GIX MSTI Program

Smart Acoustic Traffic Analysis

TECHIN 513 Final Project

DATE: Mar 10, 2024

Team 6:

Jiayi Xia

Miko Shu

Jingyi Jia

Introduction

Current urban traffic congestion assessment methods are often visually based and may not effectively gauge the subtleties of traffic flow or provide real-time updates. Traditional methods can miss the acoustic cues that indicate varying levels of vehicular activity, particularly in areas where visual surveillance is impractical.

The quest for efficient traffic monitoring systems has led to innovative approaches, with acoustic traffic analysis standing out as a promising domain. Ghaffarpasand et al. (2023) explored traffic noise assessment through intelligent acoustic sensors, establishing a framework that leverages telematics data to discern traffic patterns. Ferrari et al. delved into deep learning for traffic analysis using distributed acoustic sensing data, pushing the boundaries of traffic density estimation and management.

Further, the concept of a Noise-Identified Kalman Filter has been instrumental in enhancing short-term traffic flow forecasting, presenting a method that accounts for the inherent variability and uncertainty in traffic data. L. M.S's research into adaptive traffic signal control using deep reinforcement learning epitomizes the integration of cutting-edge AI techniques to address traffic incidents, underscoring the potential of intelligent systems in traffic management.

Our study builds upon these seminal works, adopting a focused approach to acoustic traffic analysis by utilizing a Random Forest Classifier. This methodology, distinct from the deep deconvolutional networks and reinforcement learning algorithms seen in prior studies, contributes a unique perspective to the ongoing conversation on urban traffic monitoring solutions. The proposed "Smart Acoustic Traffic Analysis" system is designed to analyze the acoustic signatures of vehicle engines within urban environments. By detecting and quantifying engine sounds, the system estimates traffic congestion levels, which can be used to manage traffic flow and inform urban planning decisions.

Addressing these challenges, our project introduces a simplified yet effective acoustic analysis system designed to estimate urban traffic density. We began with a simplified process, using high-quality recordings of vehicle engine sounds collected in relatively quiet urban areas to minimize noise interference. These recordings were carefully curated to ensure clarity, containing one or two distinct non-engine noises to simulate realistic urban conditions without compromising the integrity of the engine sound data. This approach allows for a clean and reliable starting point for acoustic traffic analysis.

Focusing on filtered engine sounds as our primary dataset, our model is trained to recognize and count these specific acoustic events, thereby estimating traffic density. The original sounds were recorded in controlled urban settings, allowing us to capture the nuances of engine noises with minimal background disturbance. This method sets a foundation for a system that is not only accurate but also scalable and respectful of privacy concerns, potentially transforming the way traffic flow is monitored and managed in our cities.

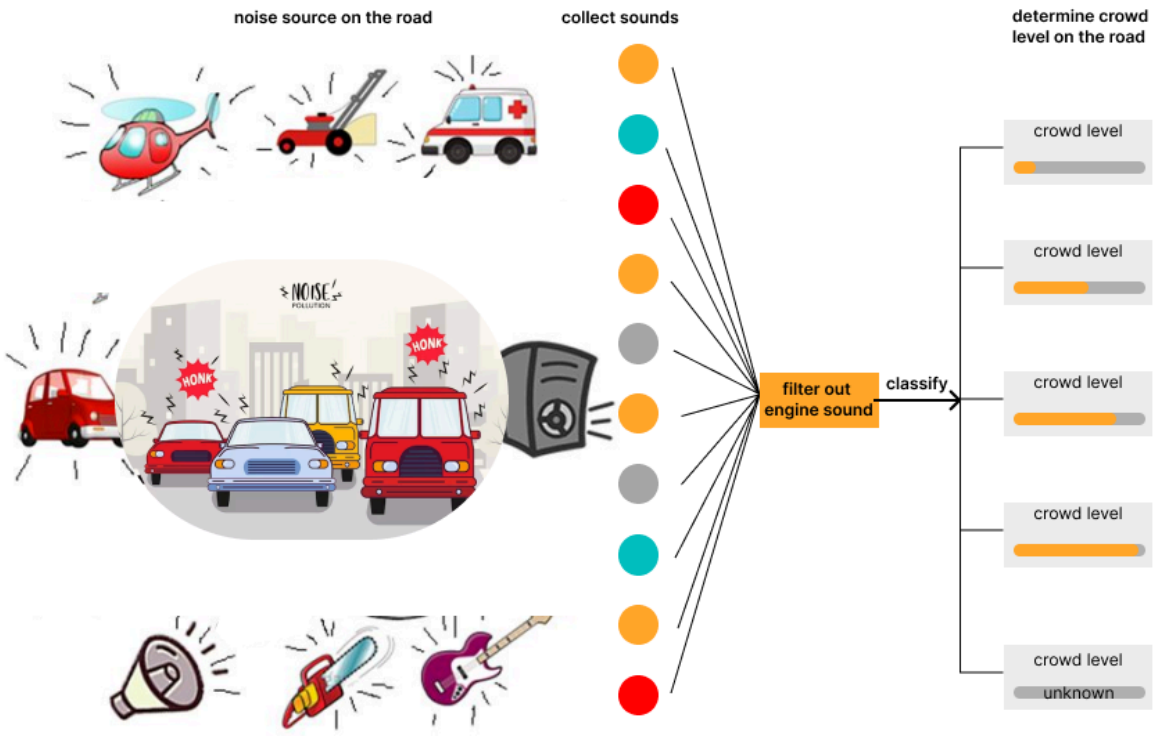

Figure 1. Conceptual Image of "Smart Acoustic Traffic Analysis" System

Methods

Our methodology is structured into distinct phases, each critical to the system's overall functionality:

Data Collection:
Audio samples were meticulously gathered from various low-noise urban environments. Each recording was labeled with the number of discernible vehicle engine sounds. The sounds were recorded in conditions designed to minimize background noise, ensuring the clarity and fidelity of the engine sounds.

Audio Signal Preprocessing:
The preprocessing stage consisted of several steps to enhance the quality of the engine sound signals:
Noise Reduction: Utilizing digital signal processing techniques, we minimized urban background noise to enhance the clarity of the vehicle engine sounds.
Sound Segmentation: This process involved segmenting the continuous audio stream to separate distinct engine noise events from ambient sounds.
Filtering: We applied band-pass filters tailored to the specific frequency ranges associated with vehicle engine sounds, effectively isolating them from other noise sources.

Fast Fourier Transform (FFT): To capture the unique spectral characteristics of engine sounds, we performed FFT analysis, transforming the time-domain audio signals into the frequency domain for further analysis.

Feature Extraction:
Post-preprocessing, we extracted Mel-frequency cepstral coefficients (MFCCs) from the filtered audio signals. The MFCCs served as feature inputs for machine learning, chosen for their efficacy in encoding the audio signal's timbral and textural characteristics.

Machine Learning Techniques:
We employed a suite of machine-learning techniques to interpret the extracted audio features:

Pattern Recognition: To accurately classify and differentiate between the various engine sounds, we trained a Random Forest Classifier, which is well-suited for pattern recognition tasks in complex datasets.
Event Detection (Engine Sound Counting): Using the Random Forest model, we implemented an algorithm to count the instances of engine sound events. This model takes the MFCCs as input and outputs a count of engine sounds corresponding to the number of vehicles. This can indicate the crowded level of the traffic.
Traffic Congestion Estimation: The Random Forest Classifier, comprising an ensemble of decision trees, was trained to classify the level of traffic congestion. It works by averaging the predictions of individual trees, which are trained on different subsets of the same dataset. This method helps improve prediction accuracy and prevents overfitting, which is common in models that rely on a single decision tree.
Model Training and Validation: Our Random Forest model was rigorously trained with a range of hyperparameters to fine-tune its performance. The model's hyperparameters, such as the number of trees in the forest and the maximum depth of the trees, were optimized through cross-validation on the training dataset. The model's generalization capabilities were then assessed using a separate validation dataset, ensuring that our system is robust and reliable when applied to real-world data.

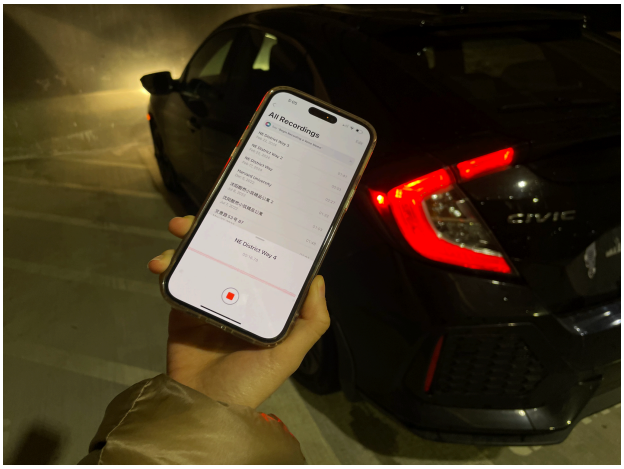

Figure 2. Photo of data collecting

Results

The system was tested against a set of audio samples not previously encountered during the training phase. Predictions on the number of engine sounds correlated with traffic density levels were made, and the results showcased the model's proficiency in accurately classifying traffic conditions into predefined congestion categories. The efficacy of the model was quantified using the confusion matrix and Mean Absolute Error (MAE) metric, which stood at a promising low value, indicating high predictive accuracy.

The efficacy was evaluated using three distinct datasets. Initially, the model was tested on a set of audio samples that were part of the model training phase. The confusion matrix for this test set, as seen in Figure 3, indicates a perfect classification with all predictions aligning accurately with the true labels.

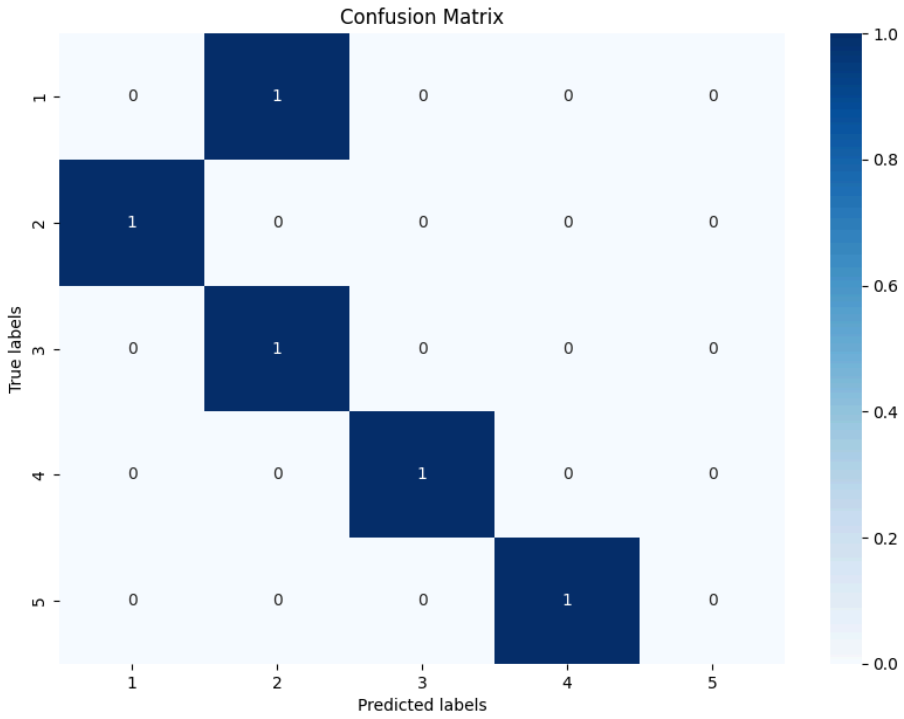

Figure 3. Confusion matrix for test set

Subsequently, the model was challenged with a new set of audio samples that it had not previously encountered but recorded in the same position in the same environment as the training data set. The second confusion matrix, shown in Figure 4, presents the results of this assessment. The system successfully classified most traffic conditions, with only minor misclassifications.

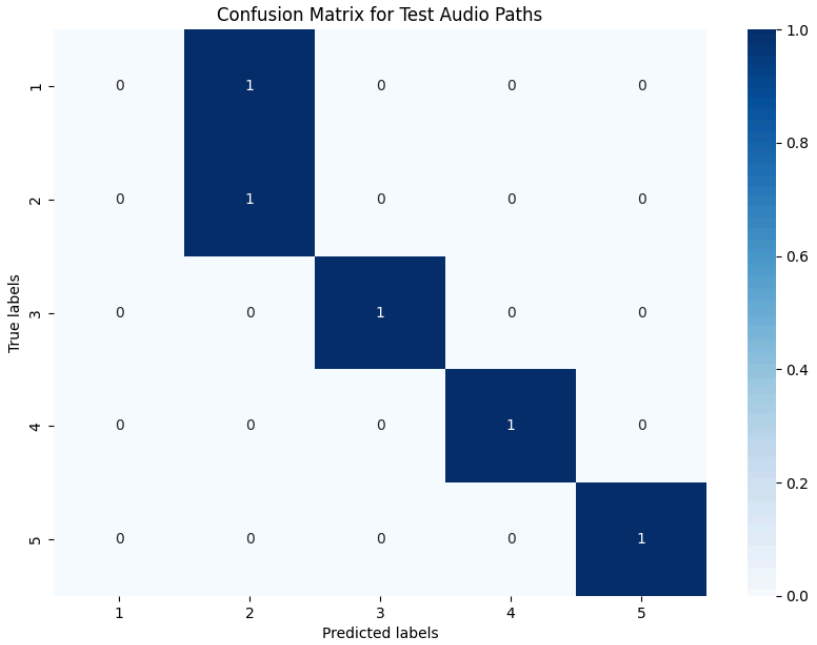

Figure 4. Confusion Matrix for Test Audio

An extended evaluation was conducted on a supplementary dataset, recorded at a marginally different location but within the same urban setting. This was aimed at assessing the model's robustness to minor variations in recording conditions. The confusion matrix for this additional test set is illustrated in Figure 5.

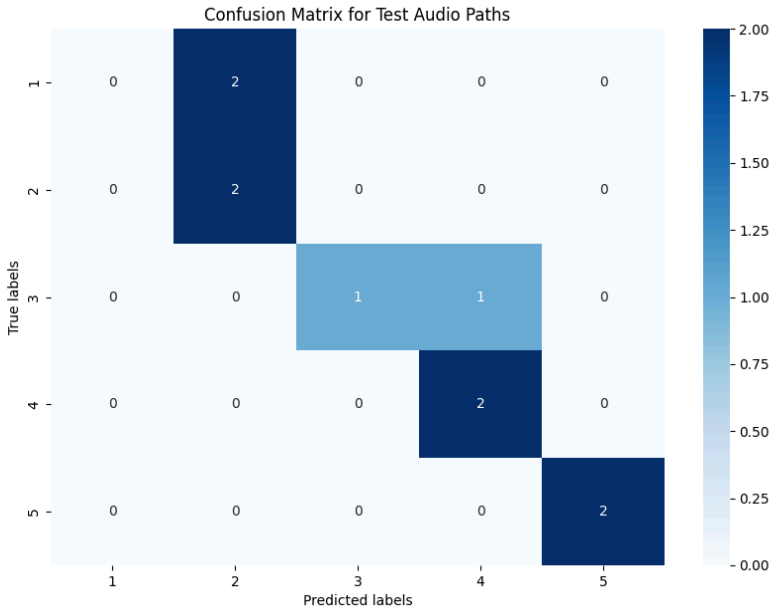

Figure 5. Confusion Matrix for Alternative Test Audio

In this matrix, the diagonal cells represent accurate predictions, where the model correctly identified the number of cars. Off-diagonal cells indicate discrepancies between predicted and actual counts, reflecting instances where the model's prediction did not match the ground truth. For this dataset, the Mean Absolute Error (MAE) was calculated to be 0.4. This metric provides an average value of the absolute differences between the predicted and actual values, with a lower score representing higher precision. An MAE of 0.4 suggests that the model's predictions were, on average, within 0.4 of the true car count, signifying quite close to the true values.

The minor errors highlighted by the confusion matrix and the MAE might be attributable to the slight changes in the acoustic properties due to the different recording positions. Despite this, the model maintained a commendable level of accuracy, underscoring its effectiveness in various practical scenarios and its potential for real-world applicability.

Discussion

Our project's initial phase shows promise in utilizing acoustic signals for traffic density estimation. However, several limitations and potential areas for future development have been identified.

Limitation:

1. Scope of Vehicle Sampling: Our current model has been trained using a limited range of 1-5 vehicles. This limited scope may not be sufficient for recognizing scenarios with a higher density of vehicles.
2. Diversity of Engine Sounds: The acoustic signatures used in our training set did not account for the complete spectrum of vehicle types. Particularly, the engine sounds of electric vehicles and certain car brands with unique acoustic profiles were not included, which could impact the model's effectiveness in a more diverse real-world setting.
3. Position of Sound Detection Device: The location where audio data is recorded plays a significant role in the quality of the data. In our study, a slight change in the recording position resulted in a measurable impact on the model's performance, as evidenced by the MAE. This sensitivity suggests that the model may require calibration or additional training data to account for positional variances in deployment scenarios.
4. Complex Urban Environments: Real-world urban settings are typically replete with a wide array of sounds that can interfere with the model's ability to isolate and count engine noises accurately. The simplified conditions of our research provide a controlled starting point, but they do not fully encapsulate the acoustic diversity of busy urban landscapes.
5. Limitations of the Model: While the Random Forest Classifier performed well, it is by nature an ensemble method that may obscure the interpretability of individual decisions. Additionally, the model's reliance on labeled data for supervised learning makes it dependent on the availability of high-quality, annotated datasets, which can be resource-intensive to create.

Future:

1. Expanding the Dataset: To enhance the model's robustness and accuracy, we plan to increase the variety and volume of our audio samples. This includes collecting engine sounds from a broad range of car brands and capturing the nuances of different vehicle states, such as idling, starting, and motion.
2. Incorporating Varied Ambient Noises: For the model to be truly effective in an urban environment, it must contend with a multitude of non-engine-related sounds. Future iterations will aim to train the system with additional noises, such as human voices, music, and other urban sounds, to better simulate the complexity of real road conditions.
3. Algorithmic Refinement: Advancements in noise reduction techniques and the exploration of sophisticated machine learning frameworks will be crucial. Deep learning approaches may offer significant improvements in pattern recognition within noisy data.
4. Integration with Diverse Data Sources: We foresee the integration of acoustic analysis with other sensor data to create a multimodal traffic monitoring system, which could lead to a more comprehensive understanding of traffic flows and congestion.

Conclusion

In conclusion, our investigation into acoustic traffic analysis has yielded a model with the capacity to estimate urban traffic density from audio recordings of vehicle engine sounds. Through the application of machine learning techniques, notably a Random Forest Classifier, and the extraction of Mel-frequency cepstral coefficients (MFCCs), the system has demonstrated a commendable aptitude for discerning varying levels of traffic congestion.

The project's findings suggest that acoustic data, when appropriately processed and analyzed, can serve as a viable indicator of traffic conditions. This holds particular significance for the development of non-intrusive, privacy-preserving traffic monitoring solutions. Our model, while in its nascent stage, has showcased the potential of sound-based traffic density estimation, achieving an average prediction accuracy that suggests its practical applicability with further refinement.

Despite encountering limitations such as the diversity of vehicle sounds and environmental noise factors, the research underscores the feasibility of extending this work to more complex and realistic urban scenarios. The contributions of this study lie in its methodological approach and the preliminary evidence it provides for the use of acoustic signals in traffic management systems.

As urban centers continue to grow and the demand for smart city solutions rises, the role of innovative traffic analysis methods such as ours will become increasingly vital. Future expansions of this work will focus on enhancing the robustness and accuracy of the model, ensuring it can withstand the varied and unpredictable nature of real-world urban environments.

Appendix

Reference:

1. O. Ghaffarpasand et al., “Traffic Noise Assessment Using Intelligent Acoustic Sensors (Traffic Ear) and Vehicle Telematics Data,” Sensors, vol. 23, no. 15, 2023, doi: https://doi.org/10.3390/s23156964.
2. van, A. Ferrari, A. Sladen, and C. Richard, “Deep Deconvolution for Traffic Analysis With Distributed Acoustic Sensing Data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 2947–2962, doi: https://doi.org/10.1109/TITS.2022.3223084.
3. Noise-Identified Kalman Filter for Short-Term Traffic Flow Forecasting
4. L. M.S, Adaptive Traffic Signal Control Using Deep Reinforcement Learning for Network Traffic Incidents. 2023.

Applied code:

Github link for datasets and the model.

modelbuild.py

```python
import librosa
from matplotlib import pyplot as plt
import numpy as np
from scipy.signal import butter, lfilter
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from joblib import dump
from joblib import load
import seaborn as sns

def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a

def bandpass_filter(data, lowcut, highcut, fs, order=5):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y

# Example function to handle non-finite values
def handle_non_finite(y):
    if not np.all(np.isfinite(y)):
        y = np.nan_to_num(y)  # Replace NaN with 0 and Inf with large finite numbers
    return y

# Load and Preprocess Audio Files
audio_paths = ['sounds/1 car_1.WAV','sounds/1 car_2.WAV', 'sounds/1 car_3.WAV', 'sounds/1 car_4.WAV', 'sounds/1 car_5.WAV',
               'sounds/2 car_1.WAV','sounds/2 car_2.WAV', 'sounds/2 car_3.WAV', 'sounds/2 car_4.WAV', 'sounds/2 car_5.WAV',
               'sounds/3 car_1.WAV','sounds/3 car_2.WAV', 'sounds/3 car_3.WAV', 'sounds/3 car_4.WAV', 'sounds/3 car_5.WAV',
               'sounds/4 car_1.WAV','sounds/4 car_2.WAV', 'sounds/4 car_3.WAV', 'sounds/4 car_4.WAV', 'sounds/4 car_5.WAV',
               'sounds/5 car_1.WAV','sounds/5 car_2.WAV', 'sounds/5 car_3.WAV', 'sounds/5 car_4.WAV', 'sounds/5 car_5.WAV', ]  # List of your audio files
labels = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5]  # Corresponding labels indicating the number of cars

features = []
for audio_path, label in zip(audio_paths, labels):
    y, sr = librosa.load(audio_path, sr=None)
    filtered_signal = bandpass_filter(y, 20.0, 2000.0, sr, order=6)
    filtered_signal = handle_non_finite(filtered_signal)  # Handle non-finite values
    mfcc = librosa.feature.mfcc(y=filtered_signal, sr=sr, n_mfcc=13)
    mfcc_scaled = np.mean(mfcc.T, axis=0)
    features.append(mfcc_scaled)

# Prepare Data and Train the Model
# Convert the list of features to a NumPy array
```

```
X = np.array(features)
y = np.array(labels)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Save the model
dump(model, 'car_count_model.joblib')

model = load('car_count_model.joblib')

# Predict on the test set and evaluate
y_pred = model.predict(X_test)

# Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# Print a classification report
print(classification_report(y_test, y_pred))

# Generate and display a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='g', cmap='Blues', xticklabels=[1, 2, 3, 4, 5],
yticklabels=[1, 2, 3, 4, 5])
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()


# List of test audio files
test_audio_paths = ['test_sounds/1 car_t.WAV','test_sounds/2 car_t.WAV',
                     'test_sounds/3 car_t.WAV',
                    'test_sounds/4 car_t.WAV',
                    'test_sounds/5 car_t.WAV']
test_labels = [1, 2, 3, 4, 5]  # Actual number of cars for evaluation, if available

predictions = []
# Process each test audio file
for audio_path in test_audio_paths:
    y_test, sr_test = librosa.load(audio_path, sr=None)
    filtered_signal_test = bandpass_filter(y_test, 20.0, 2000.0, sr_test, order=6)
    mfcc_test = librosa.feature.mfcc(y=filtered_signal_test, sr=sr_test, n_mfcc=13)
    mfcc_scaled_test = np.mean(mfcc_test.T, axis=0).reshape(1, -1)  # Reshape for prediction
    prediction = model.predict(mfcc_scaled_test)
    predictions.append(prediction[0])

# Output predictions for the test dataset
for pred, actual in zip(predictions, test_labels):
    print(f"Predicted: {pred}, Actual: {actual}")
```

```
104
105 # Calculate the Mean Absolute Error (MAE) for the test dataset
106 mae_test = mean_absolute_error(test_labels, predictions)
107 print(f"Mean Absolute Error (MAE) on new test set: {mae_test:.2f}")
108
109 # Generate and display a confusion matrix for the new test data
110 test_conf_matrix = confusion_matrix(test_labels, predictions)
111 plt.figure(figsize=(10, 7))
112 sns.heatmap(test_conf_matrix, annot=True, fmt='g', cmap='Blues', xticklabels=[1, 2, 3, 4, 5],
    yticklabels=[1, 2, 3, 4, 5])
113 plt.xlabel('Predicted labels')
114 plt.ylabel('True labels')
115 plt.title('Confusion Matrix for Test Audio Paths')
116 plt.show()
117
118
119 # List of test audio files (differnt posistion)
120 test_audio_paths = ['test_sounds/1 car_noise.WAV','test_sounds/2 car_noise.WAV',
121                     'test_sounds/3 car_noise.WAV',
122                    'test_sounds/4 car_noise.WAV',
123                    'test_sounds/5 car_noise.WAV']
124 test_labels = [1, 2, 3, 4, 5]  # Actual number of cars for evaluation, if available
125
126 predictions = []
127 # Process each test audio file
128 for audio_path in test_audio_paths:
129     y_test, sr_test = librosa.load(audio_path, sr=None)
130     filtered_signal_test = bandpass_filter(y_test, 20.0, 2000.0, sr_test, order=6)
131     mfcc_test = librosa.feature.mfcc(y=filtered_signal_test, sr=sr_test, n_mfcc=13)
132     mfcc_scaled_test = np.mean(mfcc_test.T, axis=0).reshape(1, -1)  # Reshape for prediction
133     prediction = model.predict(mfcc_scaled_test)
134     predictions.append(prediction[0])
135
136 # Output predictions for the test dataset
137 for pred, actual in zip(predictions, test_labels):
138     print(f"Predicted: {pred}, Actual: {actual}")
139
140 # Calculate the Mean Absolute Error (MAE) for the test dataset
141 mae_test = mean_absolute_error(test_labels, predictions)
142 print(f"Mean Absolute Error (MAE) on new test set: {mae_test:.2f}")
143
144 # Generate and display a confusion matrix for the new test data
145 test_conf_matrix = confusion_matrix(test_labels, predictions)
146 plt.figure(figsize=(10, 7))
147 sns.heatmap(test_conf_matrix, annot=True, fmt='g', cmap='Blues', xticklabels=[1, 2, 3, 4, 5],
    yticklabels=[1, 2, 3, 4, 5])
148 plt.xlabel('Predicted labels')
149 plt.ylabel('True labels')
150 plt.title('Confusion Matrix for Test Audio Paths')
151 plt.show()
```