Classification of Drum Set Components using Machine Learning Models

1st Johannes Franz Müller
University of Applied Sciences Cologne
Cologne, Germany
johannes_franz.mueller@smail.th-koeln.de

2nd Arn Jonas Dieterich
University of Applied Sciences Cologne
Cologne, Germany
arn_jonas.dieterich@smail.th-koeln.de

***Abstract*—This work presents a structured approach to the classification of drum set components using machine learning models. Furthermore, a variety of machine learning models and neural networks are evaluated to assess and identify models with superior performance.**

***Keywords*—audio classification, drum set, machine learning, neural networks, MFCC**

I. Introduction

The classification of drum set components is a challenging task due to the high variability in drum sounds and the potential presence of multiple components in a single audio sample. Potential use-cases include applications in music production, such as automated mixing and mastering and sample organization. The proposed approach involves the collection of a diverse dataset of drum samples, feature extraction using Mel Frequency Cepstral Coefficients (MFCC), and the training of machine learning models for classification. The performance of different machine learning models is evaluated based on their accuracy and precision. The results demonstrate the effectiveness of the proposed approach in accurately classifying drum set components and highlight the best-performing models for this task.

II. Data Acquisition and Data Set Creation

This section discusses the data acquisition process and the creation of the drum set component classification dataset.

A. Data Acquisition

The dataset consists of audio samples of different drum set components, including bass drum, snare drum, hi-hat, cymbal, and claps. These drum samples stem from various online sources, in the form of free contributions to the music community.

B. Data Discovery and Visualization

To train models in machine learning, it is important to look into the nature of the dataset to gain an insight into how the dataset has to be cleaned, to achieve the best possible accuracy.

The dataset comprises a total of 12324 samples. The statistical insights can be seen in the table below.

Statistic	Value
Total Samples	12324
Mean Duration	1.13 seconds
Median Duration	0.50 seconds
Max Duration	186.02 seconds
Min Duration	0.00 seconds

TABLE I
Statistical information calculated on the data set

As can be seen from the Max Duration (186.02 seconds), the dataset contains audio samples which are too long and should not be used for the dataset. The Min Duration (0.00 seconds) is also too low, which means that it is not possible to extract enough features from it, and therefore should not be used for the dataset.

To gain further insight it is also important to look into the distribution of the audio samples in relation to the classes, which can be seen in the figure below.

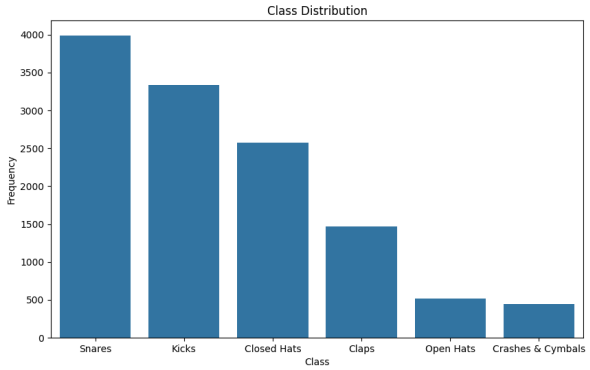

Fig. 1. Dataset Class Distribution

In the figure it becomes clear that the dataset has an unequal number of samples for each class and is therefore unbalanced. So it is important to also compare how models perform when the dataset is balanced.

The last information which should be taken into consideration is the sampling rate of the audio samples, which can be seen in Figure 2.

The figure reveals that most audio samples were recorded at a sampling rate of 44.1 kHz, some at 48 kHz and some at 22.05 kHz. So for cleaning the dataset, it is also important to resample the audio samples at one consistent sample rate.

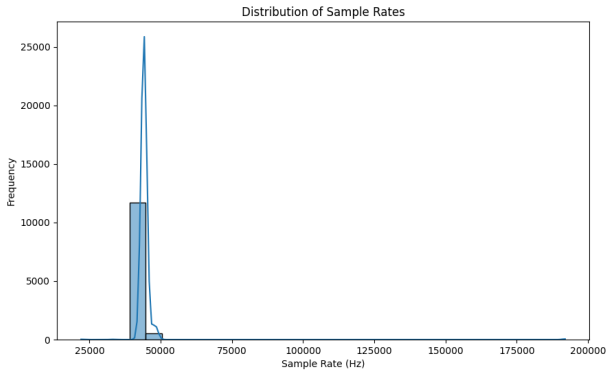

Fig. 2. Distribution of Sample Rate in the Dataset

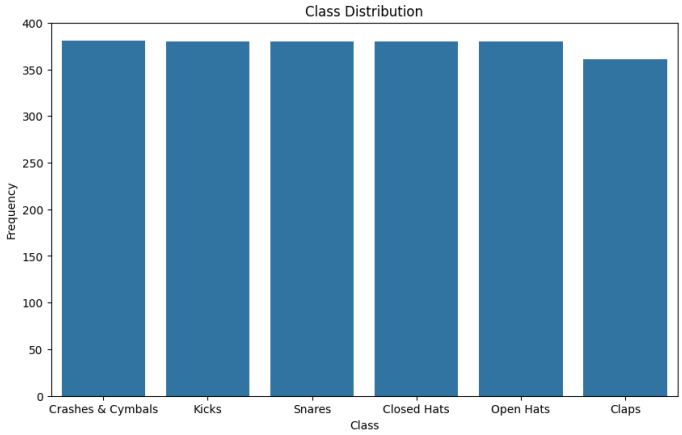

Fig. 3. Balanced Dataset Class Distribution

III. Data Preparation

This section outlines the steps involved in preparing the data for classification, including data balancing, labeling, data cleaning, feature extraction, and feature scaling.

A. Data Cleaning

The data cleaning process involves the removal of outliers based on duration criteria. Samples with a duration less than 0.1 seconds or greater than 8 seconds are removed from the dataset. Furthermore, the audio samples are resampled to a uniform sampling rate of 22.05 kHz to ensure consistency across the dataset.

The following table shows the new statistics for the cleaned dataset.

Statistic	Value
Total Samples	11239
Mean Duration	0.87 seconds
Median Duration	0.54 seconds
Max Duration	7.59 seconds
Min Duration	0.08 seconds

TABLE II
STATISTICAL INFORMATION CALCULATED ON THE DATA SET

B. Data Balance

To ensure fair representation of each drum component in the training process and also prevent the model from being biased towards the majority class, a second dataset is created from the cleaned unbalanced dataset, which has a balanced ratio of audio samples for each class. In Figure 1, the class *Crashes & Cymbals* has the least amount of audio samples, with around 375 audio samples.

The other classes must be adjusted to contain the same amount of audio samples. The following figure will show the distribution of the audio samples in the different classes for the balanced dataset.

As can be seen, every class now has around 380 audio samples, with the claps being around 370 audio samples. This dataset now can be used to compare the impact of the imbalance of the original dataset.

C. Labeling

As labels, the different classes (Kicks, Snares, Closed Hats, Open Hats, Crashes & Cymbals, Claps) from the dataset are used. The next step is to encode the labels using label encoding, where each drum component is assigned a unique integer value. This is a crucial step in preparing the data for classification, as machine learning models prefer numerical inputs for training.

D. MFCC - Mel Frequency Cepstral Coefficients

The Mel Frequency Cepstral Coefficients (MFCC) are extracted from the audio samples to represent the features and distinct characteristics of the individual audio samples. The MFCCs are a widely utilized feature extraction technique in audio signal processing, based on the extraction of spectral information from the signals. The MFCCs provide a compact and concise representation of the audio samples, in comparison to Mel spectrograms and linear spectrograms, as MFCCs require fewer coefficients per time frame.

This project utilizes the librosa library in Python to extract the MFCCs from the audio samples. The MFCCs are extracted using the parameters listed in the table below.

Parameter	Value
Number of MFCCs	40
FFT window size	2048 samples
Hop size	512 samples
Window function	Hann

TABLE III
MFCC EXTRACTION PARAMETERS

The **FFT window size** defines the size of the Fast Fourier Transform (FFT) window used during the analysis. A window size of 2048 samples means that each FFT operation will analyse 2048 samples of the audio signal at a time.

The **Hop size** specifies the number of samples between successive frames. A hop size of 512 samples means that the FFT windows will overlap, providing more detailed temporal resolution.

Window function is the window function applied to each frame before performing the FFT. The Hann window is used here to reduce spectral leakage by tapering the beginning and end of each frame to zero.

E. Feature Shaping

The dimensionality of the extracted MFCCs is reduced by calculating the mean of the MFCCs across the time axis. This results in a single vector representing the MFCCs of each audio sample. This reshaping is essential for feeding the MFCCs into the machine learning models and to increase performance while using fewer resources.

F. Train-Test Split

The dataset is split into training and test sets using an 80:20 split ratio. This is achieved through the use of the Python library Scikit-Learn. The training set is used to train the machine learning models, while the test set is used to evaluate the performance of the models. The shape of the datasets is (number of samples, number of features), which means a 2D array.

G. Feature Scaling

The MFCC features are scaled using the StandardScaler from Scikit-Learn. This step assists the data preparation process, as it ensures that the features are on the same scale. Having features with a large scale difference can negatively impact the performance of machine learning models and should therefore be avoided.

IV. Model Selection

- **K-nearest Neighbor (KNN)**
- **Random Forest (RF)**
- **Gradient Boosting Machine (GBM)**
- **Support Vector Machine (SVM)**
- **Feedforward Neural Network (FNN)**
- **Recurrent Neural Network (RNN)**
- **Convolutional Neural Network (CNN)**

A. Data Reshaping

For CNN, the data needs to be reshaped from a 2D array into a 4D array (number of samples, height, width, channels) to leverage the spatial hierarchies through convolutional layers, which can capture local patterns and spatial dependencies in the data.

For RNN, data needs to be reshaped from a 2D array into a 3D array (number of samples, timesteps, features) to model temporal sequences and dependencies in the data.

V. Model Training

A. KNN, GBM, SVM, RF Model Training

1) Cross Validation: The project makes further use of cross-validation to ensure the generalization of the models. Cross-validation is implemented using a 5-fold stratified split. This ensures that each fold contains an equal distribution of the drum components and prevents the model from overfitting to the training data. Cross-validation works by splitting the data into k-folds, training the model on k-1 folds, and evaluating the model on the remaining fold. This process is repeated k times, with each fold being used for validation once.

2) Hyperparameter Tuning: In the context of this project, hyperparameter tuning in combination with cross-validation was implemented for the models KNN, RF, GBM, and SVM. The methods to implement hyperparameter tuning are Grid Search for the models KNN, RF, and GBM, which trains the models with all possible combinations of defined parameters, and Randomized Search for SVM, which in the context of this project randomly samples 12 iterations of hyperparameter combinations from the grid.

Randomized Search is used for SVM because Grid Search, the algorithm of SVM involves solving a quadratic optimization problem and transforming the input space into higher-dimensional spaces, which leads to significantly increased computational complexity and processing time.

The following figure shows the parameters for the hyperparameter tuning.

Model	Hyperparameters
RF	n_estimators: [50, 100, 200] max_depth: [None, 10, 20, 30] min_samples_split: [2, 5, 10] min_samples_leaf: [1, 2, 4]
GBM	n_estimators: [50, 100, 200] learning_rate: [0.01, 0.1, 0.2] max_depth: [3, 5, 7] subsample: [0.8, 1.0]
SVM	C: [0.1, 1, 10] gamma: ['scale'] kernel: ['rbf', 'linear']
KNN	n_neighbors: [3, 5, 7, 9] weights: ['uniform', 'distance'] metric: ['euclidean', 'manhattan']

TABLE IV
HYPERPARAMETER CONFIGURATIONS FOR DIFFERENT MODELS

B. FNN, CNN, RNN Model Training

For the training of the Models FNN, CNN and RNN, the process is an iterative approach to train a model using a specified learning rate and number of iterations. The training process entails computing the loss using the cross-entropy criterion, backpropagating the error, and updating the model parameters with the Adam optimizer. Following each iteration, the average training loss is calculated.

1) Early Stopping: In the project, the number of iterations is 100, but this comes with the downside that the model tends to overfit and therefore deteriorate its performance. To prevent overfitting, early stopping is implemented. Early stopping allows for the halting of training if the training loss does not significantly improve for a number of consecutive epochs, as defined by the patience parameter. This is achieved through the use of a threshold to determine significant improvement. If the training loss does not decrease by more than this threshold for the defined patience iteration, training is stopped early.

VI. Model Evaluation and Visualization

A. Comparison of Model Accuracy

To evaluate the performance of the model, first the accuracy of the different models is compared and visualized in the following figure.

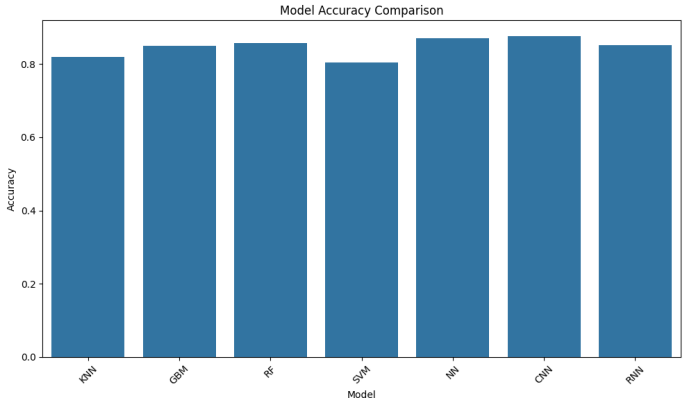

Fig. 4. Accuracy Comparison of Models

As can be seen, the best-performing models are CNN and GBM with an accuracy of around 88%, followed by the NN (86%), RNN (86%), RF (86%) and SVM (85%). The worst model is the KNN model, with an accuracy of 82%.

B. Confusion Matrix

To understand the accuracy, it is important to look into the confusion matrix of each model, which shows how many audio samples got predicted correctly for every call. The confusion matrix for the best-performed model (CNN) can be seen below.

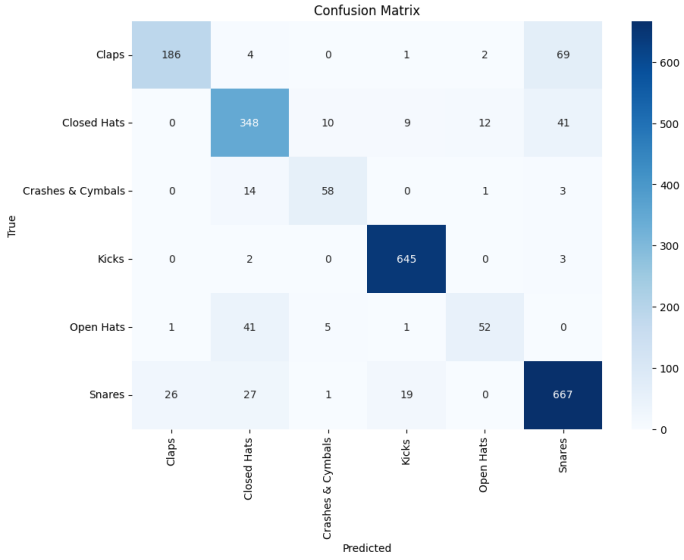

Fig. 5. Distribution of Accuracy for Classes (CNN)

The figure shows that the lowest accuracy is for the **Open Hats** (52%), followed by the **Claps** (71%), **Crashes & Cymbals** (76%), **Closed Hats** (83%), **Snares** (90%) and **Kicks** (99%).

When interpreting these results, especially for the classes with the lowest accuracy, two major factors are crucial. On the one hand, the imbalance of the dataset reduces the performance of the classes which are underrepresented. On the other hand, the audio samples in the class **Open Hats** and **Closed Hats** sound really similar, which is also true for the **Claps** and **Snares**. The best accuracy for the **Kicks** audio samples can be explained due to its lower frequency, which makes the sound unique.

C. Results

1) Impact of Hyperparameter Tuning: To evaluate the impact of hyperparameter tuning, the accuracy of the models with tuned hyperparameters against those with default configurations are compared. As result the KNN model's accuracy improved by 0.1655, the GBM model by 0.0.4862, the SVM model by 0.3341 and the RF model by 0.472, indicating that the default parameters work good with the dataset already, but underlines the impact of hyperparameter tuning as well.

2) Impact of Balancing the Data Set: Balancing the dataset by reducing the number of audio samples led to unexpected results. The accuracy of each model decreased between 8-14%. Therefore, simply removing audio samples from the overrepresented classes to achieve balance is not effective. Instead, the dataset should be balanced by adding new audio samples to the underrepresented classes.

VII. Conclusion

This report discusses the creation of a usable dataset of drum audio samples, including the cleaning and preparation of the data as well as the extraction of MFCCs features. The dataset then gets used to train and evaluate different models to classify the samples. At the end, the models are evaluated and compared. As a result, it can be said that the model accuracy is located between 82 - 88%. The best-performing models are thereby CNN and GBM. Also in the confusion matrix, it was shown that the highest error occurrence lies between Claps and Snares and Closed/Open Hats due to an imbalanced dataset and an equal audio sample sound.

As future work, there should be an implementation of cross-validation and hyperparameter tuning for the neural network models as well as a more complex implementation with more layers.

References

[1] "Tutorial — librosa 0.8.0 documentation," librosa.org. https://librosa.org/doc/main/tutorial.html

[2] IBM, "What are Convolutional Neural Networks? — IBM," www.ibm.com, 2023. https://www.ibm.com/topics/convolutional-neural-networks

[3] IBM, "What are Recurrent Neural Networks? — IBM," www.ibm.com, 2023. https://www.ibm.com/topics/recurrent-neural-networks

[4] A. Géron, "Praxis Einstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme". O'Reilly Media, 2017

[5] IBM, "What Are Neural Networks?," www.ibm.com, 2023. https://www.ibm.com/topics/neural-networks