# Improving Quantization using SPADE for Vehicle Audio Classification

Ashitabh Misra
University of Illinois
Urbana-Champaign
USA
misra8@illinois.edu

Isha Chaudhary
University of Illinois
Urbana-Champaign
USA
isha4@illinois.edu

Utkarsh Sharma
University of Illinois
Urbana-Champaign
USA
usharma4@illinois.edu

## ABSTRACT

Vehicular audio classification is an important task as part of driver assistance and vehicle diagnostics. Typically, the corresponding deep neural network classifiers need to be deployed on edge devices, which require small models due to their limited memory capacity. Hence, quantized models are preferred for this application. However, to get good performance, it is crucial to quantize with heuristics that capture domain-specific insights. In this work, we propose a novel heuristic to quantize the vehicle audio classifiers, wherein we leverage frequent patterns mined from the spectrograms of audio data corresponding to every class to identify the parts of the model that should be quantized more than the others.Our methodology produces adaptable models that can run on different precisions selected at runtime. We observe that pattern generated quantization models can give a comparable accuracy of 85% (32-bit variant gives86.4%)in an adaptable multiprecision setting.

**ACM Reference Format:**
Ashitabh Misra, Isha Chaudhary, and Utkarsh Sharma. 2024. Improving Quantization using SPADE for Vehicle Audio Classification. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Vehicle audio classification involves identifying and classifying audio samples from vehicles into different classes, based on their features. It is fundamental in numerous contexts, including driver assistance, vehicle diagnostics, and surveillance. Unveiling meaningful patterns within vehicle audio signals is paramount for comprehending driving behaviors, discerning vehicle conditions, and pinpointing anomalies. Typically, deep neural networks are trained on the spectrograms of audio samples to perform audio classification. However, as these models have to be deployed on edge devices, popular model compression techniques such as pruning and quantization are applied. These optimizations generally impose a tradeoff between the model's accuracy and size reduction. As the size of the model is reduced, its accuracy drops. Carefully chosen optimizations can, however, limit the loss of the model's accuracy. Hence, it is important to devise heuristics to guide the optimizations to apply during the training and testing of the models.

**This work**. In this work, we develop novel heuristics to guide the quantization of vehicular audio classification deep neural networks. In our current setup, we use the data for 3 vehicles, Tesla (Sedan), Silverado (SUV), and "Motor" (an open-top utility vehicle). Our initial experimental setup contains vehicles with an intuitively distinctive sound. Particularly, we investigate the feasibility and applicability of frequent and domain-specific patterns mined from real-world vehicular audio data to guide the quantization of Convolutional Neural Network (CNN) based classifiers. The extracted patterns can be considered as distillations of the given audio data and the frequent patterns in the data for a specific label would indicate the salient features of the samples. Hence, the quantization patterns can be designed such that the features not in the important set of features are quantized more heavily.

**Key challenges**. (1) To mine frequent patterns, we need to obtain discrete feature itemsets, which are not available. Hence, we need to design the features that will constitute the mined patterns. (2) Given a set of frequent patterns, the heuristic used to guide the quantization is unclear and needs domain-specific insights.

**Our approach**. To mine frequent patterns from audio sample data, we first identify the features of the spectrogram that can form meaningful patterns. We do so by binning the available frequencies and considering frequency bins to be dominant if they have high amplitude. We use the SPADE [15] pattern mining algorithm to identify the frequent sequential patterns in the data samples for every label. Our quantization algorithm uses the dominant frequency bins identified in the patterns as heuristics, wherein we quantize the values corresponding to these frequencies less than others.

**Contributions**. We make the following contributions.

- We mine frequent sequential patterns that characterize the vehicular audio data labeled by a particular class.
- We develop quantization algorithms that utilize the mined frequent patterns to selectively quantize parts of the audio classifier, such that the accuracy of the model does not drop by much, despite the reduction in the model's size.
- We observe that making distinct precision choices for different frequency bins does improve accuracy. With the right choice of bit allocation, were able to make our model adaptable to a wide variety of precisions. Specifically, we show that both pattern-aware quantization and naive splitting of frequency bins into high and low frequencies give us accuracy comparable to the floating point variant. We open-source our code and data at: https://github.com/ashitabh8/DataMiningProject/.


*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 2 RELATED WORK

### 2.1 Audio Feature Extraction

Audio feature extraction is a crucial step that affects the accuracy of the downstream application (like classification) [5]. General properties of audio that are relevant for pattern mining are discussed in [1][3]. We intend to focus mostly on pattern mining from a frequency representation and linear time-frequency representation given by Fast Fourier Transform and Short-time Fast Fourier Transform[3]. We intend to perform pattern mining on features like MFCC[9], Spectral Centroids[14], Average Energy, and Spectrograms as described in [5].

### 2.2 Constrained Frequent Pattern Mining

Constrained frequent pattern mining involves algorithms to extract the frequent patterns from given data and constraints on the mined patterns. A naive approach is to mine all frequent patterns and identify those that satisfy the constraints. However, prior work such as [6, 12] present algorithms that introduce the constraints early on in pattern mining algorithms to efficiently mine the constrained patterns. [7, 8, 13] present algorithms to mine constrained *sequential* patterns. Lemieux et al. [10] presents an enumeration-based approach and a tool to mine constrained patterns following given linear temporal logic [4] templates. We want to extend their algorithm to scale better and develop custom mining methods based on it for vehicular audio data.

## 3 SELECTED AUDIO FEATURES AND THEIR SIGNIFICANCE

This section delves into key features extracted from vehicle audio data and their importance in pattern discovery from vehicle audio samples.

- **Amplitude:** represents the strength or intensity of the audio signal. In the context of vehicle audio, it reflects the loudness or magnitude of sounds such as engine noises, tire screeches, or horn blasts.
  **Formula:**
  $$A = \max(|x(t)|)$$
  **Significance:** provides a fundamental measure of the presence and intensity of vehicle-related sounds. Patterns in amplitude variations can indicate changes in vehicle speed, engine load, or road conditions. For instance, sudden increases in amplitude may correspond to acceleration events or abrupt braking, while consistently high amplitudes may signify constant-speed driving.
- **Frequency Peaks:** identify dominant spectral components or peaks within the audio spectrum. These peaks represent characteristic frequencies associated with different vehicle-related sounds.
  **Significance:** enables the detection and categorization of specific vehicle sounds: such as engine revving, tire vibrations, or exhaust noises. Patterns in frequency peak distribution and intensity can reveal trends in engine performance, vehicle speed, or road surface conditions, aiding in the recognition of common driving scenarios or vehicle malfunctions.
- **Root Mean Square (RMS) Energy:** represents the average power of the audio signal over a specified time interval. It is calculated as the square root of the mean of the squared signal samples.
  **Formula:**
  $$\text{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x^2[n]}$$
  **Significance:** provides a normalized measure of the signal's power, accounting for its amplitude fluctuations over time. Patterns in RMS energy variations can reveal trends in sound intensity and temporal dynamics, aiding in the detection of distinct audio events and driving behaviors. By analyzing RMS energy patterns, it is possible to identify common driving scenarios, such as accelerating, cruising, or braking, based on their characteristic energy profiles.
- **Short-Time Fourier Transform (STFT):** decomposes a signal into its frequency components over short, overlapping time windows, providing a time-frequency representation of the signal.
  **Significance:** enables the analysis of temporal variations in the frequency content of vehicle sounds, capturing transient events and dynamic changes in audio characteristics over time. By computing STFT spectrograms, it is possible to identify time-varying spectral patterns associated with specific driving events, such as engine accelerations, gear shifts, or tire screeches. STFT facilitates extracting temporal-frequency features for machine learning-based classification and pattern recognition tasks in-vehicle audio analysis.
- **Spectral Centroid:** indicates the "center of mass" of the audio spectrum, providing insights into its brightness or tonal characteristics.
  **Formula:**
  $$C = \frac{\sum_{k=0}^{N-1} f[k] \cdot X[k]}{\sum_{k=0}^{N-1} X[k]}$$
  **Significance:** helps in distinguishing between different types of vehicle sounds based on their spectral distribution. Patterns in spectral centroid variations can reveal changes in engine speed, load, or operating conditions. For example, shifts in spectral centroid towards higher frequencies may correspond to acceleration events or gear changes, while lower centroid values might indicate engine idling or low-speed driving.
- **Spectral Bandwidth:** reflects the spread or dispersion of frequencies in the audio spectrum around its centroid.
  **Formula:**
  $$B = \frac{\sum_{k=0}^{N-1} (f[k] - C)^2 \cdot |X[k]|}{\sum_{k=0}^{N-1} |X[k]|}$$
  **Significance:** characterizes the spectral richness or complexity of vehicle sounds. Patterns in spectral bandwidth variations can indicate changes in engine performance, exhaust emissions, or vehicle speed. For instance, widening bandwidth may accompany increased engine load or higher-speed driving, while narrowing bandwidth might suggest smoother engine operation or cruising conditions.

- **Spectral Roll-off:** identifies the frequency below which a certain percentage of the total spectral energy is contained. **Significance:** Spectral roll-off provides insights into the high-frequency content of vehicle sounds, which is often associated with transient or abrupt events such as tire squeals, engine revs, or horn blasts. Patterns in spectral roll-off can help distinguish between different driving scenarios, such as urban traffic, highway cruising, or off-road conditions, based on their spectral energy distribution and high-frequency components.

# 4 FEATURE ENGINEERING

**Pre-processing**. We create chunks of audio data of the length of 0.5 seconds, which is originally recorded at a sampling rate of 8000Hz. On this data, we perform the following two steps:

- Down-sample the data to 1000Hz
- Remove the samples in the lower 30%ile of Root Mean Square Energy for each label. These samples are removed because they represent a very low amount of activity implying that the vehicle was far away during the recording.

Currently, we only use the Short-Time-Fast-Fourier-Transform (STFT) of each sample to perform our analysis. The parameters are nperseg = 125, Hop ratio= 0.5, and for windowing we use the 'haan' algorithm. According to the Nyquist Theorem, the maximum frequency we can analyze in this setup is 500Hz. The accuracy achieved by our baseline CNN model indicates that this is enough to distinguish between the labels. Our final input to both the CNN and SPADE pattern mining is a $63 \times 7$ tensor where the first dimension is the frequency bins, and the second dimension is the time bins.

Other features calculated but not used are Spectral Centroid and Spectral Roll-off.

# 5 FREQUENT PATTERN MINING

As we have sequential data of the amplitudes of various frequencies at different time steps of the audio samples, we mine frequent sequential patterns, using the SPADE sequential pattern mining algorithm [15]. We first form features and events consisting of itemsets from the features, from the raw data (Section 5.1) and then apply SPADE on the sequential data (Section 5.2) to obtain frequent sequential patterns. We discuss the insights and intuitive correctness of the mined patterns in Section 5.3.

## 5.1 Pre-processing Spectrograms

Our original data consists of time sequences, where each step is annotated with 63 features, as explained in Section 4 and demonstrated in Figure 2(a). We reduce the number of features by passing them through a MaxPool layer, where the maximum value over all non-overlapping windows of size 7 is computed. We treat the hence-obtained 9 values (Figure 2(b)) as features, which denote the maximum amplitude of the frequencies in a certain range. Our patterns are mined separately for each label in our data. Next, we describe the conversion of the samples corresponding to a given label to sequences from which we mine frequent patterns. We generate the patterns by forming itemsets from the features at each time step. Here, items denote a particular frequency range, and the presence of an item in an itemset indicates a higher amplitude (thus dominance) of that frequency range at a particular step. We identify the high amplitude frequency ranges, by thresholding the amplitudes with the $80^{th}$ percentile value of all the feature values corresponding to the particular label (Figure 2(c)). If the amplitude is higher than our threshold, the itemset will contain the item corresponding to the frequency range. This constitutes our entire binarization pipeline, which is summarized in Figure 2.

## 5.2 SPADE for Pattern Mining

The frequent patterns from the pre-processed sequences (Section 5.1) are mined using the SPADE algorithm. SPADE is a popular sequential pattern mining algorithm that operates on a vertical data format and leverages the Apriori candidate generation principle. We use the implementation of SPADE in the open-source package *PythonCSPADE*[1], which consists of a Python wrapper on the original implementation of the SPADE algorithm in C. The patterns mined using SPADE are shown in Table 1 for each of the 3 labels in our dataset — *Tesla, Silverado, Motor*. The minimum support threshold we set for the patterns shown is 50% of the sequences considered. We generate the patterns separately for the training partition (that is typically used to train classifiers for audio classification) and the testing partition of the original dataset. We observe that the frequent sequential patterns obtained from the training and testing partitions overlap significantly, confirming the correctness of our approach and the generalizability of the mined patterns.

## 5.3 Insights from Patterns Generated

We can gather some basic and intuitive insights from the patterns generated. First, the patterns generated lie in lower frequency bandwidths. This is in line with the common knowledge that lower-frequency sounds generally have less noise compared to higher-frequency signals. Small changes in low-frequency signals are more discernible, and our generated patterns seem to support that idea. Second, we see the high commonality between the patterns generated in the test and train set, which further supports the validity of our methodology and motivation to use this analysis to optimize machine learning models. Third, the patterns generated do not have any abrupt changes in frequency bins. The patterns only increment by a single bin which makes sense since a vehicle's sound signature does not change drastically.

Furthermore, a good sanity check for the patterns generated in Table 1 is the lack of erratic change in frequency bins. Since the motion of the recorded vehicles was at low speeds in a straight line there should not be such patterns. Finally, we observe either monotonic increase or decrease but not both in a single sequential pattern. This observation is in line with the *Doppler Effect*, observed frequency increases if the source is moving towards the observer and decreases when moving away.

Since the patterns are in line with physical principles and void of noise, they are an ideal candidate to act as an indicator that algorithmically assigns importance to different locations of the input.

[1] https://github.com/fzyukio/python-cspade

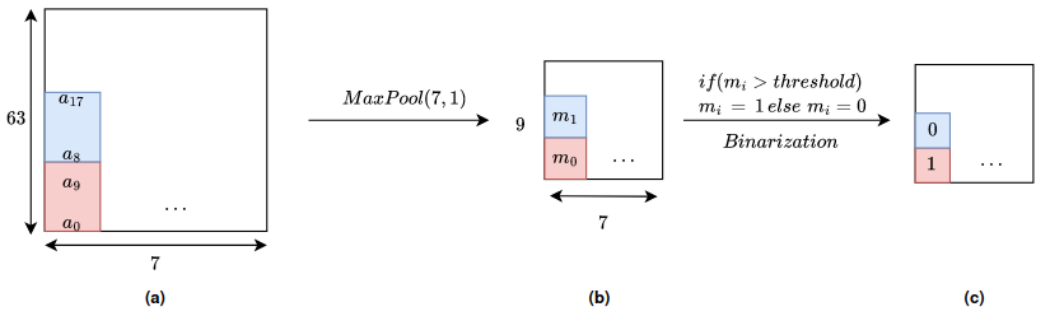


**Figure 1: Binarization Pipeline**

This is essential for different performance optimization problem applications such as neural network compression.

| | Tesla | Silverado | Motor |
|---|---|---|---|
| **Train Data** | (2) -> (2)<br>(2) -> (2) -> (2)<br>(2) -> (3)<br>(2) -> (3) -> (2)<br>(3) -> (2)<br>(3) -> (3) | (2) -> (2)<br>(2) -> (2) -> (2)<br>(2) -> (3)<br>(3) -> (2)<br>(3) -> (3) | (3) -> (3)<br>(3) -> (3) -> (3)<br>(3) -> (4)<br>(4) -> (3)<br>(4) -> (4) |
| **Test Data** | (2) -> (2)<br>(2) -> (2) -> (2)<br>(2) -> (3)<br>(2) -> (3) -> (2)<br>(3) -> (2)<br>(3) -> (2) -> (2)<br>(3) -> (3) | (2) -> (2)<br>(2) -> (2) -> (2)<br>(2) -> (3)<br>(2) -> (3) -> (2)<br>(3) -> (2)<br>(3) -> (2) -> (2)<br>(3) -> (3) | (3) -> (3)<br>(3) -> (3) -> (3)<br>(3) -> (4)<br>(4) -> (3) |

**Table 1: Sequences from Train and Test Data. Minimum support threshold** = 50%

## 6 MODEL ARCHITECTURE

We performed some initial baseline experiments, for classifying the labels using a **Fully Convolutional Network** [11]. The CNN is based on the All-CNN architecture. The high-level results are as follows:

- Training Accuracy: 92.4%
- Validation Accuracy: 89%
- Test Accuracy: 86.4%

We used the ADAM optimizer with a learning rate of $1e-3$, and CrossEntropyLoss as the loss function. A unique characteristic of our model was that the memory consumption of the model is $20KBs$. Since our downstream applications target model compression we focused and creating a small model but accurate model. The low memory consumption was achieved by eliminating Dense Linear layers and using Global Average Pooling in the last layer to perform classification.

## 7 APPLYING PATTERN MINING TO MODEL QUANTIZATION

We use the patterns mined from SPADE and the Binarization pipeline described in Figure 2 for Neural Network Quantization. We create a *Pattern Band* that is informed by the bounds generated by the observed patterns (Table 1). The pixels occupied in this pattern band are quantized differently from all other pixels in the input and intermediate activations. The fact that convolution operation is applied to spatially neighboring pixels and the output activation retains the order of the pixels means we can **Propagate the pattern band** through the network. We com

## 8 TRAINING A PATTERN AWARE STATIC QUANTIZED NETWORK

We use DoReFa[16] quantization scheme. Both Weights and Activations use DoReFa quantization(described in )

### 8.1 DoReFa Quantization Scheme

The DoReFa quantization scheme is as follows:

$$w_{quantized} = 2{*}quantize\left(\frac{\tanh(w_{input})}{2 * max(|\tanh w_{input}|) + 0.5}, NBIT\right) - 1 \quad (1)$$

The formulation gives us a quantized network with each entry being between $[0, 1]$. The $max()$ operation takes the maximum value of the entire tensor. The same formulation is used for weights and activations. *NBIT* is the target bit-width which can be anything $\geq 1$. We only consider the precisions $\geq 2$ in our experiments. The **quantize** function can be ay quantization function like Symmetric, Asymmetric, for Affine (Asymmetric with offset). We use symmetric quantization described as follows:

$$x_{quantized} = \frac{\lfloor x_{float} \times 2^{NBIT-1} \rceil}{2^{NBIT-1}} \quad (2)$$

The $\lfloor \cdot \rceil$ refers to the rounding function.

### 8.2 Straight Through Estimator

We can see that $\lfloor \cdot \rceil$ is a non-differentiable function. Hence we need a mechanism to resolve that. We use the **Straight through estimator** [2] technique which essentially lets the gradients flow

unchanged through a non-differentiable node of the Computational Graph. The approximate gradient computation is as follows:

$$\frac{\partial \mathcal{L}}{\partial r_i} = \frac{\partial \mathcal{L}}{\partial r_o}\frac{\partial r_o}{\partial r_i} = \frac{\partial \mathcal{L}}{r_o} \tag{3}$$

### 8.3 Training Algorithm

We use **Joint Quantization** to train a **Intra-layer** multi-precision quantized network. Where each layer consists of two different precision, the first in the **pattern band gap**, the second on the remaining pixels. This way we can fine-tune the model precision to significantly better precision compared to conventional static quantization and post-training quantization methods. We use a variant of the **Joint Quantization** technique where we calculate the loss of the network over multiple randomly generated bit allocations. **Bit Allocations** is the bit widths assigned to each layer and its *pattern band gap*.

**Algorithm 1** Train model with dynamic bit-width configurations

**Inputs:** M,T,V, $\mathcal{P}_{BandGap}$, $\mathcal{P}_{outer}$, $\mathcal{N}_{configs}$, N, L,U
**Output:** Trained model
$BestAcc_{val} = 0.0$
**for** $epoch = 1$ **to** N **do**
  $C$ = **GenerateConfigs**($\mathcal{P}_{BandGap}, \mathcal{P}_{outer}, \mathcal{N}_{configs}$) ▷ Generate $\mathcal{N}_{configs}$ random bit-width allocations
  **for** each batch in T **do**
    **for** each config in $C$ **do**
      **Setbitwidths**(model, config,L,U) ▷ Each layer will create a pattern band according from [ H × L, H× U] where H is the height of the layer
      **Forward pass:** $outputs = model(inputs)$
      **Compute loss:** $loss = criterion(outputs, labels)$
      **Backward pass:** $loss.backward()$
    **end for**
    **Update model parameters:** $optimizer.step()$
    $Val_{acc}$ = **Validate**(model, V, $C$) ▷ Validate model for each config in $C$
    **if** $Val_{acc} > BestAcc_{val}$ **then**
      $BestAcc_{val} = Val_{acc}$
      Save model
    **end if**
  **end for**
**end for**

Each configuration that is used for bit-allocation of a model is a list of tuples like $[(b_1, b_2) \cdots]$ where $b_1$ is the bit-width of the pattern band gap and $b_2$ is the bit width of all other pixels in the input and intermediate activation. We operate on the intuition that identified patterns are the *important* sections of the spectrogram and **GenerateConfigs** will sample from a multinomial distribution where the probability of sampling higher bit widths for pattern bands and vice versa for others.

## 9 EXPERIMENTS

To investigate the effectiveness of our pattern-guided quantization approach, we conducted a series of training experiments. We trained a CNN-based classifier using different configurations of bit-width probabilities and lower/upper bound ratios for the pattern and non-pattern bands.

| Bitwidth | Pattern Band | Non-Pattern Bands |
|---|---|---|
| 2 | 0.0 | 0.5 |
| 4 | 0.2 | 0.3 |
| 8 | 0.3 | 0.2 |
| 16 | 0.5 | 0.0 |

**Table 2: Probability distribution for sampling bit widths during training for Experiment A**

| Bitwidth | Pattern Band | Non-Pattern Bands |
|---|---|---|
| 2 | 0.0 | 0.4 |
| 4 | 0.1 | 0.3 |
| 8 | 0.2 | 0.2 |
| 12 | 0.3 | 0.1 |
| 16 | 0.4 | 0.0 |

**Table 3: Probability distribution for sampling bit widths during training for Experiment B**

| Bitwidth | Pattern Band | Non-Pattern Bands |
|---|---|---|
| 4 | 0.0 | 0.5 |
| 8 | 0.1 | 0.4 |
| 12 | 0.4 | 0.1 |
| 16 | 0.5 | 0.0 |

**Table 4: Probability distribution for sampling bit widths during training for Experiment B. The trend is reversed for Table 2 and 3**

### 9.1 Experiment A

In this experiment, we allocate a higher bit width to the pattern band while keeping the non-pattern band at lower bitwidths. The probability distribution is described in Table 2. This bit-allocation scheme prioritizes the important features captured by the pattern band while heavily quantizing the (possibly) less informative features in the non-pattern bands.

### 9.2 Experiment B

In this experiment, we introduce bit-width 12 and make the probabilities of sampling bit-width 16 higher for the pattern band gap. The lower and upper bound ratios remain the same as inferred from SPADE (Table 1).

### 9.3 Experiment C

In this experiment, we reverse the trend of probabilities mentioned in Experiments A & B. The probabilities are mentioned in Table 4.

| Configs | Non-Pattern Bitwidth Probabilities | Pattern Band Bitwidth Probabilities | Bound Ratios |
|---|---|---|---|
| 1 | {2: 0.5, 4: 0.3, 8: 0.2, 16: 0.0} | {2: 0.0, 4: 0.2, 8: 0.3, 16: 0.5} | (0.33, 0.67) |
| 2 | {2: 0.4, 4: 0.3, 8: 0.2, 12: 0.1, 16: 0.0} | {2: 0.0, 4: 0.1, 8: 0.2, 12: 0.3, 16: 0.4} | (0.33, 0.67) |
| 3 | {4: 0.5, 8: 0.4, 12: 0.1, 16: 0.0} | {4: 0.0, 8: 0.1, 12: 0.4, 16: 0.5} | (0.5, 1) |

**Table 5: Training Configurations**

| Configs | Bitwidth Probabilities | Avg Bitwidth | Bound Ratios | Avg Test Accuracy |
|---|---|---|---|---|
| 1 | Non-Pattern: {2: 0.5, 4: 0.3, 8: 0.2, 16: 0.0}<br>Pattern: {2: 0.0, 4: 0.2, 8: 0.3, 16: 0.5} | Non-Pattern: 3.8, Pattern: 11.2 | (0.33, 0.67) | 83.27% |
| 1a | Non-Pattern: {2: 0.6, 4: 0.2, 8: 0.1, 16: 0.1}<br>Pattern: {2: 0.0, 4: 0.2, 8: 0.4, 16: 0.4} | Non-Pattern: 4.4, Pattern: 10.4 | (0.33, 0.67) | 82.67% |
| 1b | Non-Pattern: {2: 0.6, 4: 0.2, 8: 0.1, 16: 0.1}<br>Pattern: {2: 0.1, 4: 0.1, 8: 0.2, 16: 0.6} | Non-Pattern: 4.4, Pattern: 11.8 | (0.33, 0.67) | 82.91% |
| 1a_mbr | Non-Pattern: {2: 0.6, 4: 0.2, 8: 0.1, 16: 0.1}<br>Pattern: {2: 0.0, 4: 0.2, 8: 0.4, 16: 0.4} | Non-Pattern: 4.4, Pattern: 10.4 | (0.5, 1) | 79.81% |
| 2 | Non-Pattern: {2: 0.4, 4: 0.3, 8: 0.2, 12: 0.1, 16: 0.0}<br>Pattern: {2: 0.0, 4: 0.1, 8: 0.2, 12: 0.3, 16: 0.4} | Non-Pattern: 4.8, Pattern: 12.0 | (0.33, 0.67) | 85.87% |
| 2a | Non-Pattern: {2: 0.3, 4: 0.3, 8: 0.2, 12: 0.1, 16: 0.1}<br>Pattern: {2: 0.1, 4: 0.1, 8: 0.2, 12: 0.3, 16: 0.3} | Non-Pattern: 6.2, Pattern: 10.6 | (0.33, 0.67) | 85.67% |
| 2b | Non-Pattern: {2: 0.35, 4: 0.35, 8: 0.2, 12: 0.05, 16: 0.05}<br>Pattern: {2: 0.05, 4: 0.05, 8: 0.2, 12: 0.35, 16: 0.35} | Non-Pattern: 5.1, Pattern: 11.7 | (0.33, 0.67) | 85.47% |
| 3 | Non-Pattern: {4: 0.5, 8: 0.4, 12: 0.1, 16: 0.0}<br>Pattern: {4: 0.0, 8: 0.1, 12: 0.4, 16: 0.5} | Non-Pattern: 6.4, Pattern: 13.6 | (0.5, 1) | 85.93% |
| 3a | Non-Pattern: {4: 0.3, 8: 0.3, 12: 0.2, 16: 0.2}<br>Pattern: {4: 0.2, 8: 0.2, 12: 0.3, 16: 0.3} | Non-Pattern: 9.2, Pattern: 10.8 | (0.5, 1) | 85.65% |
| 3b | Non-Pattern: {4: 0.45, 8: 0.45, 12: 0.05, 16: 0.05}<br>Pattern: {4: 0.05, 8: 0.15, 12: 0.4, 16: 0.4} | Non-Pattern: 6.8, Pattern: 12.6 | (0.5, 1) | 86.00% |

**Table 6: Testing Configurations and Average Test Accuracy. In Experiment A, higher bit-widths are allocated to the pattern band, prioritizing important features. Experiment B introduces bit-width 12 and increases the probability of sampling bit-width 16 for the pattern band. Experiment C reverses the probability trends from Experiments A and B, allocating lower bit-widths to the pattern band.**



### 9.4 Evaluation

To evaluate the effectiveness of our pattern-guided quantization approach, we conducted extensive testing experiments using various randomly generated bit allocations (Table ??). Each row was run 20 times and the average accuracy was given.

Our observations are as follows:

- In **Experiment A** where we keep the precision of the *non-pattern band* the lowest we are achieving the highest accuracy. This indicates that our identified patterns are truly able to capture the important parts of the spectrogram.
- In **Experiment B** observe that over-all we get better accuracy when we allow the non-pattern band to train for 16 bit precision as well, as it improves the flexibility of the model. It increases the variance we can afford in the type of bit allocations at runtime.
- In **Experiment C** we observe that even doing a half split, gives us comparable accuracy to pattern-aware methodology. This indicates that it is indeed useful to split the precision between different frequency bins when using convolution neural networks with spectrograms. If we look at row **3b** we see that indeed keeping the bitwidth for the pattern band gives the highest accuracy.

## 10 CONCLUSION

Through extensive experimentation, we are able to see that there is merit in exploring **Intra-layer frequency pattern** based quantization for adaptive quantization models. Both pattern-generated and naive splitting have the potential to give good accuracy in an adaptable setting. While most static and post-training quantization methods give good accuracy their runtime latency characteristics cannot be changed in real time on edge. Unlike our methodology where this is possible.

## 11 FUTURE DIRECTIONS

The main future direction in which this work can be applied is energy-conserving real-time models. Energy conservation and therefore, adaptable quantization has a crucial use in edge machine learning. More research needs to happen to find techniques to generate more intricate precision differentiating boundaries that may or may

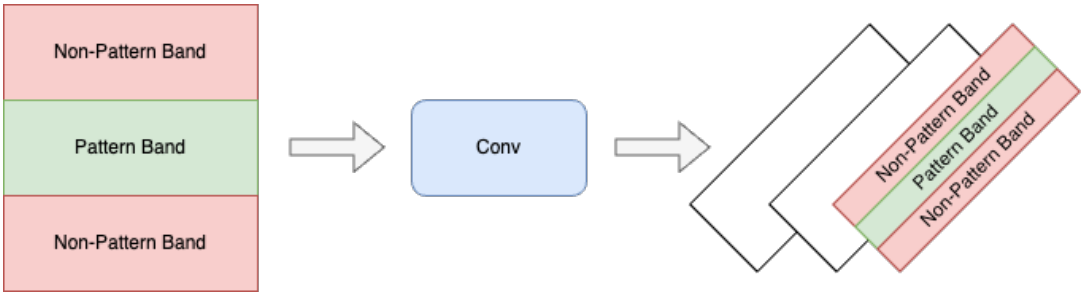


**Figure 2: Propagation of Bands across Conv layers. Since convolution retains the spatial order of pixels we preserve the lower and upper ratios and generate the pattern bands for each layer accordingly.**

not be structured. While unstructured quantization has merit in flexibility and intricacy, a structured approach will be more amenable to hardware-level optimizations.

### 11.1 Engine fault detection

Employing pattern mining techniques to identify unusual engine sounds presents a sophisticated approach to bolstering preventive maintenance efforts in the automotive industry. Future work can focus on developing robust fault detection systems by leveraging frequent patterns in engine audio data.

### 11.2 Traffic monitoring and management

Harnessing the power of pattern mining provides a comprehensive and intelligent approach to understanding and managing diverse traffic conditions. Future research can explore the integration of pattern mining with real-time traffic monitoring systems to generate alerts for traffic management authorities, enabling swift responses to congestion hotspots or emergencies.

### 11.3 Driver behavior analysis

Pattern mining can help identify distinctive acoustic patterns linked to specific driver actions such as aggressive driving, speeding, or abrupt braking. Future work can focus on developing advanced driver assistance systems that leverage these insights for continuous driver monitoring, providing valuable information for insurance companies and fleet management. These systems can foster safer driving practices and contribute to accident reduction.

### 11.4 Exploration of other audio features and pattern mining algorithms

While our work focused on using the Short-Time Fourier Transform (STFT) and the SPADE algorithm, future research can explore the effectiveness of other audio features and pattern mining algorithms for vehicular audio classification. Investigating the impact of different feature representations and mining techniques can lead to further improvements in model performance and efficiency.

By pursuing these future directions, we can enhance the applicability and impact of pattern mining in the domain of vehicular audio analysis, contributing to advancements in intelligent transportation systems, automotive safety, and driver assistance technologies.



## REFERENCES


[1] Francesc Alías, Joan Claudi Socoró, and Xavier Sevillano. 2016. A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds. *Applied Sciences* 6, 5 (2016). https://doi.org/10.3390/app6050143

[2] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR* abs/1308.3432 (2013). arXiv:1308.3432 http://arxiv.org/abs/1308.3432

[3] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. 2005. A Review of Audio Fingerprinting. *J. VLSI Signal Process. Syst.* 41, 3 (nov 2005), 271–284. https://doi.org/10.1007/s11265-005-4151-3

[4] Christian Colombo and Gordon J. Pace. 2022. *Linear Temporal Logic.* Springer International Publishing, Cham, 109–122. https://doi.org/10.1007/978-3-031-09268-8_9

[5] Ngoc Q. K. Duong and Hien-Thanh Duong. 2015. A Review of Audio Features and Statistical Models Exploited for Voice Pattern Design. *CoRR* abs/1502.06811 (2015). arXiv:1502.06811 http://arxiv.org/abs/1502.06811

[6] J. Han, J. Pei, and H. Tong. 2022. *Data Mining: Concepts and Techniques.* Elsevier Science. https://books.google.com/books?id=XmfvzgEACAAJ

[7] J. Han, J. Pei, and X. Yan. 2005. *Sequential Pattern Mining by Pattern-Growth: Principles and Extensions**. Springer Berlin Heidelberg, Berlin, Heidelberg, 183–220. https://doi.org/10.1007/11362197_8

[8] Amin Hosseininasab, Willem-Jan van Hoeve, and Andre A. Cire. 2018. Constraint-based Sequential Pattern Mining with Decision Diagrams. arXiv:1811.06086 [cs.LG]

[9] Johannes Krasser, Jakob Abeßer, Holger Großmann, Christian Dittmar, and Estefanía Cano. 2012. Improved music similarity computation based on tone objects. In *Proceedings of the 7th Audio Mostly Conference: A Conference on Interaction with Sound* (Corfu, Greece) *(AM '12)*. Association for Computing Machinery, New York, NY, USA, 47–54. https://doi.org/10.1145/2371456.2371464

[10] C. Lemieux, D. Park, and I. Beschastnikh. 2015. General LTL Specification Mining. In *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 81–92.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2014. Fully Convolutional Networks for Semantic Segmentation. *CoRR* abs/1411.4038 (2014). arXiv:1411.4038 http://arxiv.org/abs/1411.4038

[12] Jian Pei and Jiawei Han. 2002. Constrained frequent pattern mining: a pattern-growth view. *SIGKDD Explor. Newsl.* 4, 1 (jun 2002), 31–39. https://doi.org/10.1145/568574.568580

[13] Jian Pei, Jiawei Han, and Wei Wang. 2007. Constraint-based sequential pattern mining: The pattern-growth methods. *J. Intell. Inf. Syst.* 28 (03 2007), 133–160. https://doi.org/10.1007/s10844-006-0006-z

[14] A. Ramalingam and S. Krishnan. 2006. Gaussian Mixture Modeling of Short-Time Fourier Transform Features for Audio Fingerprinting. *Trans. Info. For. Sec.* 1, 4 (dec 2006), 457–463. https://doi.org/10.1109/TIFS.2006.885036

[15] Mohammed Zaki. 2001. Zaki, M.J.: SPADE: An efficient algorithm for mining frequent sequences. Machine Learning 42(1), 31-60. *Machine Learning* 42 (01 2001), 31–60. https://doi.org/10.1023/A:1007652502315

[16] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. 2016. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *CoRR* abs/1606.06160 (2016). arXiv:1606.06160 http://arxiv.org/abs/1606.06160

## 12 CONTRIBUTIONS

- **Ashitabh Misra**: Data preprocessing of the audio files. Implementing the Quantization Model. Designed the training algorithm for adaptable quantization. Contributed to the pattern-aware end-to-end model classification.
- **Isha Chaudhary**: Designed the conversion of audio spectrograms to itemsets for sequential pattern mining, conducted experiments with SPADE to extract the patterns, and analyzed the intuitive correctness and insights from the mined patterns.
- **Utkarsh Sharma**: Contributed to identifying crucial audio features. Implementing parts of the Dual Quantization Model (DualQuanv). Additionally, I trained and tested the DualQuanv model and performed the Experiments.