Weapon System Classification Using Acoustic Signatures

G. M. Chase, K. V. Gasser

March 19, 2020

Abstract

We apply several classification techniques to identify common weapons systems (AK-47, M4, FN FAL, and M2 50 cal) based on their unique acoustic signatures. We show that using Linear Discriminant Analysis (LDA) we can predict AK-47 gunshots with at least 81% accuracy, FN FAL shots with at least 73% accuracy, M4 shots with at least 78% accuracy, and M2 shots with at least 94% accuracy. We achieve these results based on low rank approximations of the audio data using Singular Value Decomposition (SVD).

1 Introduction and Overview

Interest in detection, classification, and localization of gunshots spans both military and civilian communities. The ability to classify the weapon used in a shot can be of value for several purposes. Quick identification of specific weapon systems firing could be used by the military to determine the appropriate response when fired upon. It could also be used by public safety officials to identify the type of weapon used in a crime. With more citizens carrying recording devices and capturing sound or video data there could be an increased likelihood of potential audio data that could be used as evidence in a crime. If it were possible to identify the characteristics of a weapon being used in a crime or an attack, it could aid in convictions or response to attack.

In this paper, we aim to show that we can predict with significant accuracy specific weapons given an audio signature captured in the form of an audio file. By using Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and several classification techniques on cleaned data, we show the ability to create a model to make such predictions. We demonstrate the use of Linear Discriminant Analysis (LDA), Naive Bayes, and Classification And Regression Trees (CART) as a means of labeling the type of weapon producing a given audio signal. Our hypothesis is that we are able to identify and classify a weapon accurately and that this information can be used by various entities for defense and protection of the public.

2 Theoretical Background

2.1 Gunshot Characteristics

The acoustic byproduct of a gunshot is divided into two components: the muzzle blast and shock wave. The muzzle blast is the result of rapidly expanding gases being ejected from the confined space inside the barrel. It typically propagates in all directions at a frequency at or below 500 Hz and lasts for 3-5 milliseconds [6, 4]. The dB strength of the signature is strongest in the direction the weapon is pointed and depreciates as the angle away from that azimuth increases[4]. The second source of information, the shock wave, exists if the bullet travels at a supersonic speed such that $V > c_{sound}$. While there are weapons that do not reach this muzzle velocity, all weapons considered in this paper meet this condition. The shock wave is a cone behind the bullet whose inner angle is proportional to the ratio V/c_{sound}. The wave propagates outward at the speed of sound and is characterized by an N-shaped signature between 1k Hz and 4k Hz[6].

The environment and orientation of the microphone in relation to the weapon induce more variability in the measured acoustic signatures. Because the shock wave is angled slightly in the direction the bullet is travelling, audio measurements taken from behind the weapon will only detect the muzzle blast. The ground, obstacles, and moisture content in the air may also reflect or absorb the sound waves causing further variation from the ground truth. The effects of these reflections and absorption's are amplified the further the detector is from the shooter[4].

Our data collection takes a number of these factors into consideration to ensure consistent classification accuracy. Recordings were taken both behind and in front of the shooter and at distances from < 1m to 100m. The orientation of the microphone was also adjusted to be both pointed at the shooter and facing the opposite direction. Unfortunately, little is known about the environment the weapon was shot in to include both terrain and weather characteristics. Even though it is safe to assume the shots were all recorded on a standard, open range, because the recordings are produced by multiple organizations, the weather conditions probably vary slightly throughout the data.

2.2 Transforming the Data

Before identifying features to be included in the classification training sets, we transformed the data from traditional audio recordings into spectrograms with a 64×64 pixel resolution. We used a windowed step-function to produce these spectrograms because of its exceptional ability to distinguish between individual shots within a burst of fire. We also performed edge detection by decomposing the spectrograms into wavelet basis functions using the Haar wavelet. During this process we further reduced the resolution to 32×32 pixels.

2.3 PCA

Principal Component Analysis (PCA) can be done by performing an eigenvalue decomposition. However, it can also be done using Singular Value Decomposition (SVD) which we do using Matlab. The process of PCA involves investigating whether your data has any correlations and if so, can we transform it into an alternate coordinate system where we can make more sense of the data. The goal is to find a collection of vectors, or Principal Components to maximize the variance of these components.

Any matrix has an SVD and is represented by the equation

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{*} \tag{1}$$

where $\mathbf{X}$ is the original data, $\mathbf{U}$ contains information about frequency content and time in relation to each other, and $\mathbf{V}$ is a description of how much $\mathbf{U}$ is needed to reconstruct a particular audio sample. The $\mathbf{\Sigma}$ matrix is a diagonal matrix containing the significance of each component[3]. This equation corresponds with the `svd(x)` as used in the Matlab code in Appendix B. And the output variables `u,s,v` correspond with the $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ above. The purpose of the SVD is to represent data in a coordinate system with no redundancy.

Because we compute the SVD for three separate experiments, the matrices $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ will include subscripts to denote which experiment they are in relation to. The subscripts SS, B, and C correspond to Experiment 1: Single Shots, Experiment 2: Bursts, and Experiment 3: Combined Shots, respectively.

2.4 Classification

The classification techniques we use in this paper are LDA, Naive Bayes, and CART prediction models.

Linear Discriminant Analysis (LDA) seeks to construct projections onto which the variance amongst the intra-class data is minimized and the distance between the class means is maximized. A decision line is then drawn between the classes to aid in classification.

The theory of Naive Bayes involves classifying data based on Bayes Theorem and the assumption that features are statistically independent. By using this ratio we are able to classify the data if we know the probability of our outcomes based on the training data set.

$$P(\theta|\mathbf{D}) = P(\theta)\frac{P(\mathbf{D}|\theta)}{P(\mathbf{D})} \tag{2}$$

Classification And Regression Trees (CART) employ a series of binary choices at the conclusion of which a decision or "bin" is reached and a prediction is made. Algorithmically, the goal is to select the moments of choice where the probability of a misclassification is lowest and establish the bins or terminal nodes at points where it no longer becomes possible to significantly reduce misclassifications[1].

Weapon (Caliber)	AK-47 (7.62x39mm)	FN FAL (7.62x51mm)	M4 (5.56x45mm)	M2 (12.7x99mm)
Audio Files	32	20	38	19
Bursts	158	54	139	153
Single Shots	32	50	32	11
Total Samples	190	104	171	164

Table 1: Weapon calibers and sample sizes. All samples are 0.7s in length.

3 Algorithm Implementation and Development

3.1 Database

Table 1 contains the list of weapons found in our database. The table includes their caliber, sample types, and sample sizes. The data is originally contained in a relatively small set of audio files containing numerous shot events. We broke the files into their individual events and categorized them as either bursts of fire, which we define as two or more shots, or as single shots. The database size is comparable to other gunshot event studies found in the literature[2].

3.2 Pre-classification Processing

We began with 109 .wav files containing acoustic signal data from various weapon firings. Our sound files contained various amounts of gunshots with bursts of fire ranging from 1 to 20 rounds fired per shot event. In order to get the most accurate results, we synchronized the beginning of each shot type as close as possible. As shown in the Matlab Code, splitFiles script, we split all of our data files when there was a lapse in shots anywhere from .7-1.5 seconds depending on the timing of the shots fired in the original sound file. One problem we faced was that some of the original sound files contained two channel data while others contained one channel. That is accounted for in the matlab code and these two cases were processed separately. Multiple attempts were used to devise the right algorithm as shown in the cleanwhitespace script, also included in the matlab code. Ultimately, we decided to split the files in order to synchronize the data and provide more accurate results with the SVD.

```
Algorithm 1: Preparing Data
for $j = 1$ : number of files do
    Read in the audiofile data.
    Set a signal threshold to determine at what signal level the shots are not firing
    for $forj = 1$ :datalength do
        Determine where the data point is in the signal: in the middle of the shot, or in the whitespace
        If the datapoint is in the middle of a shot, keep recording
        If the datapoint is in the whitespace, and it has been there for 1 second, then stop recording and
        save the file
    end for
end for
```

After isolating each shot event, we converted them into wavelet expanded spectrogram images and compiled the resultant data into a single matrix $\mathbf{X}$ with size 1024 $\times n$ where n is the number of samples used for a particular experiment. Despite Maher's suggestion that the muzzle blast holds limited value for classifying weapons, we tested our classification algorithms for spectrograms carrying data from 0-1k Hz and 0-4k Hz[4].

All of the data was then used for the SVD.

3.3 Classification

The algorithm used to build the classification models is shown in Algorithm 2. For all classes, with the exception of the M2 50 cal single shots, we left seven samples out for each training iteration to test on. We used the leave-one-out method for the M2 single shots because of how few samples existed within the class.

Algorithm 2: Training and Cross Validation

```
for $j = 1$ : number of desired trials do
    Create a vector $q$ of integers from 1 to n samples for each class and randomly shuffle it.
    Create a matrix for each class containing the number of elements of $\mathbf{V}$ to be included in training
    Assign $i$ training components where $i < n$ and are the first $i$ positions in $q$
    Assign the remaining members of each class's data as the testing components
    Create a vector of labels equal in size to the training set
    Train LDA, Naive Bayes, and CART prediction models
    Collect predictions of test data classes as columns
end for
Find the accuracy for each class within each model
```

4 Computational Results

4.1 Experiment 1: Single Shots

A PCA of single shot events in our database shows a very gradually depreciating set of diagonal components of $\mathbf{\Sigma}_{SS}$ after the fourth component. This is shown in Figure 1. The first four elements account for 21.5% of $\mathbf{\Sigma}_{SS}$ for the muzzle blasts and 21.2% for both the muzzle blasts and shock waves. The first 15 elements contain 37.6% of $\mathbf{\Sigma}_{SS}$ for muzzle blasts and 38.7% for the combined acoustic data. In both cases, 95% of $\mathbf{\Sigma}_{SS}$ is contained in the first 81% of its diagonal elements.

Figures 2a, and 2b show the first 12 modes of $\mathbf{U}_{SS}$ and the first 3 modes of $\mathbf{V}_{SS}$, respectively for the combined muzzle blast and shock wave trial.. The dominant feature is low frequency content across the first four modes $\mathbf{U}_{SS}$. This may explain both the tendency for M4 and AK-47 samples to cluster tightly, as well as the fact that many 0-1k Hz models performed equally or better than their 0-4k Hz model counterparts.

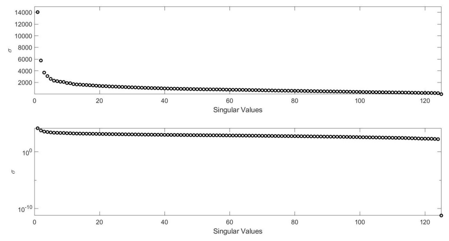
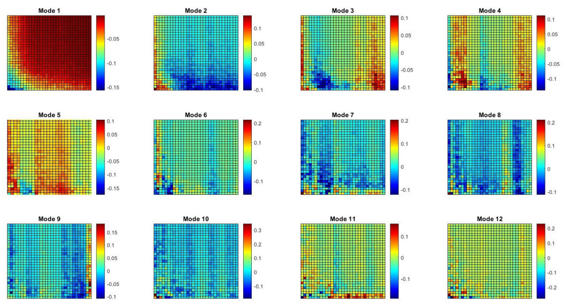
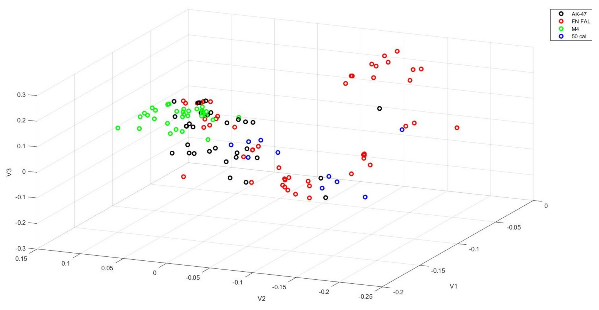


Figure 1: $\mathbf{\Sigma}_{SS}$ values for Experiment 1 when evaluating the 0-4k Hz frequency range.



(a) The first 12 modes of $\mathbf{U}_{SS}$ for Experiment 1.



(b) The first 3 modes of $\mathbf{V}_{SS}$ for Experiment 1.

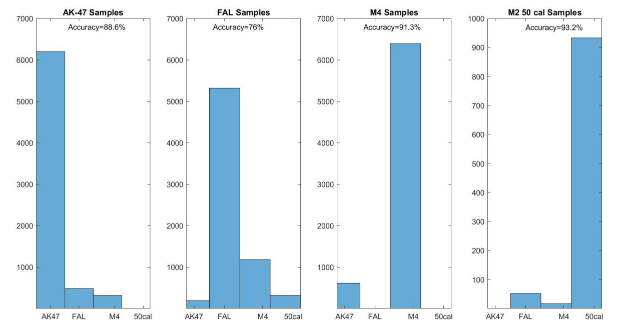
Figure 2: Principal components for individual shots when evaluating the muzzle blast and shock wave.

More so than in Experiment 2, the increase in the number of features considered from 4 to 15 played a substantial role in improving our classification models' accuracy. In several cases, this accuracy improvement

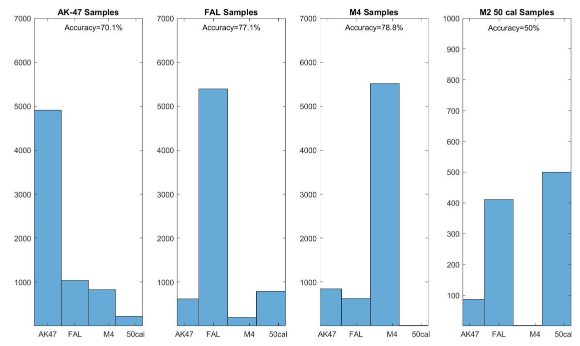
Model	Hz Range	Modes $\mathbf{V}_{SS}$	AK-47	FN FAL	M4	M2
LDA	1k Hz	4	44-50%	27-31%	91-93%	54-57%
LDA	1k Hz	15	87-90%	81-85%	91-94%	58-66%
LDA	4k Hz	4	60-68%	29–38%	89-92%	57-66%
LDA	4k Hz	15	86-89%	75-78%	91-94%	90-95%
NB	1k Hz	4	41-48%	56-63%	93-94%	35-41%
NB	1k Hz	15	75-79%	69-74%	94-95%	33-42%
NB	4k Hz	4	51-55%	44-54%	82-86%	51-62%
NB	4k Hz	15	59-63%	71-76%	81-87%	74-87%
CART	1k Hz	4	64-71%	66-74%	69-74%	42-48%
CART	1k Hz	15	74-79%	78-83%	76-79%	65-72%
CART	4k Hz	4	66-75%	75-82%	76-82%	46-54%
CART	4k Hz	15	64-72%	84-88%	74-81%	60-66%

Table 2: Cross-validation accuracy scores for each model during Experiment 1. The accuracy ranges include scores from 5 sets of 100 iterations for each model.

is more than double that of the lower rank model's results. Accuracy scores can be found in Table 2. The biggest distinguishing feature between the models was how well they handled the small number of training samples for the M2 50 cal. Because there were only 11 samples in this category, we chose the leave-one-out validation method for this class. Nevertheless, the small size of the M2 training set encouraged wide variance in accuracy both within a given model and across all 12 models. Because it performed the best at classifying the M2, the LDA for 0-4k Hz with 15 modes $\mathbf{V}_{SS}$ is considered the most successful and is shown in Figure 3a. The CART results at 0-4k Hz with 4 moves $\mathbf{V}_{SS}$ are shown in Figure 3b for comparison.



(a) The LDA classification results with 15 features $\mathbf{V}_{SS}$.



(b) The CART classification results with 4 features $\mathbf{V}_{SS}$.

Figure 3: Classification results after 1000 iterations for 0-4k Hz for Experiment 1.

4.2 Experiment 2: Bursts

The burst PCA reveals a set of $\mathbf{\Sigma}_B$ components for the 0-4k Hz trial shown in Figure 4. The first four σ values are the most dominant and account for 12.6% of the energy of $\mathbf{\Sigma}_B$. The first 20 components of $\mathbf{\Sigma}_B$ account for 25.9%, and the first 254 (50% of $\mathbf{\Sigma}_B$) of the σ values account for 86.8% of $\mathbf{\Sigma}_B$. This is in part because the elements of $\mathbf{\Sigma}_B$ from 430 onward surpass numerical round-off and are essentially zero.

An analysis of $\mathbf{U}_B$ and $\mathbf{V}_B$ are similarly revealing. Even before seeking to classify the bursts based on their respective classes, they clearly separate themselves into sub-components of the database. Figure 5b shows the values for the first 3 modes of $\mathbf{V}_B$ for each sample. M2 50 cal samples are the most clearly separated, but closer evaluation also reveals that FN FAL samples primarily exist in an outer shell surrounding the rest of the data. M4 samples concentrate in two clusters. One cluster is segregated enough to encourage distinction, but the second cluster is approximately 50% mixed with a cluster of AK-47 samples. This may in part explain the tendency to misclassify AK-47 samples as M4 samples and vice versa when only considering the first 4 modes of $\mathbf{V}_B$ as shown in Figure 6b. Because there was very little difference between the 0-1k Hz and 0-4k Hz PCAs, only the 0-4k Hz PCA results are shown and discussed in detail.

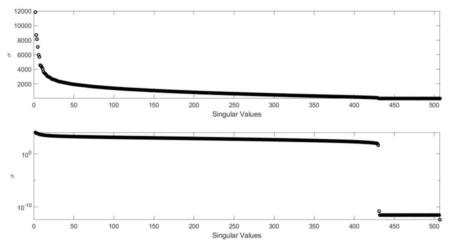
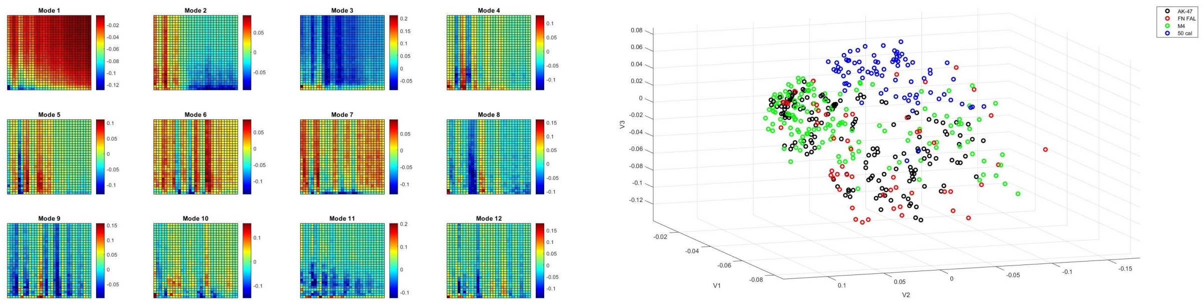


Figure 4: $\mathbf{\Sigma}_B$ values for Experiment 2 when evaluating the 0-4k Hz frequency range.

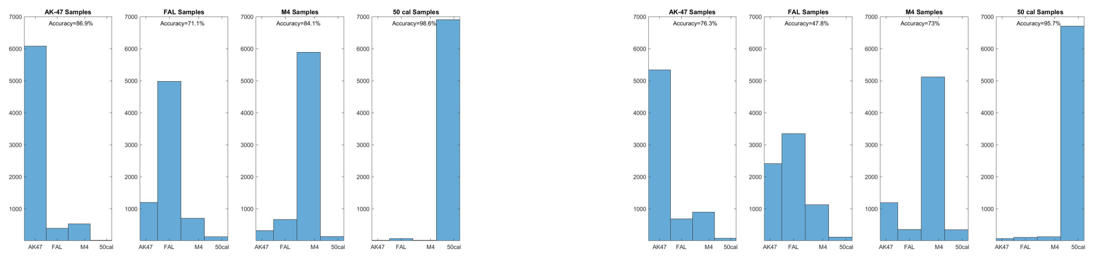


(a) The first 12 modes of $\mathbf{U}_B$ for Experiment 2. (b) The first 3 modes of $\mathbf{V}_B$ for Experiment 2.

Figure 5: Principal components for bursts of fire when evaluating the muzzle blast and shock wave. Note the obvious separation between M2 50 cal samples and the rest of the database on mode 3 of $\mathbf{V}_B$ corresponding to echo captured at the bottom of mode 3 of $\mathbf{U}_B$.

The results for all models evaluated between 0-1k Hz and 0-4k Hz with either 4 or 20 modes $\mathbf{V}_B$ are shown in Table 3. There is an obvious advantage to including the shock wave information, which often induces as much as a 5% increase in prediction accuracy. Interestingly, the LDA and Naive Bayes models suffer substantially (25% less accurate for multiple classes) when reducing the number of features to 4, but CART's accuracy only suffers marginally (no more than 9% less accurate).

Histograms comparing the best results at 20 features versus 4 features are shown in Figure 6.



(a) The LDA classification results with 20 modes $\mathbf{V}_B$. (b) The CART classification results with 4 modes $\mathbf{V}_B$.

Figure 6: Classification results after 1000 iterations for 0-4k Hz for Experiment 2.

4.3 Experiment 3: Single Shots and Bursts Combined

Unsurprisingly, the principal components of Experiment 3 are dominated by the burst data. The first 12 modes of $\mathbf{U}_C$ are shown in Figure 8a, and all but mode 2 characterize at least 2 shots. Figure 8b, which contains the first 3 modes of $\mathbf{V}_C$, also looks very similar to Figure 5b from Experiment 2.

Model	Hz Range	Modes $\mathbf{V}_B$	AK-47	FN FAL	M4	M2
LDA	1k Hz	4	47-55%	53-56%	61-65%	92-95%
LDA	1k Hz	20	83-86%	74-76%	76-80%	95-96%
LDA	4k Hz	4	56-58%	56-60%	72-74%	97-99%
LDA	4k Hz	20	87-90%	70-73%	83-86%	98-99%
NB	1k Hz	4	73-78%	17-19%	60-63%	89-91%
NB	1k Hz	20	86-88%	62-65%	82-85%	92-93%
NB	4k Hz	4	67-72%	20-23%	73-77%	92-95%
NB	4k Hz	20	91-93%	48-51%	85-88%	91-93%
CART	1k Hz	4	68-73%	45-48%	64-68%	93-95%
CART	1k Hz	20	82-85%	52-57%	75-77%	94-96%
CART	4k Hz	4	74-76%	43-47%	72-75%	95-97%
CART	4k Hz	20	81-85%	59-63%	75-80%	95-97%

Table 3: Cross-validation accuracy scores for each model during Experiment 2. The accuracy ranges include scores from 5 sets of 100 iterations for each model.

Figure 7 shows the singular values for Experiment 3. Just like $\mathbf{\Sigma}_B$, the diagonal components of $\mathbf{\Sigma}_C$ approach zero after the 552^{nd} value. The first 4 σ values account for 11.2% of $\mathbf{\Sigma}_C$ for the 0-1k Hz data and 11.6% of $\mathbf{\Sigma}_C$ for the 0-4k Hz data. Similarly, the first 27 σ values account for 25.7% of $\mathbf{\Sigma}_C$ for the 0-1k Hz trial, and 27.4% of $\mathbf{\Sigma}_C$ for the 0-4k Hz trial.

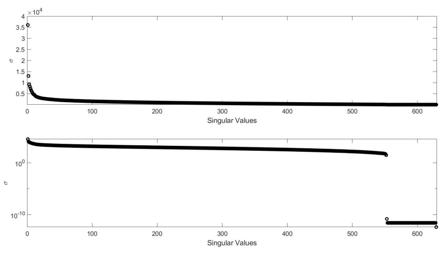
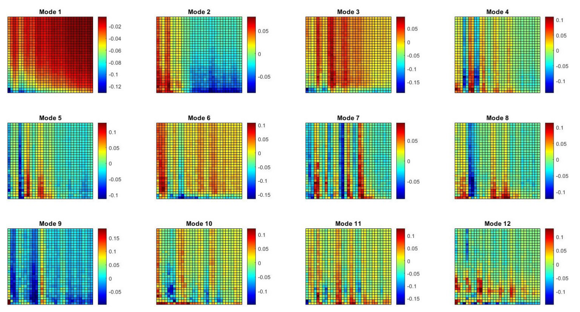
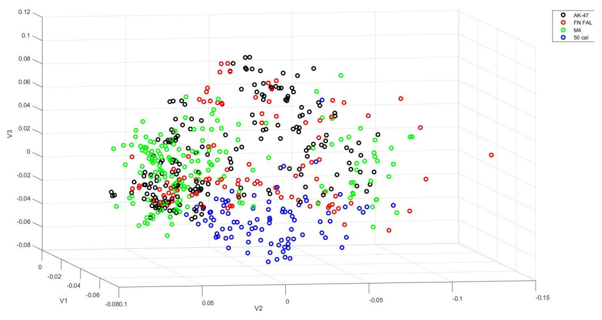


Figure 7: $\mathbf{\Sigma}_C$ values for Experiment 3 when evaluating the 0-4k Hz frequency range.



(a) The first 12 modes of $\mathbf{U}_C$ for Experiment 3.



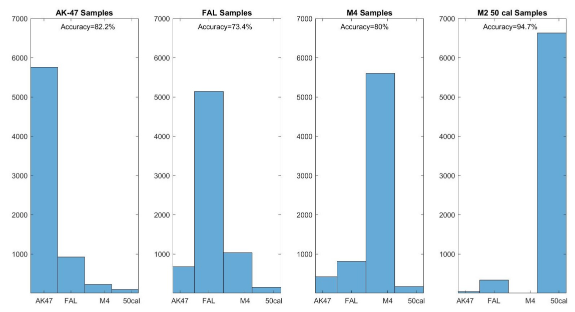
(b) The first 3 modes of $\mathbf{V}_C$ for Experiment 3.

Figure 8: Principal components for all shot events when evaluating the muzzle blast and shock wave.

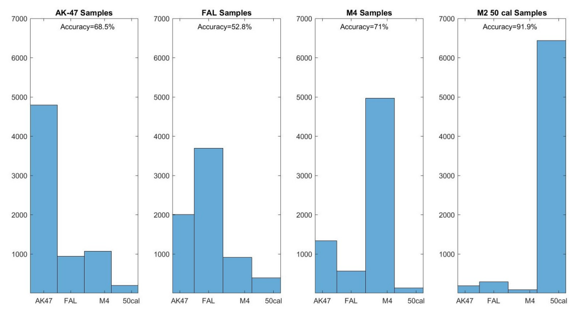
Much like Experiments 1 and 2, LDA for 0-4k Hz and 27 modes $\mathbf{V}_C$ proved to be the most accurate model, and CART for 0-4k Hz best preserved accuracy with a smaller number of features. The results for both models are shown in Figure 9, and Table 4 shows the results for the remaining models tested. Unlike Experiment 1, the variance of accuracy scores was much tighter. This is most likely due to the larger size of the combined data. However, Experiment 3 was very similar to Experiment 1 in that the difference of accuracy between the muzzle blast only data and the full frequency range data was minimal. In multiple cases, the accuracy ranges even overlap.

Model	Hz Range	Modes $\mathbf{V}_C$	AK-47	FN FAL	M4	M2
LDA	1k Hz	4	34-40%	34-36%	58-63%	85-89%
LDA	1k Hz	27	79-83%	77-80%	78-82%	90-92%
LDA	4k Hz	4	54-59%	26-32%	58-62%	90-93%
LDA	4k Hz	27	81-84%	73-75%	78-81%	94-95%
NB	1k Hz	4	62-66%	39-44%	44-50%	81-85%
NB	1k Hz	27	77-82%	73-75%	74-77%	90-93%
NB	4k Hz	4	62-67%	28-30%	59-62%	89-92%
NB	4k Hz	27	82-84%	60-61%	77-80%	91-92%
CART	1k Hz	4	72-76%	53-54%	65-70%	87-90%
CART	1k Hz	27	84-87%	67-69%	66-72%	93-95%
CART	4k Hz	4	66-70%	51-57%	68-73%	91-93%
CART	4k Hz	27	80-85%	62-65%	76-80%	94-95%

Table 4: Cross-validation accuracy scores for each model during Experiment 3. The accuracy ranges include scores from 5 sets of 100 iterations for each model.



(a) The LDA classification results with 27 modes $\mathbf{V}_C$.



(b) The CART classification results with 4 modes $\mathbf{V}_C$.

Figure 9: Classification results after 1000 iterations for 0-4k Hz for Experiment 3.

5 Summary and Conclusions

In this paper we apply several classification techniques to identify common weapon systems by their acoustic signature. We demonstrate that incorporating the shock wave component of a gunshot event in classification models does improve prediction accuracy, but only marginally if the training set is large enough. This supports the possibility of using stationary audio surveillance to identify weapons because it is no longer essential to have the detector in front of the weapon when its fired. We also demonstrate that it is possible to identify specific weapons within a broader class such as assault rifle or machine gun, regardless of caliber. This result is important for military applications because it allows users to identify adversary capabilities without exposing personnel to harm. These results could be further tested by using recordings from military actions to demonstrate their usefulness in a less ideal environment.

5.1 Acknowledgements

This paper would not be possible without the assistance of Post 2.0 and their head of mixing, Ian Chase. Post 2.0 provided all of the data used in this paper. The views expressed herein are those of the authors and do not purport to reflect the position of the Department of the Army, or Department of Defense.

References

[1] Leo Breiman. *Classification and regression trees.* Routledge, 2017.

[2] Chloé Clavel, Thibaut Ehrette, and Gaël Richard. "Events detection for an audio-based surveillance system". In: *2005 IEEE International Conference on Multimedia and Expo*. IEEE. 2005, pp. 1306–1309.

[3] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[4] Robert C Maher. "Modeling and signal processing of acoustic gunshot recordings". In: *2006 IEEE 12th Digital Signal Processing Workshop & 4th IEEE Signal Processing Education Workshop*. IEEE. 2006, pp. 257–261.

[5] *MATLAB Documentation*. URL: `https://www.mathworks.com/help/symbolic/singular-value-decomposition.html`.

[6] J Millet and B Baligand. *Latest achievements in gunfire detection systems*. Tech. rep. 01DB-Metravib Limonest (France), 2006.

Appendix A MATLAB Functions

What follows is a list of MATLAB functions as they appear in Appendix B. Much of the text below with additional examples can be found in the Mathworks Documentation Library.

- `[cA,cH,cV,cD]=dwt2(X,'waveletname')`computes the single-level 2-D discrete wavelet transform (DWT) of the input data `X` using the `waveletname` wavelet. dwt2 returns the approximation coefficients matrix `cA` and detail coefficients matrices `cH`, `cV`, and `cD` (horizontal, vertical, and diagonal, respectively).
- `wcodemat(cH,n)` rescales an input matrix `cH` to a specified range `n` for display
- `randperm(n)` returns a row vector containing a random permutation of the integers from 1 to n without repeating elements.
- `[predictions]=classify(test,train,labels)` classifies each row of the data in `test` into one of the groups in `training`.The output `predictions` indicates the group to which each row of `test` has been assigned, and is of the same type as `labels`.
- `fitcnb(train,labels)` returns a multiclass naive Bayes model (`Mdl`), trained by the predictors in `train` and class labels.
- `predict(Mdl, test)` returns a vector of predicted class labels for `test` based on the output `Mdl` from `fitcnb` or `fitctree`
- `fitctree(train,labels)` returns a fitted binary classification decision tree based on the input variables `train` and output `labels`
- `[u,s,v] = svd(x)`[5] To compute singular values and singular vectors of a matrix, use three output arguments.

Appendix B MATLAB Code

This appendix includes some of the MATLAB code used to produce the results in this report. The full codes can also be found at `https://github.com/gchase15/WPN-Audio-Signatures`

```matlab
function dcData = dc_wavelet(dcfile)
[m,n]=size(dcfile); % 4096 x 80
nw=32*32; % wavelet resolution

nbcol = size(colormap(gray),1);
for i=1:n
  X=double(reshape(dcfile(:,i),64, 64));
  [cA,cH,cV,cD]=dwt2(X,'haar');
  cod_cH1 = wcodemat(cH,nbcol);
  cod_cV1 = wcodemat(cV,nbcol);
  cod_edge=cod_cH1+cod_cV1;
  dcData(:,i)=reshape(cod_edge,nw,1);
end
```

Listing 1: MATLAB code covering edge detection with the Haar wavelet.

```matlab
 %% Reduce White Space on JPEGs

 mainpath = 'C:\Users\gavin\Documents\MATLAB\ProjectData\Bursts\M4\4k spectrograms';
D=dir(fullfile(mainpath,'*.jpg'));
%     figure(8)
    for j=1:numel(D)
        thisfile = fullfile(mainpath,D(j).name);
        SpectroImg = imread(fullfile(mainpath,D(j).name));
        SpectroGray = double(rgb2gray(SpectroImg));
        SpectroCrop = SpectroGray(75:566,121:785);
%           imshow(uint8(SpectroCrop))
%           pause(.2)
        newFileName = split(thisfile,".");
        filename = string(newFileName(1,1));
        newFileNameChar = filename + '_cropped.jpg';
        imwrite(uint8(SpectroCrop),newFileNameChar);
    end
```

Listing 2: MATLAB code covering the task of removing excess information from the spectrogram images and converting them to grayscale.

```matlab
clear all; close all; clc;
% load 'Gavin_WeaponSounds/AK47/*.wav';

files = dir('Gavin_WeaponSounds/FN FAL/*.wav') ;


minWhitespace = 5;
N = length(files);
for i = 1:N
    test = [];
    consecutiveWhitespace = [];
    thisfile = files(i).name ;
    [data,fs] = audioread(thisfile);
%     sound(data, fs);
    datalength = length(data);
    for j = 1:datalength
        if (j > datalength)
            break;
        end
        if (data(j,1) ~= 0)
            consecutiveWhitespace = [];
            continue;
        else
            consecutiveWhitespace = [consecutiveWhitespace data(j,1)];
        end
        whitespacesize = size(consecutiveWhitespace,2);
        if (whitespacesize == minWhitespace)
            for k=1:whitespacesize
                data(j+1-k,:) = [];
                if (j+1-k > length(data(:, 1)))
                    break;
                end
                vtest = [test data(j+1-k,1)];
                datalength = datalength - 1;
            end
            consecutiveWhitespace = [];
        end
    end
    newFileName = split(thisfile,".");
    filename = string(newFileName(1,1));
    newFileNameChar = filename + 'Cleaned.wav';
    audiowrite(newFileNameChar, data, fs);

end
```

Listing 3: MATLAB code covering the removal of white space within audio recordings.

```
N = length(files);
for i = 1:N
    remove = [];
    consecutiveWhitespace = [];
    thisfile = files(i).name ;
    [data,fs] = audioread(thisfile);
    [rows, columns] = size(data);
    if (columns == 1)
        continue;
    end
    datalength = length(data);
    signalCounter = 0;
    inSignal = false;
    dataForNewFile = [];
    signalThreshold = 0.07;
    for j = 1:datalength
        if (j == datalength)
            break;
        end
        testVariable = data(j,1);
        if (abs(data(j,1)) < signalThreshold && abs(data(j,2)) < signalThreshold && inSignal == true)
            consecutiveWhitespace = [consecutiveWhitespace data(j,1)];
            cwsize = size(consecutiveWhitespace,2);
            if (cwsize > 36000)
                newFileName = erase(thisfile,".wav");
                newFileNameChar = append(newFileName,"_", int2str(signalCounter), ".wav");
                audiowrite(newFileNameChar, dataForNewFile, fs);
                dataForNewFile = [];
                inSignal = false;
            else
               saveDataPoint = data(j,:);%either one or two columns
               dataForNewFile = [dataForNewFile; saveDataPoint];
               inSignal = true;
               continue;
            end
        elseif (abs(data(j,1)) < signalThreshold && inSignal == false)
            consecutiveWhitespace = [consecutiveWhitespace data(j,1)];
            inSignal = false;
        elseif ((abs(data(j,1)) > signalThreshold || abs(data(j,2)) > signalThreshold) && inSignal == fa
            signalCounter = signalCounter + 1;
            consecutiveWhitespace = [];
            saveDataPoint = data(j,:);%either one or two columns
            dataForNewFile = [dataForNewFile; saveDataPoint];
            inSignal = true;
            continue;
        elseif ((abs(data(j,1)) > signalThreshold || abs(data(j,2)) > signalThreshold) && inSignal == t
            consecutiveWhitespace = [];
            saveDataPoint = data(j,:);%either one or two columns
            dataForNewFile = [dataForNewFile; saveDataPoint];
            inSignal = true;
            continue;
        end
    end
end
```

Listing 4: MATLAB code to split audio files into individual signals.

```
mainpath = 'C:\Users\gavin\Documents\MATLAB\ProjectData\Bursts\M4';
D=dir(fullfile(mainpath,'*.wav'));
dimensions = [1000 2000 4000]; numSeconds=.7;
 for r=1:3
    for j=1:numel(D)
         thisfile = fullfile(mainpath,D(j).name);
         [gunshotfile,fs] = audioread(fullfile(mainpath,D(j).name));
         shortgunshot = gunshotfile(1:(numSeconds*fs),:);
         aveshot = [];
         %This mini for loop averages the stereo sound to produce a mono sound.
         for k=1:size(shortgunshot,1)
             aveshot = [aveshot; mean(shortgunshot(k,:))];
         end
         aveshot = aveshot';
         nclip = length(aveshot);
         L = length(aveshot)/fs;
         tp2=linspace(0,L,nclip+1); tp=tp2(1:nclip);
         if rem(length(aveshot),2) == 0
             kp=(2*pi/L)*[0:nclip/2-1 -nclip/2:-1];
             kps=fftshift(kp);
         else
             kp=(2*pi/L)*[0:nclip/2 -nclip/2:-1];
             kps=fftshift(kp);
         end
     width=414;
    step=zeros(1,length(tp));
    mask=ones(1,2*width+1);
    Shotstept_spec=[];
    tslide=(width+1):300:(length(step)-width);
    for z=1:length(tslide)
         step=zeros(1,length(tp));
         step(tslide(z)-width:1:tslide(z)+width)=mask;
         Shotstep=step.*aveshot;
         Shotstept=fft(Shotstep);
         Shotstept_spec=[Shotstept_spec;
         abs(fftshift(Shotstept))];
    end
scales = [1 2 4];
tslidep=linspace(0,L,length(tslide));
f=figure('visible', false);
pcolor(tslidep,kps/(2*pi),Shotstept_spec.'),
shading interp, set(gca,'Ylim',[0 dimensions(r)],'Fontsize',[14])
ylabel('Frequency [Hz]'); xlabel('Time [sec]');
colormap(hsv)
newFileNameChar = 'M4Bursts' + string(scales(r)) + 'k' + string(j) + '.jpg';
print(newFileNameChar,'-djpeg');
close(f)
filescomplete = j
    end
    fileSetFinished = r
 end
```

Listing 5: MATLAB code covering the production of spectrograms for each audio sample.

```
%% Compiles elements of test data into individual matrices
clear all; close all; clc
mainpath = 'C:\Users\gavin\Documents\MATLAB\ProjectData\Combined Files\AK47\1k spectrograms';
    D=dir(fullfile(mainpath,'*cropped.jpg'));
    Band1Samp=[]; Band1AveSamp=[];
    for j=1:numel(D)
           song = imread(fullfile(mainpath,D(j).name));
         blurspectro = double(imresize(song, [64,64]));
%           imshow(blurspectro)
         avespectro = double(blurspectro) - mean(double(blurspectro));
         skinnyspectro =reshape(blurspectro,1,4096);
         skinnyavespectro = reshape(avespectro,1,4096);
         Band1AveSamp = [skinnyavespectro' Band1AveSamp];
         Band1Samp = [skinnyspectro' Band1Samp];
    end
mainpath = 'C:\Users\gavin\Documents\MATLAB\ProjectData\Combined Files\FAL\1k spectrograms';
    D=dir(fullfile(mainpath,'*cropped.jpg'));
    Band2Samp=[]; Band2AveSamp=[];
    for j=1:numel(D)
           song = imread(fullfile(mainpath,D(j).name));
         blurspectro = double(imresize(song, [64,64]));
         avespectro = double(blurspectro) - mean(double(blurspectro));
         skinnyspectro =reshape(blurspectro,1,4096);
         skinnyavespectro = reshape(avespectro,1,4096);
         Band2AveSamp = [skinnyavespectro' Band2AveSamp];
         Band2Samp = [skinnyspectro' Band2Samp];
    end
mainpath = 'C:\Users\gavin\Documents\MATLAB\ProjectData\Combined Files\M4\1k spectrograms';
    D=dir(fullfile(mainpath,'*cropped.jpg'));
    Band3Samp=[]; Band3AveSamp=[];
    for j=1:numel(D)
           song = imread(fullfile(mainpath,D(j).name));
         blurspectro = double(imresize(song, [64,64]));
         avespectro = double(blurspectro) - mean(double(blurspectro));
         skinnyspectro =reshape(blurspectro,1,4096);
         skinnyavespectro = reshape(avespectro,1,4096);
         Band3AveSamp = [skinnyavespectro' Band3AveSamp];
         Band3Samp = [skinnyspectro' Band3Samp];
    end
mainpath = 'C:\Users\gavin\Documents\MATLAB\ProjectData\Combined Files\50 cal\1k spectrograms';
        D=dir(fullfile(mainpath,'*cropped.jpg'));
    Band4Samp=[]; Band4AveSamp=[];
    for j=1:numel(D)
           song = imread(fullfile(mainpath,D(j).name));
         blurspectro = double(imresize(song, [64,64]));
         avespectro = double(blurspectro) - mean(double(blurspectro));
         skinnyspectro =reshape(blurspectro,1,4096);
         skinnyavespectro = reshape(avespectro,1,4096);
         Band4AveSamp = [skinnyavespectro' Band4AveSamp];
         Band4Samp = [skinnyspectro' Band4Samp];
    end
```

Listing 6: MATLAB code covering the task of compiling the spectrogram data into a single matrix.

```
 %% Wavelet Transform
    band1_wave = dc_wavelet(Band1AveSamp);
    band2_wave = dc_wavelet(Band2AveSamp);
    band3_wave = dc_wavelet(Band3AveSamp);
    band4_wave = dc_wavelet(Band4AveSamp);
    [U,S,V]=svd([band1_wave,band2_wave,band3_wave,band4_wave],0);
    %% U Analysis
    figure(1)
for j=1:12
  subplot(3,4,j)
  ut1 = reshape(U(:,j),32,32);
  ut2 = ut1(32:-1:1,:);
  pcolor(ut2), colormap(jet)
  set(gca,'Xtick',[],'Ytick',[])
  colorbar
    title('Mode ' + string(j))
end
%% Analyze S
figure(2)
subplot(2,1,1)
plot(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14])
axis([0 630 0 4.6*10^4])
yticks([5000 10000 15000 20000 25000 30000 35000 40000 45000])
xlabel('Singular Values')
ylabel('\sigma')
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14])
axis([0 630 0 4.8*10^4])
xlabel('Singular Values')
ylabel('\sigma')

sig=diag(S);
energy1=sig(1)/sum(sig)
energy2=sum(sig(1:27))/sum(sig)
eng10perc=sum(sig(1:63))/sum(sig)
eng25perc=sum(sig(1:158))/sum(sig)
eng50perc=sum(sig(1:315))/sum(sig)
eng=sum(sig(1:404))/sum(sig)
%% 3D version of V analyzation
figure(3)
plot3(V(1:190,1),V(1:190,2),V(1:190,3),'ko','Linewidth',[2])
hold on
plot3(V(191:294,1),V(191:294,2),V(191:294,3),'ro','Linewidth',[2])
hold on
plot3(V(295:465,1),V(295:465,2),V(295:465,3),'go','Linewidth',[2])
hold on
plot3(V(466:629,1),V(466:629,2),V(466:629,3),'bo','Linewidth',[2])
grid on, xlabel('V1'), ylabel('V2'), zlabel('V3')
legend('AK-47','FN FAL', 'M4', '50 cal')
```

Listing 7: MATLAB code covering the SVD analysis.

```
%% Classification Task
n=100

lda1 = []; lda2 = []; lda3 = []; lda4 = [];
nb1 = []; nb2 = []; nb3 = []; nb4 = [];
tr1 = []; tr2 = []; tr3 = []; tr4 = [];
for p=1:5
    band1prelda = []; band2prelda = []; band3prelda = []; band4prelda = [];
    band1prenb = []; band2prenb = []; band3prenb = []; band4prenb = [];
    band1pretree = []; band2pretree = []; band3pretree = []; band4pretree = [];
for k=1:n
    %pick a new randomization for each test.
    q1 = randperm(190); q2 = randperm(104); q3 = randperm(171); q4 = randperm(164);
    xband1 = V(1:190,1:27); xband2 = V(191:294,1:27);
    xband3 = V(295:465,1:27); xband4 = V(466:629,1:27);

    xtrain = [xband1(q1(1:183),:); xband2(q2(1:97),:);...
        xband3(q3(1:164),:); xband4(q4(1:157),:)];
    xtest = [xband1(q1(184:end),:); xband2(q2(98:end),:);...
        xband3(q3(165:end),:); xband4(q4(158:end),:)];
    ctrain = [ones(183,1); 2*ones(97,1); 3*ones(164,1); 4*ones(157,1)];

    %the classifiers (LDA,Naive Bayes, CART)
    [predictlda] = classify(xtest,xtrain,ctrain);
    nb = fitcnb(xtrain,ctrain); predictnb = nb.predict(xtest);
    tree=fitctree(xtrain,ctrain); predicttree = predict(tree,xtest);

% Collect the results
band1prelda = [band1prelda predictlda(1:7)]; band2prelda = [band2prelda predictlda(8:14)];
band3prelda = [band3prelda predictlda(15:21)]; band4prelda = [band4prelda predictlda(22:28)];
band1prenb = [band1prenb predictnb(1:7)]; band2prenb = [band2prenb predictnb(8:14)];
band3prenb = [band3prenb predictnb(15:21)]; band4prenb = [band4prenb predictnb(22:28)];
band1pretree = [band1pretree predicttree(1:7)]; band2pretree = [band2pretree predicttree(8:14)];
band3pretree = [band3pretree predicttree(15:21)]; band4pretree = [band4pretree predicttree(22:28)];

end
success1lda = sum(band1prelda(:) == 1)/(n*7); success2lda = sum(band2prelda(:) == 2)/(n*7);
success3lda = sum(band3prelda(:) == 3)/(n*7); success4lda = sum(band4prelda(:) == 4)/(n*7);
success1nb = sum(band1prenb(:) == 1)/(n*7); success2nb = sum(band2prenb(:) == 2)/(n*7);
success3nb = sum(band3prenb(:) == 3)/(n*7); success4nb = sum(band4prenb(:) == 4)/(n*7);
success1tree = sum(band1pretree(:) == 1)/(n*7); success2tree = sum(band2pretree(:) == 2)/(n*7);
success3tree = sum(band3pretree(:) == 3)/(n*7); success4tree = sum(band4pretree(:) == 4)/(n*7);

lda1 = [lda1 success1lda]; lda2 = [lda2 success2lda];
lda3 = [lda3 success3lda]; lda4 = [lda4 success4lda];
nb1 = [nb1 success1nb]; nb2 = [nb2 success2nb];
nb3 = [nb3 success3nb]; nb4 = [nb4 success4nb];
tr1 = [tr1 success1tree]; tr2 = [tr2 success2tree];
tr3 = [tr3 success3tree]; tr4 = [tr4 success4tree];
end
```

Listing 8: MATLAB code covering the classification tasks.