Like with KNN, we perform a hyperparameter search to find the ideal settings. Increasing *max_depth* allows deeper nested decisions and improves the fit on the training data but also raises the risk of overfitting. To counteract that problem, we can also raise *min_samples_split* and *min_samples_split*, which forces each node or leaf to have a minimum number of samples before the split can occur. Despite this regularization, our best single-tree model still performs worse than the KNN on our noisy data. The peak balanced accuracy was only 0.28 (ranging from 0.23 to 0.28 max).

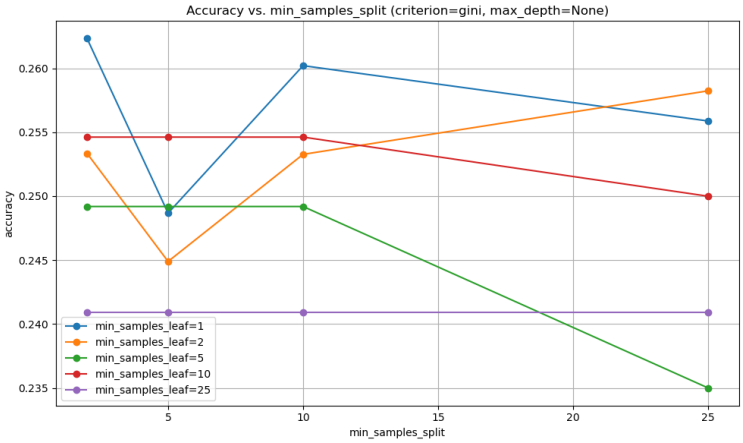

Figure 5: Results of the DTC for the label "Thunder"

5.3 Random Forest Classifier with Multi-Label Classification (MLC)

Since we have multiple classes in our data, we also evaluated a Random Forest classifier on all 58 labels and observed a significant improvement over both KNN and a single Decision Tree. Using 58 trees (*n_estimators=58*), a maximum depth of 30, min. samples per split of 10 and the "entropy" criterion, the model achieved a balanced accuracy score of 0.6472. This result shows that Random Forests combined with the Multi-Output Classifier works best with our data. These hyperparameters were then applied to train our final model on the full training dataset, and its predictions were evaluated across all classes. (see Section 6)

A frequently used approach for multi-label classification is to plot confusion matrices to visualize the performance of our model and to identify which classes are often confused with each other. For this reason we have plotted the top 6 classes with the highest confusion rates using the formula:

$$1 - \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{2}$$

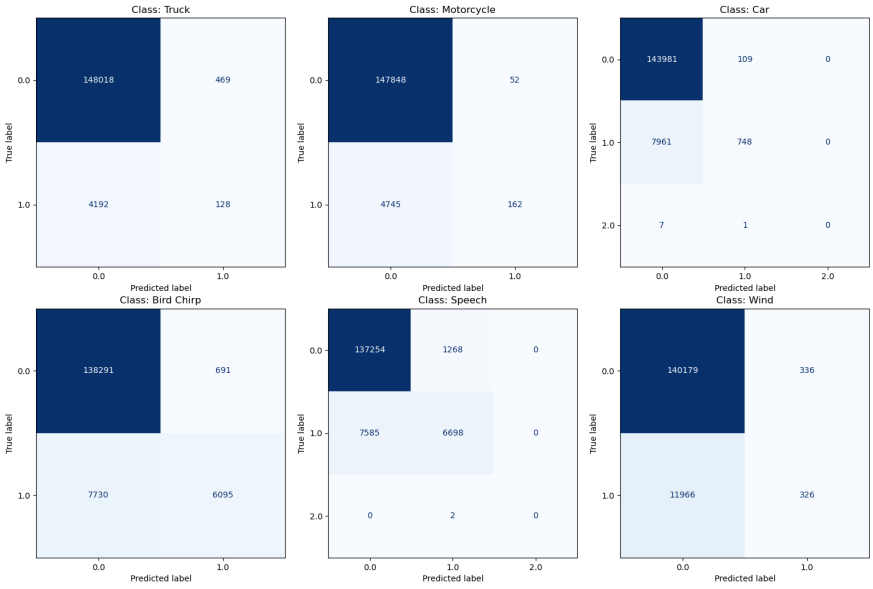

Figure 6: Confusion Matrix for the Random Forest Classifier with MLC

6 Analyzing Predictions

After we have found the "best" hyperparameters, we trained our model on the full training set (see Section 2).
At first the predictions of the model seemed pretty disappointing. The model did not capture different sound events or predicted the wrong labels. We assumed it would only work for simple examples:

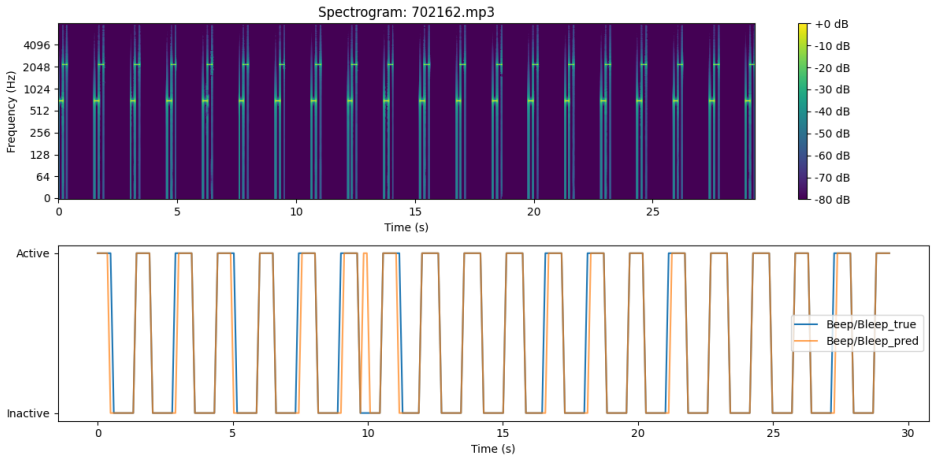

Figure 7: 702162.mp3: Ground Truth vs Prediction

The beep sounds in '702162.mp3' are clearly audible and it's a pretty unique sound. So we would expect the model to be able to capture this kind of sound event and indeed the predictions are pretty accurate.
However, after further investigation and comparing the true annotations with our prediction we found many samples where the model actually performed better than the annotators themselves. This also applies for more complex sound events. Lets look at two examples:

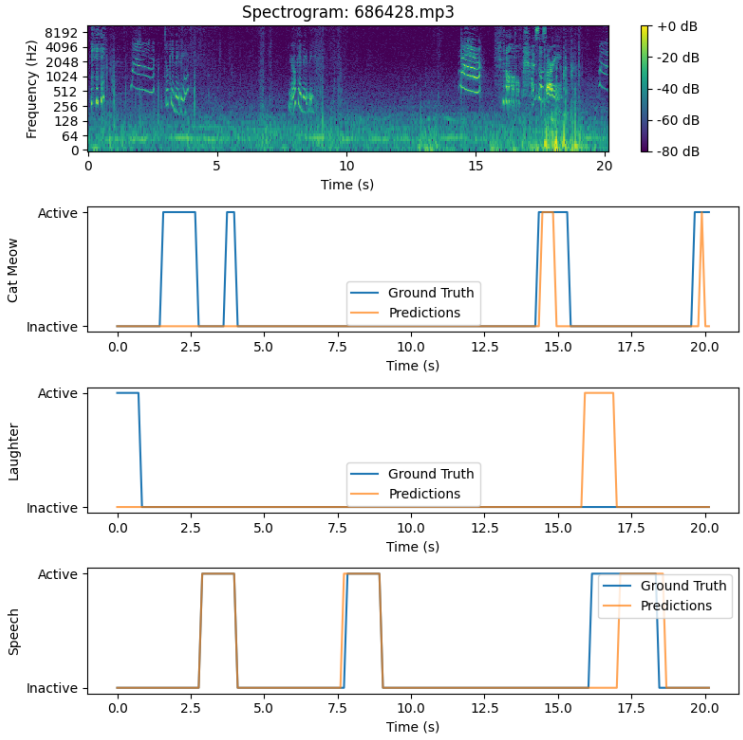

Figure 8: 686428.mp3: Ground Truth vs Prediction

Despite the fact that the model misses the first occurences of the meowing cat, it captured the talking person very accurately. When we listen to the audio file, we can clearly hear the person laughing at around 16s. This was captured by our model, but was described as 'speech' by the annotator.

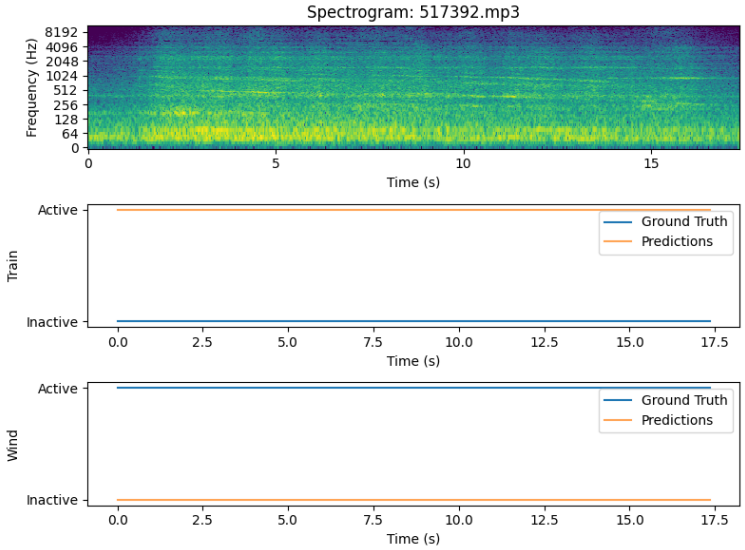

Figure 9: 517392.mp3: Ground Truth vs Prediction

This time, the Ground Truth label is completely wrong. The file definitely contains the sound of a passing train, which was correctly identified by the model.

Overall we see that the performance is not that bad. Some sound events are pretty hard to distinguish and our model completely fails, but there are also many examples where the performance is actually very good.

One of the biggest problems we encountered was that the predictions were very fragmented - short spikes with activity/inactivity of different labels. To mitigate this problem we should definitely consider smoothing the predictions as a postprocessing step. One simple solution could be a majority voting for a certain audio window (multiple combined audio frames).