December 2016

Acoustic Detection, Source Separation, and Classification Algorithms for Unmanned Aerial Vehicles in Wildlife Monitoring and Poaching

Carlo Lopez-Tello
University of Nevada, Las Vegas

ACOUSTIC DETECTION, SOURCE SEPARATION, AND CLASSIFICATION ALGORITHMS FOR UNMANNED AERIAL VEHICLES IN WILDLIFE MONITORING AND POACHING

By

Carlo I. Lopez-Tello

Bachelor of Science in Engineering - Computer Engineering
University of Nevada, Las Vegas
2014

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Engineering – Electrical Engineering

Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
December 2016

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

November 17, 2016

This thesis prepared by

Carlo I. Lopez-Tello

entitled

Acoustic Detection, Source Separation, and Classification Algorithms for Unmanned Aerial Vehicles in Wildlife Monitoring and Poaching

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Engineering – Electrical Engineering
Department of Electrical and Computer Engineering

Venkatesan Muthukumar, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

Emma Regentova, Ph.D.
Examination Committee Member

Peter Stubberud, Ph.D.
Examination Committee Member

Sidkazem Taghva, Ph.D.
Graduate College Faculty Representative

Abstract

This work focuses on the problem of acoustic detection, source separation, and classification under noisy conditions. The goal of this work is to develop a system that is able to detect poachers and animals in the wild by using microphones mounted on unmanned aerial vehicles (UAVs). The classes of signals used to detect wildlife and poachers include: mammals, birds, vehicles and firearms. The noise signals under consideration include: colored noises, UAV propeller and wind noises.

The system consists of three sub-systems: source separation (SS), signal detection, and signal classification. Non-negative Matrix Factorization (NMF) is used for source separation, and random forest classifiers are used for detection and classification. The source separation algorithm performance was evaluated using Signal to Distortion Ratio (SDR) for multiple signal classes and noises. The detection and classification algorithms where evaluated for accuracy of detection and classification for multiple signal classes and noises. The performance of the sub-systems and system as a whole are presented and discussed.

Acknowledgements

I would like to thank the following people for their input on this work: Drs. Venkatesan Muthukumar, Emma Regentova, and Peter Stubberud. For their input in my graduate education, I would like to thank: Drs. Brendan Morris, Pushkin Kachroo, and Ebrahim Saberinia. I would also like to thank Sidkazem Taghva for being part of my Advisory Committee as the Graduate College Representative. Finally, I would like to thank the Department of Electrical and Computer Engineering for supporting me financially and logistically.

Table of Contents

List of Figures

Introduction

This work focuses on the problem of acoustic detection, source separation, and classification under noisy conditions. The goal of this work is to develop a system able to detect poachers and animals in the wild by using microphones mounted on unmanned aerial vehicles (UAVs). From the perspective of realizing this goal, this thesis deals with detecting man-made sounds, and animal sounds by using classification techniques in noisy conditions. However, there are two problems that must be addressed prior to classification. The first is signal or target detection in the presence of noise. A signal detector must first be implemented to make sure that the input to the classifier does not consist of only noise. The second problem that must be addressed is the separation of mixed signals into components. This is needed because the classifier expects single source signals not mixed signals. The noise conditions considered in this work are colored noise or *1/f* noise, wind noise, and propeller noise. The signals of interest considered in this work come from mammals, birds, vehicles, and firearms.

In the past, detection and tracking of humans and their artifacts has been conducted using video-based techniques [1]. However, video techniques require a direct line of sight, un-occluded targets, conditions with proper illumination, and adequate video resolution. Acoustic based detection techniques require none of the above. However, they are prone to interference from background noise and interference from other signals of interest that can occur simultaneously.

Background

Source Separation

Source separation refers to the process of splitting a mixed signal into signals containing its constituent components. For example, a signal containing two people talking would produce a signal containing the voice of each person. Source separation can be performed using Principal Component Analysis (PCA) [2] and Independent Component Analysis (ICA) [3], while multiple simultaneous recordings of the mixed signal are available. A distinct recording is needed for every possible source signal. When multiple recordings are not available, source separation techniques rely on having a dictionary of signals of interest. The most common technique for single channel source separation is Non-negative Matrix Factorization (NMF) [4].

NMF is a matrix factorization technique that tries to factor a matrix *V* into matrices *H* and *W*.

$$V = WH$$

NMF assumes that *V*, *H*, and *W* are all non-negative, that is all of their entries are greater than or equal to zero. The matrix *V* represents the magnitude of the Short Time Fourier Transform (MSTFT) [5] of a signal. The columns of *H* represent a set of NMF feature vectors. The rows of *H* represent a set of weights for each feature vector.

$$V = [w_1 \quad \cdots \quad w_n] \begin{bmatrix} h_1^T \\ \vdots \\ h_n^T \end{bmatrix}$$

The product of w_1 and h_1^T produce a matrix with the same dimensions as *V*. Thus, one can view *V* as the sum of simpler MSTFTs. The non-negative property is key, since it is assumed that the

MSTFTs of signals add up. A factorization that allowed entries of *H* to be negative would produce frequency components with negative amplitudes.

If two signals are added together then their MSTFTs will be:

$$V_3 = V_1 + V_2 = [w_{11} \quad \cdots \quad w_{n1}] \begin{bmatrix} h_{11}^T \\ \vdots \\ h_{n1}^T \end{bmatrix} + [w_{12} \quad \cdots \quad w_{n2}] \begin{bmatrix} h_{12}^T \\ \vdots \\ h_{n2}^T \end{bmatrix}$$

Which is equivalent to:

$$V_3 = V_1 + V_2 = [w_{11} \quad \cdots \quad w_{n1} \quad w_{12} \quad \cdots \quad w_{n2}] \begin{bmatrix} h_{11}^T \\ \vdots \\ h_{n1}^T \\ h_{12}^T \\ \vdots \\ h_{n2}^T \end{bmatrix}$$

$$V_1 + V_2 = [W_1 \quad W_2] \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}$$

From these equations, one can see that the MSTFTs of each signal can be recovered by finding H_1 and H_2, if W_1 and W_2 are known. This is accomplished by building a dictionary of features for V_1 and V_2. Two dictionaries are created, W_{D1} from signals that sound like the source of V_1 and another W_{D2} from signals that sound like the source of V_2. Since the dictionaries contain similar sounds it is expected that they will contain linear combinations of the features found in W_1 and W_2. The features will not be in the same order as W_1 and W_2, and obviously the dictionaries will contain more features. V_3 is factored into *W* and *H*. *W* contains both dictionaries of features. The algorithm then tries to minimize a cost function, while updating *H* and keeping *W* constant. The final iteration of *H* is then used to reconstruct V_1 and V_2.

$$V_3 = WH = [W_{D1} \quad W_{D2}] \begin{bmatrix} H_{D1} \\ H_{D2} \end{bmatrix}$$

$$V_1 = W_{D1} H_{D1}$$

$$V_2 = W_{D2}H_{D2}$$

Any vectors in W_{D1} and W_{D2} not present in W_1 and W_2 are expected to produce row vectors in W_{D1} and W_{D2} with all entries as zero. Once the reconstructed MSTFT is found all that is left is producing an estimate of the phase of the Short Time Fourier Transform (PSTFT) to compute a reconstructed STFT. The single source signals are then computed using the Inverse Short Time Fourier Transform (ISTFT) [5].

There are three common performance metrics used to evaluate source separation. Signal to Interference Ratio (SIR) tries to measure the amount of sound remaining in a reconstruction from the other signals. Signal to Artifact Ratio (SAR) tries to measure the amount of sound introduced by the source separation procedure, i.e. sound not coming from other channels. Signal to Distortion Ratio (SDR) is a measure that combines both SIR and SAR to produce a well-rounded metric. It is computed using the following equation:

$$SDR = 10log_{10}\frac{\sum_1^N\sum_1^M\|s^{true} + e^{spat}\|^2}{\sum_1^N\sum_1^M\|e^{interf} + e^{artif}\|^2}$$

Were s^{true} is a matrix whose rows are made up by the true signals that the separation is trying to reconstruct. e^{spat} is a matrix containing the filtering distortion component. e^{interf} is a matrix containing the interference component. e^{artif} is a matrix containing the artifacts component [6].

Signal Detection

Signal detection refers to the problem of determining the presence of a signal. It is an old problem that has roots in radar. The classical radar problem involves detecting a signal reflected from an aircraft. In essence, there is a receiver constantly scanning and recording noise, but once in a while the receiver will receive the reflected signal with noise added to it. One cannot

simply check if the received signal is identical to the transmitted one, because that will never occur due to the noise in the environment.

The theory of signal detection also covers cases with multiple hypotheses. One might wonder what the difference between the classical signal detection and modern classification techniques, since they seem to try to solve the same problem. The difference comes from the approach taken to solve the problem. Classical detection techniques are usually applied when the signals of interest follow simple models. This requirement comes from the need to perform hypothesis tests [7]. The construction of these tests requires that the joint probability distribution of the signals for each hypothesis be known. Classification techniques take a different approach. Instead of trying to model the signals a dataset of features extracted from the signals is created. This dataset is then used to divide the feature space with the hope that new signals that are not part of the dataset will be correctly classified.

Classification

Classification is the process of labeling new unmarked data points based on prior information obtained from labeled data points. Each data point is composed of a set of numbers, called a feature vector. There are various methods for classification [8]. Some consider the labels of points near a new point. Others try to find a hyper-plane that separates the labeled points with different labels, and some try to split the space into other shapes. In essence, all classifiers try to find a way of partitioning the vector space R^m, where m is the size of the feature vectors.

The process of classification consists of the following steps: 1) Collect data relevant to the problem, 2) Extract features from data relevant to problem domain, 3) Training and validation of the classifier, and 4) Testing of the classifier. Relevant features can be found

through domain insight or from examining the literature. The extracted features are then used to create training, validation, and test data for the classifiers.

The training set is the data the classifier uses to learn how to partition the vector space. Usually the features of this dataset are normalized, so that all features have the same mean and variance. There are other normalizations that can be used depending on the problem. Normalization is used to produce features of the same scale. Otherwise, features with large values relative to other features could dominate the partitioning. When unlabeled feature vectors are classified they are normalized using the mean and variance of the training set.

The validation set is used to tune hyper-parameters. Hyper-parameters are values that affect the partitioning of a specific classifier, for example the number of neighbors in a K Nearest Neighbors classifier [8]. The validation set is used to compare the performance across different values of hyper-parameters. Then, the hyper-parameter values that result in better performance are used to create a final model. The accuracy of the final model is then evaluated on the test set. The performance on the test set (unseen or new data) is used to evaluate the performance of the classifier. A test set is used to avoid over fitting during hyper-parameter selection.

A decision tree is a model that can be used for both classification and regression [8]. When used for classification a decision tree splits the feature space into a number of hyper-rectangles, or intersections of half spaces. It does this by looking at every entry in the feature vector and comparing it to one or more thresholds. The decision tree partitions the feature space so that every hyper-rectangle contains mostly feature vectors from a single class, called growing. To classify an unlabeled feature vector, the tree finds which hyper-rectangle the feature vector belongs to and labels it according to the class of the majority of the training vectors in that hyper-rectangle.

A random forest [8] is a model that uses a collection of decision trees. When classifying a new instance each tree produces a vote for what class it thinks the new instance belongs to. The class with the majority of votes wins and the new instance is labeled accordingly. Each tree in the forest is trained with a dataset sampled from the training set. If there are *N* samples in the training set, then *N* samples are drawn with replacement. Out of the *M* features *K* are chosen at random and used to grow each tree. Each tree is grown until every hyper-rectangle contains a single class of vectors.

Cross validation is a way to perform validation without having a validation set [8]. The training set is used for both training and validation. The training set is split into sets called folds. The samples in each fold are chosen randomly and every fold contains the same number of samples. One fold is used as a validation set and the remaining folds are used as a training set. This procedure is repeated over all of the folds and then the results are averaged. The average score is then used to select from different models instead of the validation score.

Principal Component Analysis (PCA) is a way of reducing the dimension of feature vectors [9]. It tries to find the best projection of each feature vector in R^m onto a lower dimensional space R^k. PCA does not remove features. Instead, it creates new features composed of linear combinations of the original features. PCA is performed on a matrix *X* made up of rows of feature vectors in order to produce a matrix *Y* with less columns. *Y* is computed by projecting *X* onto a set of *K* singular vectors with the largest singular values of the covariance matrix of *X*. The idea is to project the feature vectors along axes that maximize the spread of the new features. This is accomplished by choosing the singular vectors with the highest singular values. The number of singular vector to use is a free parameter that depends on the application. In this work, *K* will be treated as a hyper-parameter that needs to be tuned using validation.

Methodology

A system was developed in order to detect the following classes of sounds: mammals, birds, firearms, and vehicles. These are the types of sounds related to wildlife and poaching. The goal of the system is to be able to monitor wildlife and detect poachers. The system consists of three parts: source separation, signal detection, and signal classification. Each part is meant to address a different kind of problem. Source separation is used to split mixed sounds into their constituent components. Detection is used to determine when a signal of interest is present in a recording. Finally, classification is used to determine which of the four classes is present in the recording. Two different models were compared for the classifier, a model trained with clean data and a model trained with noisy data.

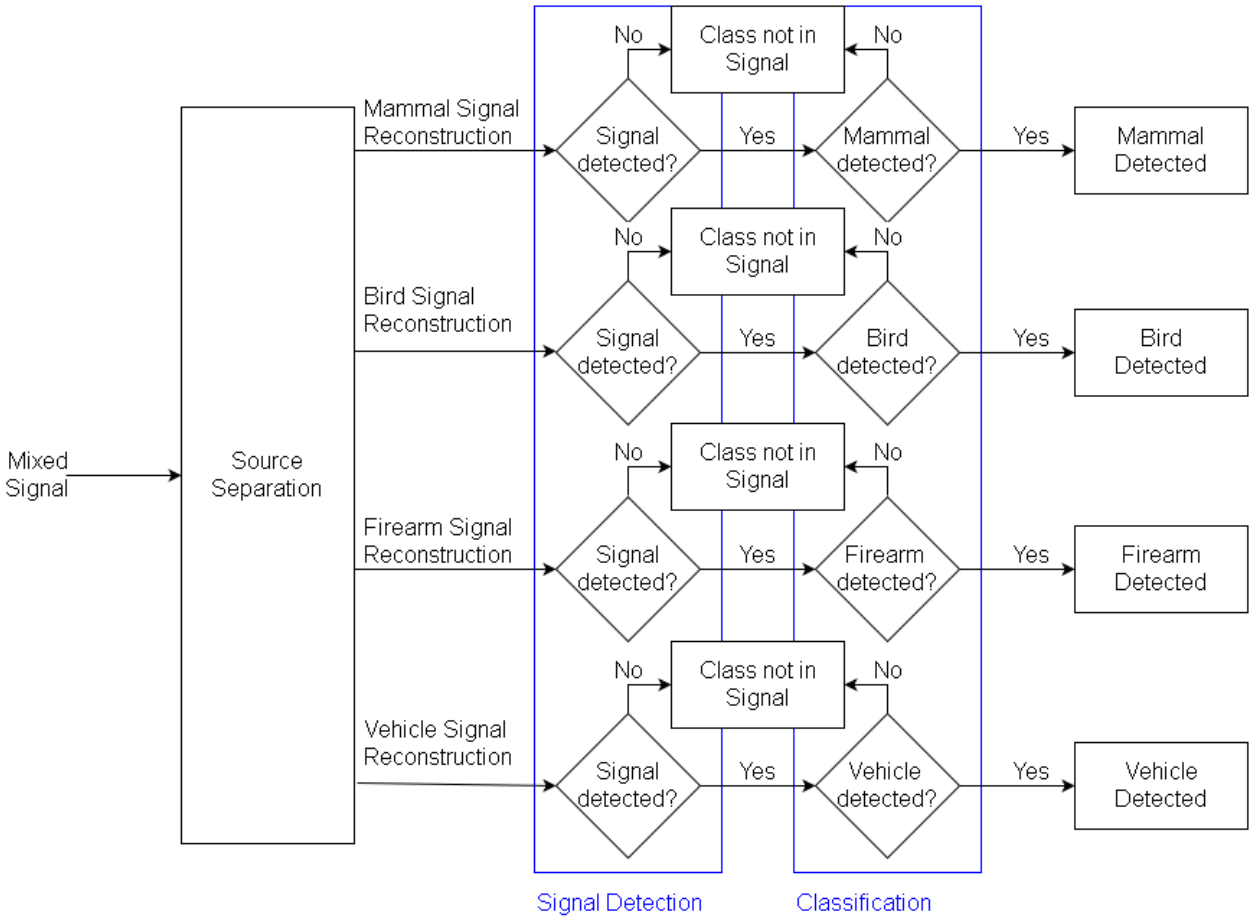

Figure 1: System diagram

Dataset

Numerous recordings were collected from the internet, primarily from findsounds.com [10]. Initially 100 recordings for each class were acquired. Due to the varying nature of the sampling frequency across recordings each recording was downsampled to 8 kHz, the minimum sampling frequency found among the recordings. These 100 recordings were then split into 1 second segments, in order to have a uniform length for the purpose of adding signals. Segments with a length less than 1 second were zero padded. Any 1 second recording without any audible sound was then removed. In total the number of 1 second recordings gathered was 1630. The number of recordings for each class was: 328 mammal recordings, 444 bird recordings, 259 firearm recordings, and 599 vehicle recordings.

The types of mammal recordings consisted of big mammals. Some examples are: chimpanzees, bears, and tigers. The types of bird recordings were mostly singing and chirping with a few howls from owls. The firearm class consisted of both automatic and semiautomatic fire from handguns and rifles. Finally, the vehicle class consisted of engine sounds from motorcycles, cars, and trucks.

These 1630 recordings were used to create three different datasets. Each dataset was used to train a different part of the system. The 1630 clean recordings were used to develop a clean classification model. They were also used to create the dictionaries for source separation. A second dataset was created by adding noise to the clean recordings. Different types of noise were added whose power was selected in order to achieve specific Signal to Noise Ratios. For example, white noise was added to the same 1 second recording in order to produce noisy recordings with the following Signal to Noise Ratios (SNRs): -20 dB, -10 dB, -15 dB, -10 dB, -5 dB, 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB. The following types of synthetic noise were added: white noise, red noise, pink noise, violet noise, and blue noise. Ideally, one would want to use noise recorded in the environment that the system is deployed in, or a big dataset of wind, rain, and propeller noise. Due to a lack these things, easy to generate types of noise with a wide range of power spectral densities were used. The second dataset consisted of 73350 recordings. Finally, a third dataset was created to train the detector. This dataset was made up of the 73350 noisy samples and another set of 73350 independently generated recordings of pure noise. The types of synthetic noise used were the same as the types added to create the noisy dataset.

Source Separation

Source Separation was done using Non-negative Matrix Factorization (NMF). The NMF implementation from NMFLib [11] was used to factor V. Due to performance issues, the Euclidean norm was used as the cost function [12]. Using the Kullback–Leibler (KL) divergence

or a regularized Euclidean norm proved to be too slow on the machine being used. In order to find the optimal frame size for the Short Time Fourier Transform (STFT) and the optimal number of NMF features 10-fold cross validation was used with 80 percent of the clean dataset. The performance of the source separation was evaluated with BSS Eval a MATLAB toolbox [6]. The frame size (256) and number of NMF features (10) that achieved the highest average Signal to Distortion Ratio (SDR) [13] were used. The performance of the source separation using the optimal parameters was then evaluated on the remaining 20 percent of the clean dataset.

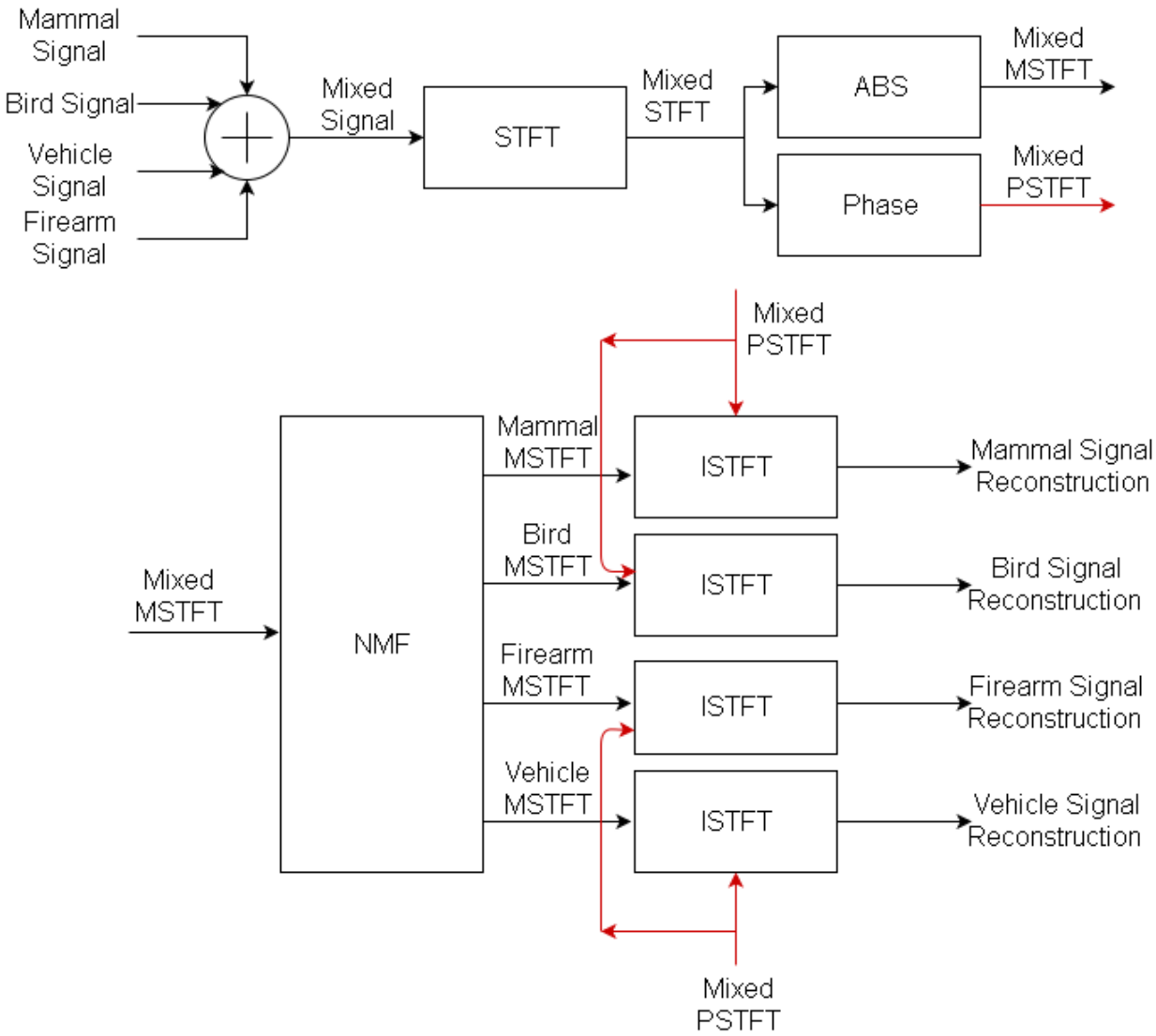

Figure 2: NMF subsystem diagram

The evaluation methodology consisted of creating a dictionary of features for each class. Each dictionary was then combined to create the matrix *W*, a block matrix where each block is a class dictionary. A mixed signal was then created by adding four randomly chosen signals not in the class dictionaries, one from each class. The power of each signal was normalized before adding them together. The magnitude of the Short Time Fourier Transform (MSTFT) of the mixed signal was then used as the matrix to deconstruct, *V*. While, at the same time the matrix *W* was kept constant for the NMF procedure. The decomposition matrix *H* was then used to reconstruct the MSTFT for each signal added. Each reconstructed MSTFT was created by multiplying a dictionary from each class with its corresponding block in *H*. The phase information from the mixed signal was then used as the phase information for the reconstructed STFT. Finally, each reconstructed signal and its original counterpart were evaluated with BSS Eval. The remaining 20 percent of the data was used to evaluate the system under noisy conditions.

Signal Detection and Signal Classification

One might wonder why signal detection and classification are needed when source separation produces four signals that are supposed to belong to only one class. Signal detection is required in case the mixed signal is not a mix of the four classes. In other words, the separation system will produce four reconstruction signals even if the mixed signal only contains two sources. Signal detection is required to address the case in which either the mixed signal is composed of pure noise or if any one of the reconstructed signals comes from noise in the mixed signal. Signal classification is needed in cases where there interference between classes in the reconstructed signals. For example, the mixed signal is composed of one source and all four of the reconstruction signals are distorted versions of the true source. Thus, a classifier is needed to check if the reconstructed signals actually belong to the class of signals used in the dictionary to reconstruct them.

Both signal detection and signal classification were performed using very similar methods. They were each achieved by training random forests. Random forests were picked over Support Vector Machines, because of faster training time [8]. The random forest implementation from Scikit-learn [14] was used. Each random forest was trained with a different dataset. The random forest used for detection was trained by a two class dataset. This dataset contained the noisy versions of the recordings as one class, and recordings made up of pure noise as the other class. Two different classifiers were trained for classification. One model was trained using the clean recordings, and the other was trained using the noisy recordings. The point of making two classification models was to compare them to each other. The same features were extracted for both detection and classification. These features were extracted, and then Principal Component Analysis (PCA) was used to decrease the dimensionality of the feature vectors. 10-fold cross validation on 80 percent of each dataset was used to find optimal parameters for each random forest (the number of PCA components, the number of trees, frame size, and number of features per tree. The optimal parameters for the detector were the following: 10, 500, 400, and 3 respectively. Both classification models used the same optimal parameters as each other: 30, 500, 400, and 5 respectively. The remaining 20 percent of data was used to evaluate each model under noisy conditions.

Feature Extraction

Due the time varying nature of acoustic signals, features were taken from overlapping frames/chunks of each signal. As a simple example, imagine using the variance of a signal as a feature. If the variance varies with respect to time, then the variance of each frame might be very different from the mean variance across frames. Each signal was split into overlapping frames. A feature vector was then extracted from each frame. The statistics of the feature vectors were calculated to produce one vector for each statistic. Finally, each statistic vector was combined to produce a single feature vector for each signal.

Feature extraction was accomplished by using Essentia, an audio analysis library [15]. Common low level descriptors used in audio were used. The features used from the library were the following (names appear as they do in the documentation):

1. Barkbands - spectral energy in 27 bands given by bark scale [16] [17]
2. Barkband Kurtosis - kurtosis of spectral energy in 27 bands given by bark scale [17]
3. Barkband Skewness - skewness of spectral energy in 27 bands given by bark scale [17]
4. Barkband Spread - variance of spectral energy in 27 bands given by bark scale [17]
5. HFC - high frequency content measure [18] [19]
6. MFCCs - Mel Frequency Cepstrum Coefficients [20] [17]
7. Pitch - fundamental frequency estimate [21]
8. Pitch Instantaneous Confidence - confidence with which the pitch was detected [0,1] [19]
9. Silence Rate 20dB - checks to see if power is higher than 20dB [19]
10. Silence Rate 40dB - checks to see if power is higher than 40dB [19]
11. Silence Rate 60dB - checks to see if power is higher than 60dB [19]
12. Spectral Complexity - spectral complexity is based on the number of peaks in the spectrum [19] [22]
13. Spectral Crest - ratio between max and arithmetic mean of spectrum [17] [19]
14. Spectral Decrease - finds slope using linear regression [17]
15. Spectral Energy - sum of spectrum samples squared [23]
16. Spectral Energy Band Low - computes energy in frequency band [15,200] [23]
17. Spectral Energy Band Middle Low - computes energy in frequency band [150,800] [23]
18. Spectral Energy Band Middle High - computes energy in frequency band [800,4000] [23]
19. Spectral Flatness dB - ratio between geometric mean and arithmetic mean converted to dB [17]
20. Spectral Flux - L2 norm of difference between consecutive frames [24] [25]

21. Spectral RMS - root mean square of spectrum [17]
22. Spectral Rolloff - frequency under 85% of the energy of the spectrum is located [17]
23. Spectral Strongpeak - ratio between the peak of the spectrum and the bandwidth around the peak. The bandwidth is cut off at half the amplitude of the max [26]
24. Zero Crossing Rate - number of zero crossings divided by the number of samples in the frame [17] [19]

These features were extracted from each frame. Then, the following statistics were computed from the frames belonging to the same signal: min, max, median, mean, variance, skewness, kurtosis, mean of the first difference, and variance of the first difference. These statistics were then combined to produce a feature vector with a length of 567. PCA was used to reduce the number of features, since the size of each of the datasets is small compared to the size of the feature space. The number of principal components was chosen with cross validation.

Complete System

The overall system operates in the following manner. A new one second signal is captured. The signal is then split into four reconstruction signals, one for each class. Each reconstruction signal is filtered with the detector. If the detector classifies a reconstructed signal as noise then that reconstructed signal is labeled as noise. Any reconstructed signal that is not labeled as noise is then fed to the classifier to confirm that the reconstruction matches the class that it is supposed to come from. Any reconstructed signal that doesn't match the class result from classification is also labeled as noise.

The system was tested on mixed signals containing random mixes of signals: pure noise, single class, two class mixes, three class mixes, and four class mixes. The signals in

each mix were chosen at random from the 20 percent of the clean dataset not used in training. The new mixes were used to evaluate the complete system under noisy conditions.

Results

Each part of the system was evaluated by using a hold out set containing 20 percent of the data for that part of the system. Various types of noise were added to the test data in order to see the effect noise has on the system. The following types of synthetic noise were used: white noise, red noise, pink noise, violet noise, and blue noise. Some natural noise samples were used to see the performance of the system under more realistic conditions. These were wind noise, propeller noise, and wind + propeller noise.

Source Separation

The performance of the source separation across various types of noise appears to be mostly independent from the amount of noise added to the mixed signal. The Signal to Distortion Ratio (SDR) after separation consistently increases around 3 dB. The SDR starts to decrease significantly after the Signal to Noise Ratio (SNR) decreases past the 10 dB mark [Figure 3].

The separation performance under different types of noise varies. At low SNR, red noise and wind noise have better results. Under the other types of noise the performance is much closer. This difference becomes less pronounced as the SNR increases [Figure 4].

The separation performance for each class is different. The reconstruction of bird signals shows the best performance. The mammal class is the second best. The firearm and vehicle classes have similar performance [Figure 5].

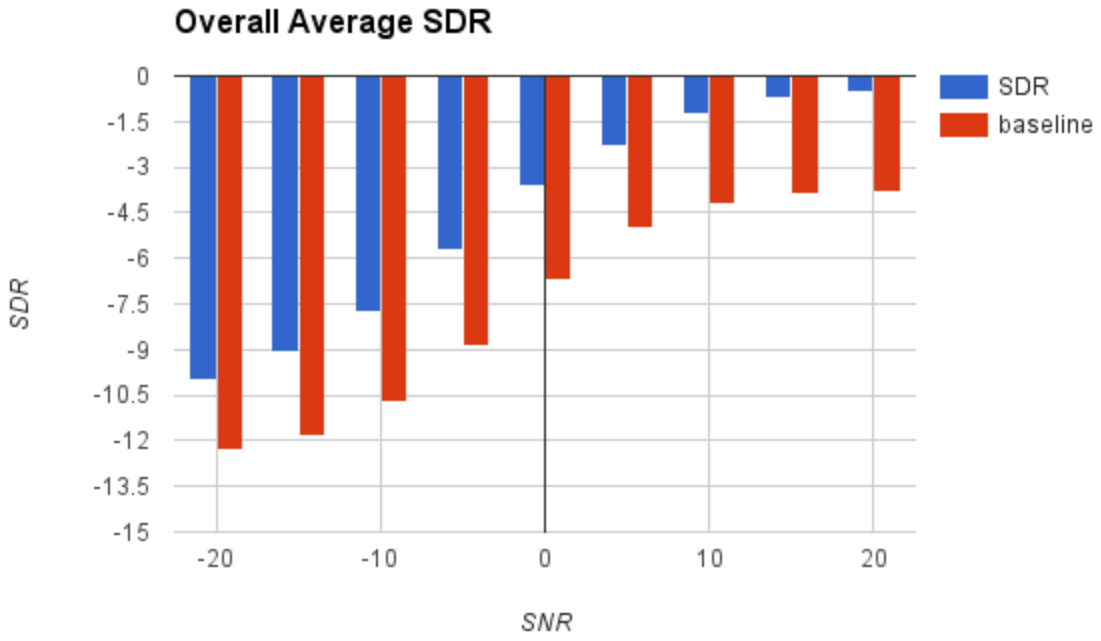

Figure 3: Average SDR across all noise types versus SNR

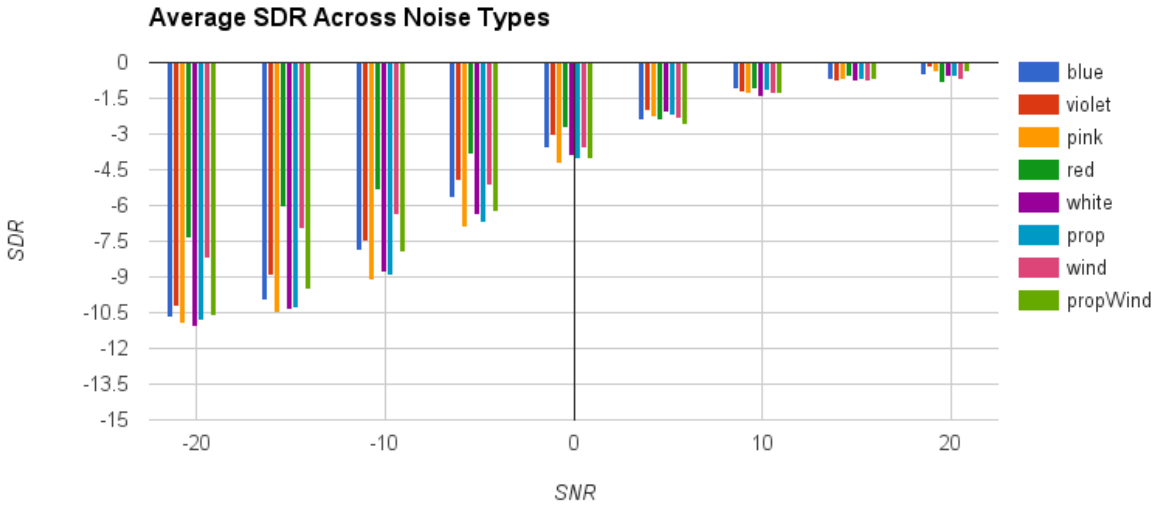

Figure 4: SDR of of each noise type versus SNR

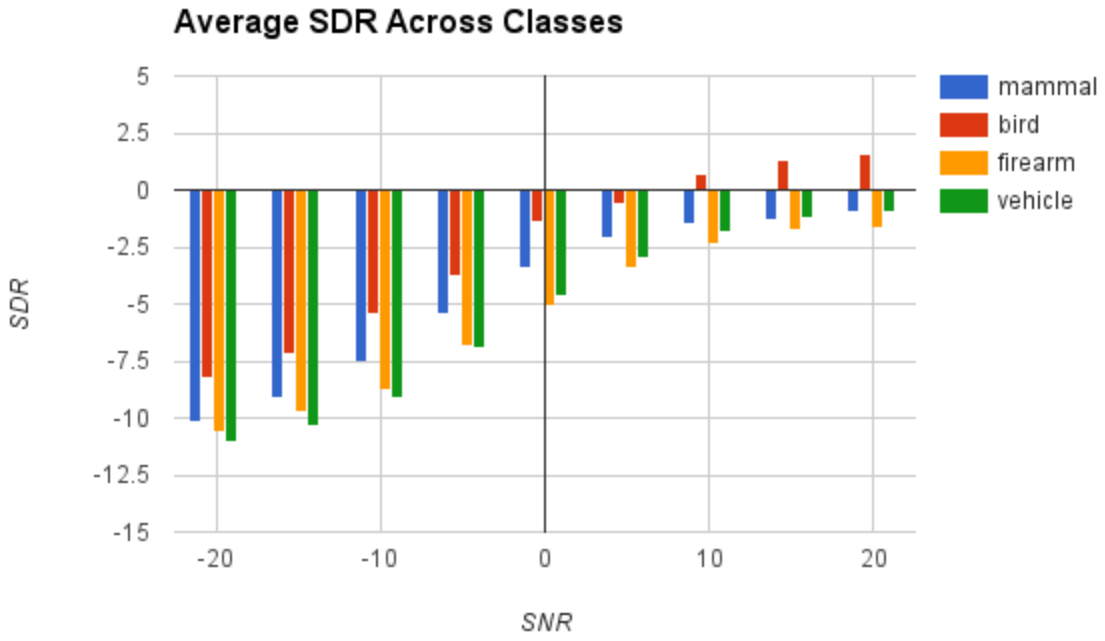

Figure 5: SDR for each class versus SNR

Detection

The accuracy of the detector seems to be robust to noise. The accuracy only starts to degrade once the SNR drops below -10 dB. Before reaching -10 dB, the accuracy is constant [Figure 6].

The accuracy under different types of noise shows that the performance of the detector largely depends on the noise used to corrupt the training set. The accuracy of the detector for signals corrupted by blue, pink, red, violet, and white noise is far superior. This was expected since the detector was trained with signals corrupted with these types of noise. A dataset of wind noise and propeller noise is probably needed to train the detector for good performance in realistic conditions. Once the SNR is -10 dB the performance under synthetic noise starts to

deteriorate. This suggests that even with a better dataset of noise the limit of performance is somewhere between -10 dB and -5 dB SNR [Figure 7].

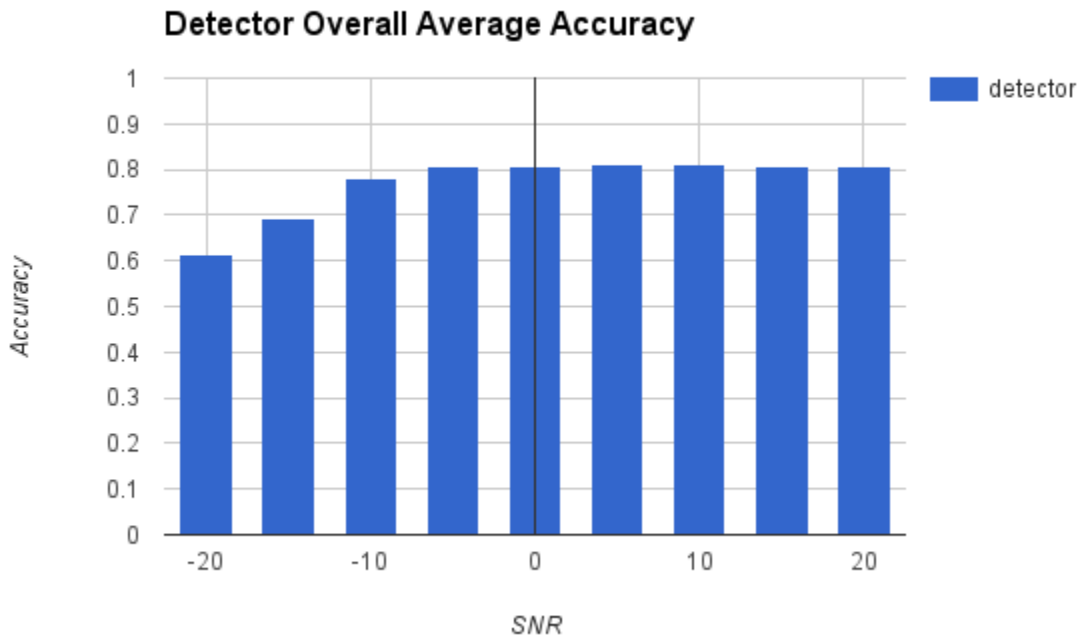

Figure 6: Average detector performance across all noise types versus SNR

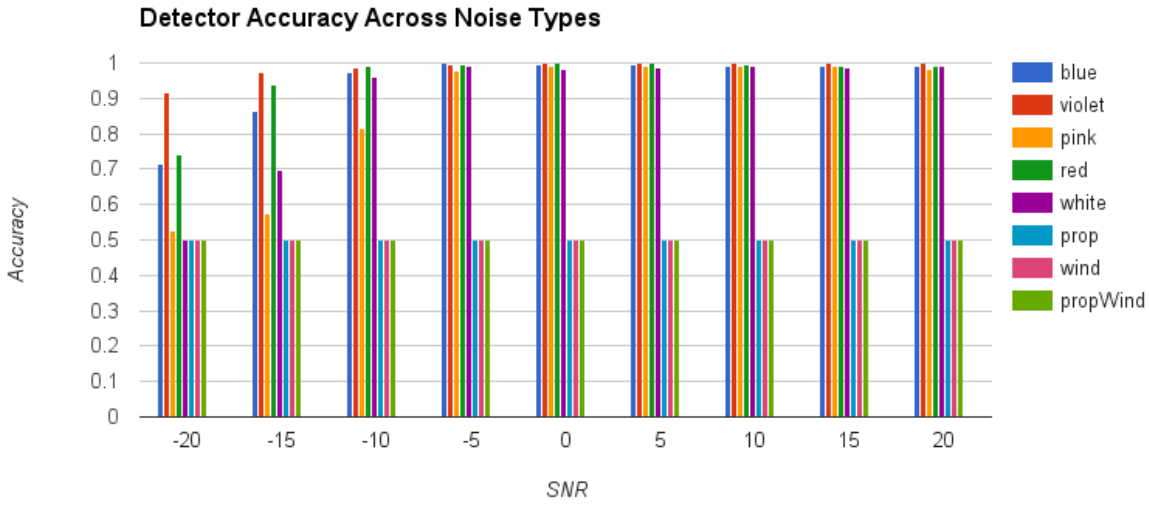

Figure 7: Detector accuracy for each noise type versus SNR

Classification

Out of the two different classification models, one trained with clean data and the other trained with noisy data, the model trained with noisy data produces a classifier that is more robust to noise. The performance of the clean model starts drops to 70 percent at an SNR of 10 dB and keeps dropping significantly as the SNR decreases. The noisy model hits the 70 percent mark at around 0 dB and always outperforms the clean model [Figure 8].

The accuracy of the clean model under different kinds of noise is similar. Two types of noise, blue noise and violet noise appear to be outliers. This is probably because the power spectral density increases with frequency for those two types of noise. Most sounds of interest have most of their energy in lower frequencies. Thus, distorting lower frequencies appears to have a more significant effect on the classification [Figure 9].

The noisy model is more robust to both synthetic noise and natural noise. This suggests that artificial noise can be used to increase the performance of the classifier under realistic conditions. The accuracy drops faster for natural noise than for the synthetic noise. This suggests that the synthetic noise should be expanded to include more types of noise. The accuracy under synthetic noise shows that good performance can be achieved at -5 dB SNR if the classifier was trained with a dataset of natural noise [Figure 10].

When using the clean classifier, the accuracy of most classes decreases as the SNR decreases, but accuracy of the vehicle class increases as the SNR decreases. The reason for this is that the other classes are being classified as vehicles as the SNR decreases. This could be caused by the fact that sounds produced by vehicles sound a lot like noise. The mammal class has the lowest accuracy. The variability of mammal sounds between species is a likely cause [Figure 11].

The trends of the noisy model are similar to the clean model, but the performance is much better. The accuracy of the vehicle class stays constant instead of increasing [Figure 12].

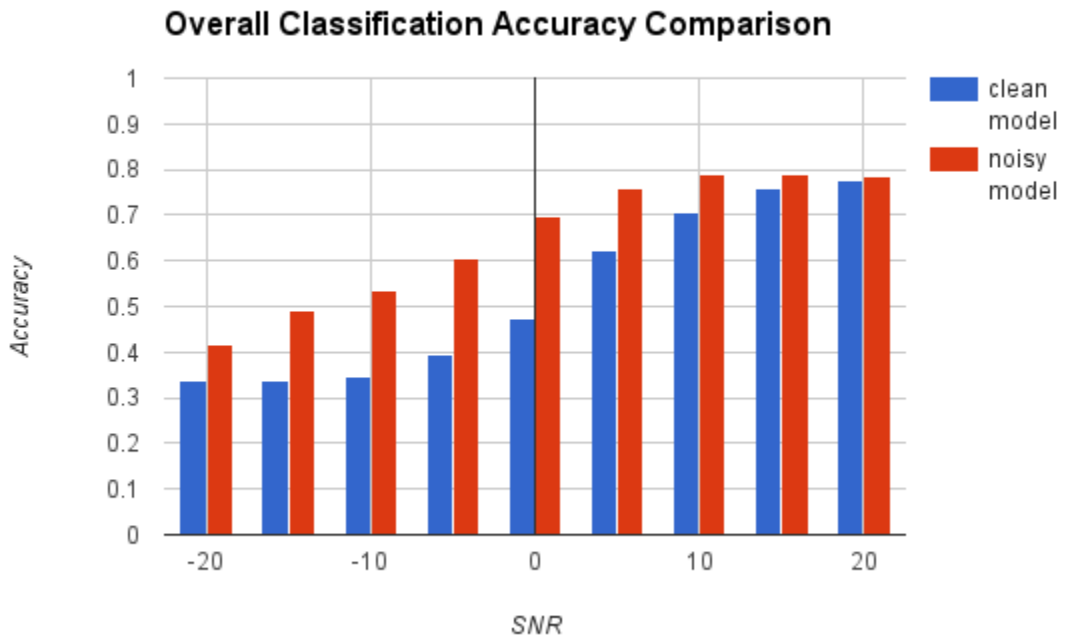

Figure 8: Comparison of the average accuracy of both classifier models across all noise types versus SNR

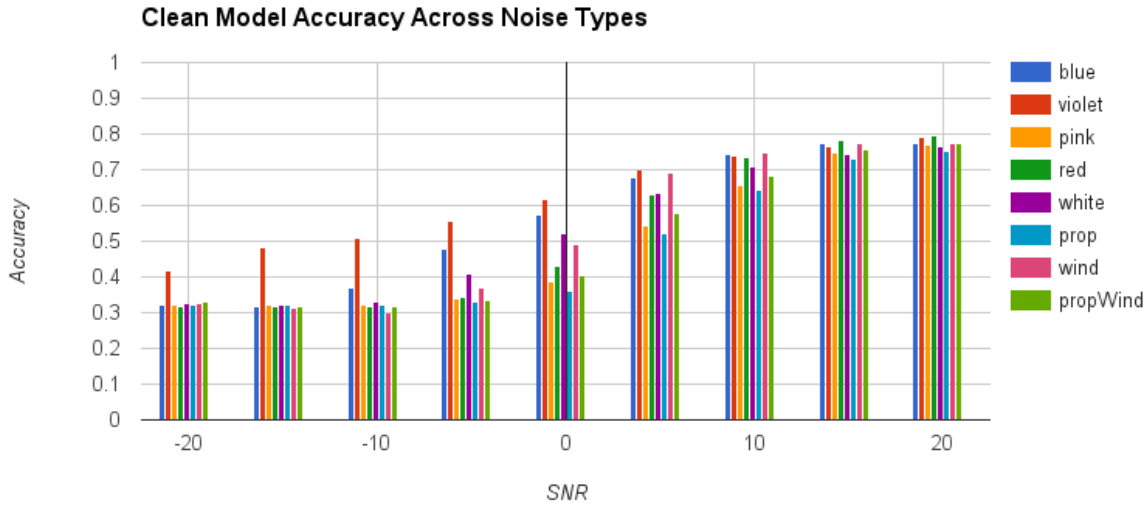

Figure 9: Clean classifier accuracy for each noise type versus SNR

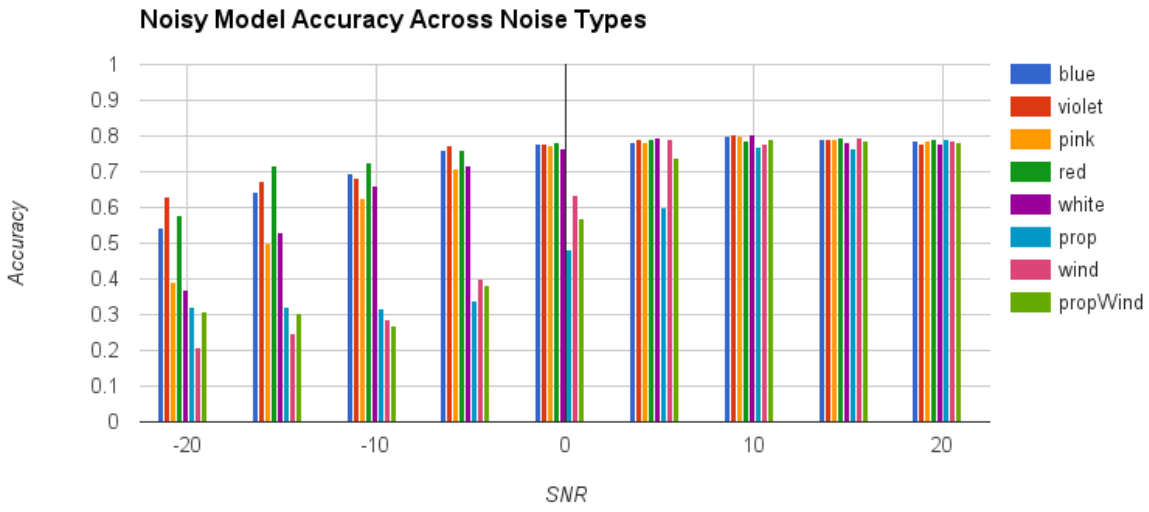

Figure 10: Noisy classifier accuracy for each noise type versus SNR

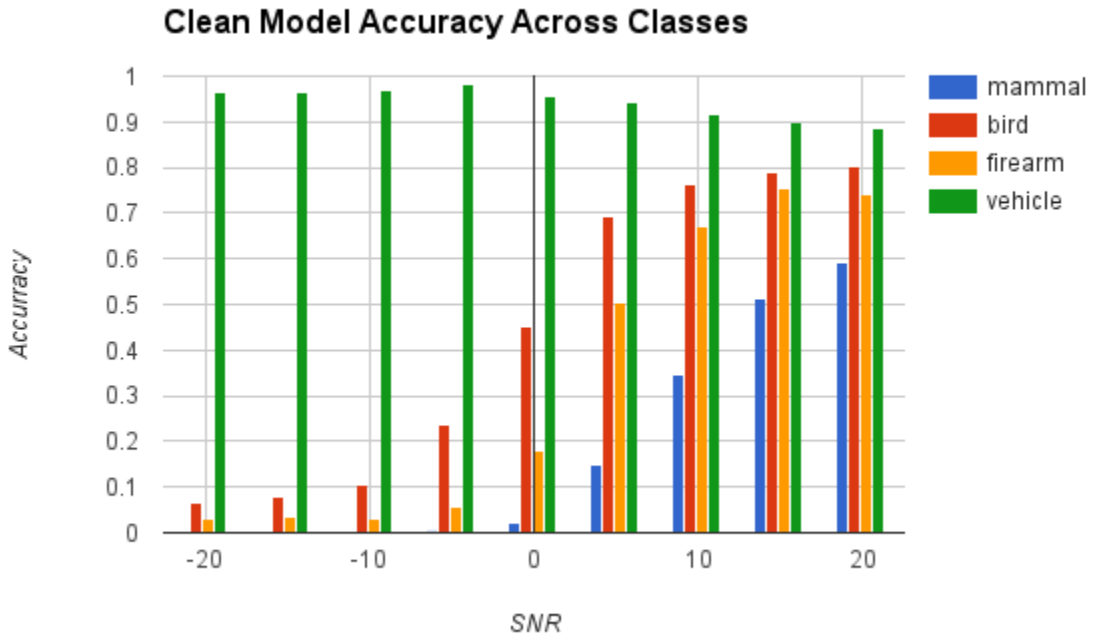

Figure 11: Clean classifier accuracy for each class versus SNR

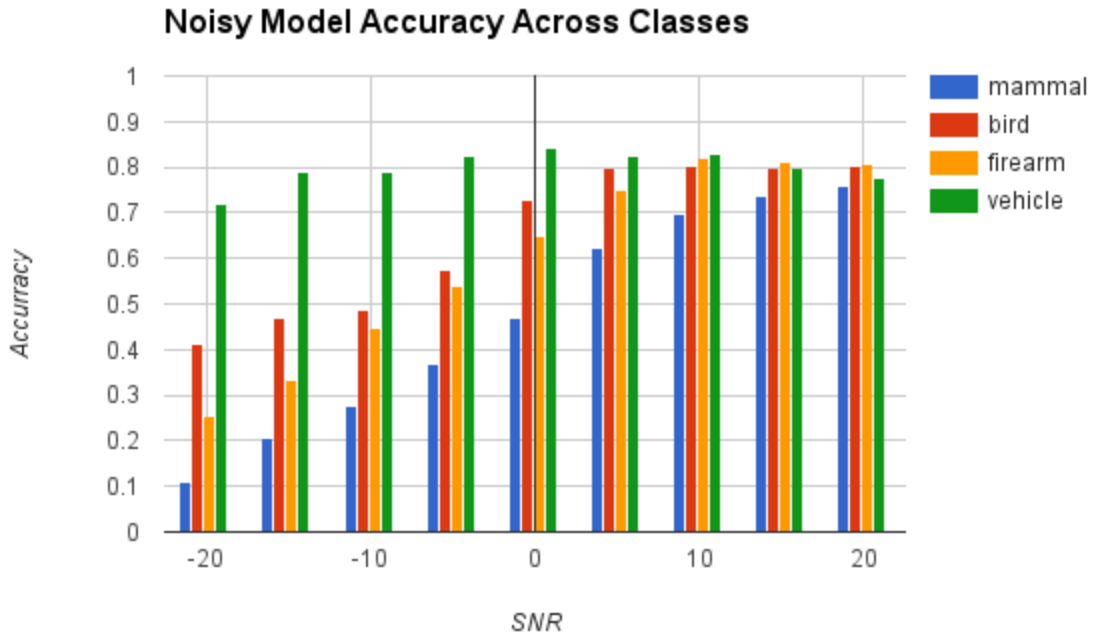

Figure 12: Noisy classifier accuracy for each class versus SNR

Complete System

Surprisingly, there does not appear to be much of a difference between the performance of the entire system using the clean model for classification or the entire system using the noisy model. The performance is poor for both systems. There is a slight drop in performance as the SNR decreases. One would expect a bigger drop, but the performance is so bad that there is little room for it to fall [Figure 13].

The performance of the system using the clean classifier model does not appear to depend on the noise type. More importantly, the accuracy at 20 dB SNR is much worse than the score of the clean classifier alone. This suggests that the source separation is not good enough, and is producing signals that with significant distortion coming from the other classes [Figure 14].

The performance of the system using the noisy classifier is very similar to the system using the clean classifier. The performance doesn't appear to be better across the synthetic types of noise, contrary to the results of the noisy classifier on its own. This gives more credence to the hypothesis that the performance of the source separation is at fault [Figure 15].

The system using the clean model for classification shows that at an SNR higher than 0 dB, the class accuracy of both mammals and vehicles starts improving. The bird and firearm classes appear to be negatively correlated to SNR. [Figure 16]

There is a slight improvement for the animal class when the system uses the noisy classifier [Figure 17].

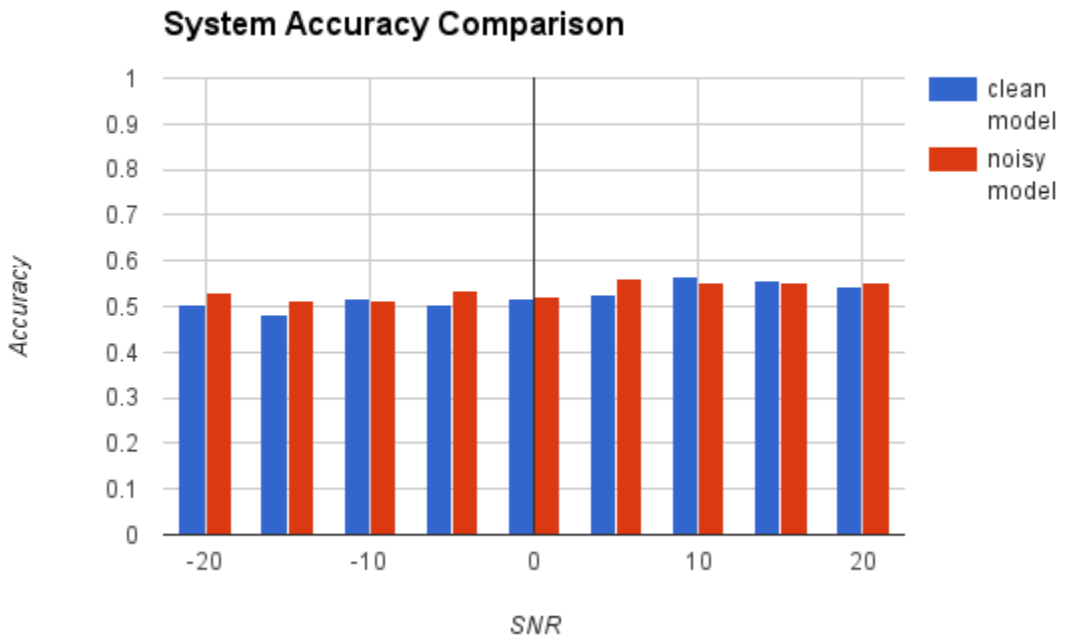

Figure 13: Comparison of the accuracy of the entire system using each classifier model across all noise types versus SNR

Figure 14: System accuracy of each noise type versus SNR, while using the clean classifier

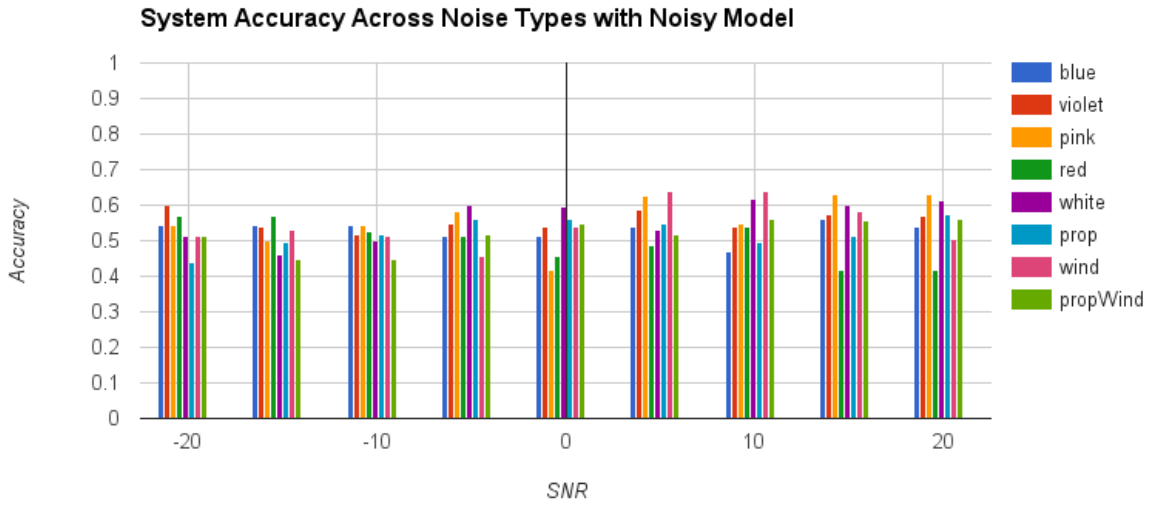

Figure 15: System accuracy of each noise type versus SNR, while using the noisy classifier

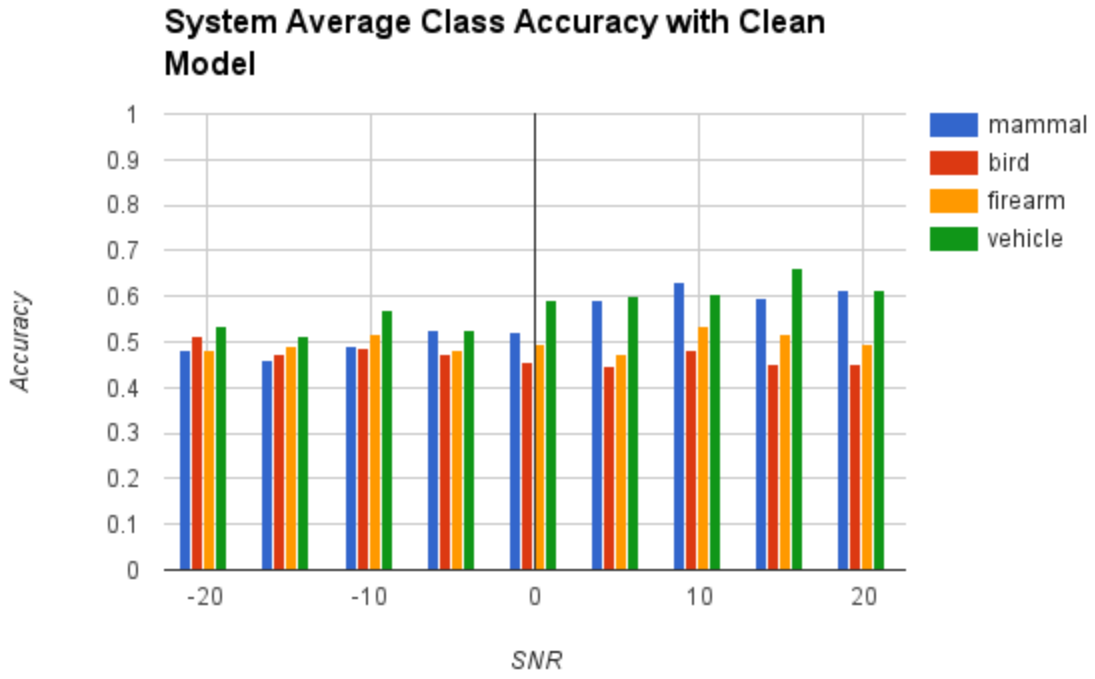

Figure 16: System accuracy for each class versus SNR, while using the clean classifier

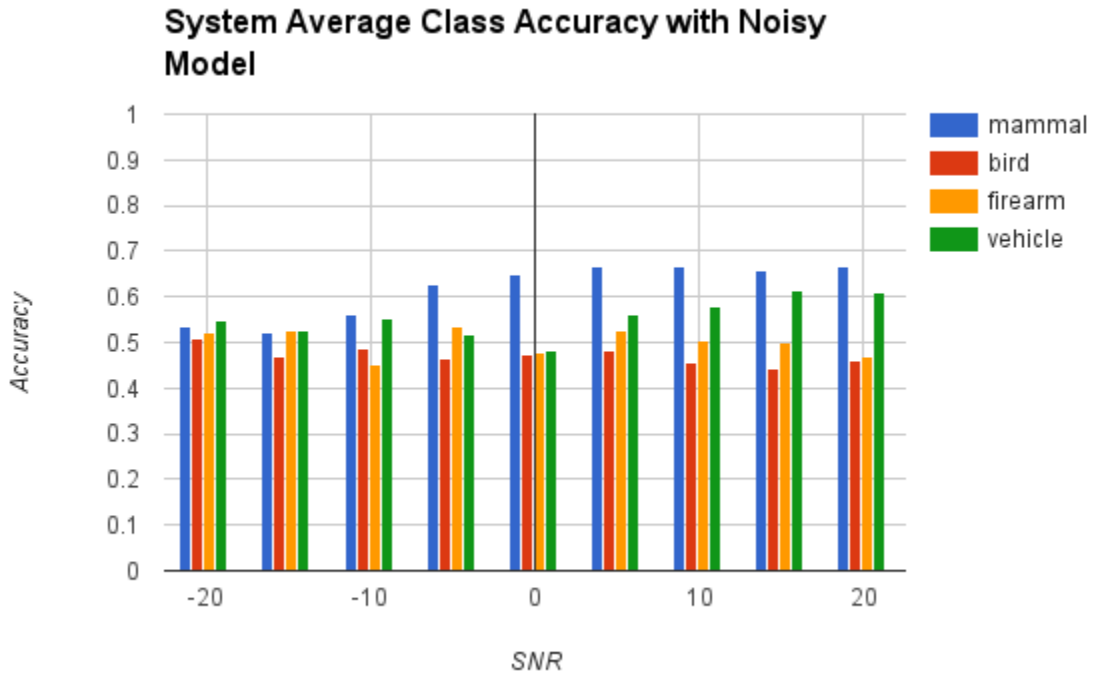

Figure 17: System accuracy for each class versus SNR, while using the noisy classifier

Conclusion

The source separation algorithm implemented is very robust to all noise types. It produced a consistent increase in Signal to Distortion Ratio (SDR). The SDR can probably be improved by using another cost function or by adding a regularization term to the cost function. If that doesn't improve the SDR, then the other thing that could be tried is expanding the dictionaries. These two solutions increase the computational requirement of source separation, so they are not without their drawbacks.

Classification is very susceptible to the noise level. A classifier trained with clean recordings showed a 7 percent drop in performance going from a Signal to Noise Ratio (SNR) of 20 dB to 10 dB. This susceptibility decreased by using noisy recordings as training data. The classifier showed better results even in the case when the noise added to the test set (natural noise) was not of the same type as the noise added to the training set (synthetic noise). The noisy model did not experience the same 7 percent drop in performance for synthetic noise until an SNR of approximately 0 dB.

The detector did not perform well under noise types not found in the training set. This came as a surprise, since the classifier showed that synthetic noise could be used to improve the accuracy of a classifier under real noise. The cause of this discrepancy is caused by the fact that the detector labels natural noise as a detected signal. The obvious solution for this problem is compiling a dataset of natural noise for training. A possible alternative could be using a wider range of synthetic noise. The detector performed very well under synthetic noise. It almost had perfect accuracy at an SNR of -10 dB.

The complete system performed poorly under all conditions. The cause of poor performance is suspected come from poor detection and poor source separation. The detector appears to be biased toward labeling every signal coming from the source separation stage as a

signal of interest. In other words, the detection stage is not able to detect signals containing pure noise. The cause of this is that the source separation stage introduces enough distortion to pure noise signals to change the decision of the detector. The source separation stage appears to distort the signals enough to cause the classification stage to be poor. Even with an SNR of 20 dB the source separation introduces enough distortion to lower the accuracy of the classifier to 50 percent.

Bibliography

[1] Miguel A. Olivares-Mendez et al., "The NOAH project: Giving a chance to threatened species in Africa with UAVs," in *International Conference on e-Infrastructure and e-Services for Developing Countries*, 2013, pp. 198-208.

[2] Futoshi Asano, Yoichi Motomura, Hideki Asoh, and Toshihiro Matsui, "Effect of PCA filter in blind source separation," in *Proc. ICA*, 2000, pp. 57-62.

[3] Daniel D. Lee and H. Sebastian Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556-562.

[4] Cyril Joder, Felix Weninger, Florian Eyben, David Virette, and Björn Schuller, "Real-time speech separation by semi-supervised nonnegative matrix factorization," in *International Conference on Latent Variable Analysis and Signal Separation*, 2012, pp. 322-329.

[5] Mikkel N. Schmidt, Jan Larsen, and Fu-Tien Hsiao, "Wind noise reduction using non-negative sparse coding," in *2007 IEEE Workshop on Machine Learning for Signal Processing*, 2007, pp. 431-436.

[6] Cédric Févotte, Rémi Gribonval, and Emmanuel Vincent, "BSSEVAL toolbox user guide--Revision 2.0," 2005.

[7] Steven M. Kay, Fundamentals of statistical signal processing: Detection theory, vol. 2, 1998.

[8] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, *The elements of statistical learning*.: Springer series in statistics Springer, Berlin, 2001, vol. 1.

[9] Ian Jolliffe, *Principal component analysis*.: Wiley Online Library, 2002.

[10] Comparisonics Corporation. (2016) FindSounds. [Online]. http://www.findsounds.com/

[11] Graham Grindlay, "NMFLib-Efficient Matlab library implementing a number of common NMF variants," *URL-http://www. ee. columbia. edu/grindlay/code. html*, 2010.

[12] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*.: John Wiley & Sons, 2009.

[13] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, pp. 1462-1469, 2006.

[14] Fabian Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[15] Dmitry Bogdanov et al., "Essentia: An Audio Analysis Library for Music Information Retrieval.," in *ISMIR*, 2013, pp. 493-498.

[16] Julius O. Smith and Jonathan S. Abel, "Bark and ERB bilinear transforms," *IEEE Transactions on Speech and Audio Processing*, vol. 7, pp. 697-708, 1999.

[17] Geoffroy Peeters, A large set of audio features for sound description (similarity and classification) in the CUIDADO project, 2004.

[18] Kristoffer Jensen and Tue Haste Andersen, "Beat estimation on the beat," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, 2003, pp. 87-90.

[19] Gerard Roma et al., "Ecological acoustics perspective for content-based retrieval of environmental sounds," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, p. 1, 2010.

[20] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis, "Comparative evaluation of various MFCC implementations on the speaker verification task," in *Proceedings of the SPECOM*, vol. 1, 2005, pp. 191-194.

[21] Paul M. Brossier, "Automatic annotation of musical audio for interactive applications," Queen Mary, University of London, Ph.D. dissertation 2006.

[22] Cyril Laurier et al., "Indexing music by mood: Design and integration of an automatic content-based annotator," *Multimedia Tools and Applications*, vol. 48, pp. 161-184, 2010.

[23] Mart\́in Haro and Perfecto Herrera, "From Low-Level to Song-Level Percussion Descriptors of Polyphonic Music.," in *ISMIR*, 2009, pp. 243-248.

[24] Simon Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*, vol. 120, 2006, pp. 133-137.

[25] George Tzanetakis and Perry Cook, "Multifeature audio segmentation for browsing and annotation," in *Applications of Signal Processing to Audio and Acoustics, 1999 IEEE Workshop on*, 1999, pp. 103-106.

[26] Fabien Gouyon and Perfecto Herrera, "Exploration of techniques for automatic labeling of audio drum tracks instruments," in *Proceedings of MOSART: Workshop on Current Directions in Computer Music*, 2001.

Curriculum Vitae

Carlo Lopez-Tello
Las Vegas, NV
qcarlox@gmail.com

EDUCATION
Master of Science in Electrical Engineering
UNLV - Las Vegas, Nevada - Expected Dec. 2016
Bachelor of Science in Computer Engineering
UNLV - Las Vegas, Nevada - May 2014

RESEARCH EXPERIENCE
Department of Electrical and Computer Engineering, Howard R. Hughes College of Engineering, UNLV
Master's Research, 2014-2016

Source separation of mixed acoustic signals.
Classification and detection of acoustic signals.
Specifically mixtures of mammals, birds, firearms, and vehicles.

TEACHING EXPERIENCE
UNLV, 2015-2016

In charge of teaching a micro-controller lab.
Duties involved teaching the concepts for each assignment, and helping students troubleshoot their code and circuits.
The lab used both AVR assembly and C.
Graded coursework for both lab and lecture sections.