MADR: Music Anomaly Detection and Removal

Ahaan Kanaujia

Department of Computer Science
University of Illinois at Urbana-Champaign
ahaank2@illinois.edu

Shayan K. Azmoodeh

Department of Computer Science
University of Illinois at Urbana-Champaign
shayana3@illinois.edu

***Abstract*—We propose a method for efficient supervised detection and removal of anomaly noises in music to clean audio clips of interruptions. We take advantage of the information encoded in the spectrogram of audio clips and assume that each frame can be classified as anomaly or music independent of the other pieces of the clip. This allows for *targeted* detection and removal of undesired noisy portions of audio, which are then imputed using standard techniques. Furthermore, we provide unsupervised extensions to our methods, enabling localized anomaly detection thus further boosting efficiency.**

I. Introduction

In this paper, we explore music audio denoising in the time-frequency domain. The problem of noise in music audio is prevalent in many different scenarios. For instance, music recordings often contain noise, such as a ringing phone, traffic noise, or applause. We aim to detect and replace these anomalies present in audio data.

Most classical denoising techniques involve spectral subtraction [1], where background noise is removed from the audio by estimating the noise profile and subtracting it from the signal. This is often performed in the frequency domain by analyzing the magnitude spectrum of the audio signal. The first step is to approximate a noise profile from the audio signal, which represents the spectral characteristics of the background noise. The second step is noise reduction using spectral subtraction. The noise profile is subtracted from the magnitude spectrum to obtain a cleaner audio signal. One significant drawback of this approach is the presence of processing distortions, or remnant noise in the cleaned audio signal.

Wiener filtering is one the most common and effective classical technique for denoising. It applies a linear time-variant filter to the noisy signal to obtain a clean audio. However, it requires a noise-only segment of the audio to create a noise profile. This is a problem since we rarely find pure noise in an audio segment. It usually overlaps with speech or music audio, which we want to retain. Thus, it is difficult to generate an accurate noise profile of pure noise in an audio signal. Our proposed approach overcomes this issue since it classifies spectral vectors of the audio as music or noise, instead of requiring pure noise segments in the audio.

Modern deep learning based approaches have become very popular for denoising. Models are trained to learn a mapping between noisy audio and its corresponding clean audio version, which enables them to estimate a clean signal from a noisy audio input. Popular models include Wave-U-Net [2] and SEGAN [3]. These large models require a very large amount of data and computational power to train. Moreover, Wave-U-Net and many other such models perform denoising in the time domain, and avoid spectral transformations. We propose a frequency-domain approach to denoising by applying a short time Fourier transform (STFT) to the input signal, and splitting the resulting complex valued spectrogram into its magnitude and phase components.

The algorithm proposed in this paper classifies every spectral vector of the spectrogram as noise or music. Based on this classification, the noise vectors are removed from the spectrogram to create gaps in the frequency representation of the audio. These gaps are then imputed using different methods, including nearest neighbours, singular value decomposition, vector autoregressive models (VAR), and some deep learning methods. Furthermore, we draw inspiration from the human perceptual system's ability to detect disturbances in an an audio clip using only that clip as context to propose a clustering-based unsupervised method to classify spectral frames as music or not.

Our underlying assumption is that the sacrifice of useful audio or music from removing the noise, can be accurately predicted by different models. Thus, the noise present in the audio is completely removed, and the music audio lost is imputed based on the surrounding spectral information. Hence, our approach does not distort any of the clean audio, it only changes the section of the audio that is noise. Other methods such as signal separation through techniques such as NMF are unable to reconstruct the clean audio perfectly.

II. Methods

We provide an overview of the methods used for performing decomposition and reconstruction of audio data: classification, anomaly removal, and imputation [4] [5]. The corresponding code can be found at https://github.com/AhaanKanaujia/MusicAudioDenoiser.

A. Supervised Classification

The first step of the algorithm is to train a classifier to classify spectral column vectors of the spectrogram of the input audio signal as music or noise. We experiment with

a gaussian classifier, a principal component analysis (PCA)-based classifier, and a K-Nearest Neighbors (KNN) classifier, ultimately finding that the Gaussian classifier performed the best at the cost of performance.

1) Training Data: To train our classifier, we use the FSD50K (Freesound Database 50K) [9], an open dataset of human-labelled events containing $51,197$ Freesound clips unequally distributed in 200 classes draw from the AudioSet Ontology. Due to constraints in time and computational power, we selected a subset of 39 sound classes, with 14 classes for noise, and 25 for music. We use a 80-20 split, using 80 audio files for training, and 20 for testing from each class. We convert every audio clip to a spectrogram using the inbuilt SciPy signal function. This creates a $129 \times n$ matrix, with n spectral column vectors for every audio clip. For every class, we concatenate these matrices to create the training data matrix for that class. Finally, we combine all the class training data matrices to obtain the final training data. Since high dimensional data such as this suffers from the curse of dimensionality, we perform PCA to reduce the number of dimensions to 20 frequencies. This dimensionality reduction is able to capture 98% of the variance in the original data, thus almost no information is lost.

2) Gaussian Classifier: The Gaussian classifier fits a multivariate normal distribution to each class in training data and classifies new data points based on the posterior probability density of the point under each distribution using the Bayes decision rule (i.e., assign the class that has the highest posterior probability given the input data point). This effectively determines a quadric decision boundary that can be used to classify any given data. An example of the such decision boundaries can be seen in Fig.1.

We define a conditional probability $p(x|y = c)$ and combine this with the class prior $p(c)$ to obtain the class posterior for each class

$$p(y = c|x) = \frac{p(x|y = c)p(y = c)}{\sum_{c'} p(x|y = c')p(c')} \tag{1}$$

We assume $p(x|y = c) = \mathcal{N}(x|\mu_c, \Sigma_c)$, where μ_c and Σ_c are the mean and covariance matrix of a class found by maximum likelihood estimation.

3) PCA Classifier: PCA is used commonly to reduce the dimensionality of a dataset and to extract features that are more informative about the data they represent. This allows classifiers trained on this feature representation to better learn about how each class is represented and thus better distinguish between them. However, PCA can also be used to directly classify points (i.e., PCA is the classifier). This is done by performing PCA to find the "best" subspace for each class within the original space the data lives in. Then given an input data point, we can classify the point as the class of the subspace it aligns the most with.

More precisely, given a dataset $\mathcal{D} = \{(\mathbf{x_i}, y_i)\}$ with classes $y_i \in \{\omega_1, \ldots, \omega_k\}$, we compute the PCA projection matrix W_i for each class and classify a given $\mathbf{x}$ as $\text{argmax}_{\omega_i} \|W_i\mathbf{x}\|$.

Here, each class is getting its own subspace *within the original space that the data lives in*. We are not synthesizing new features to represent the data, but rather finding a subspace that best explains the data in each class. Each class is effectively being reduced to a summary, which can be used to determine which class a new data point should belong to. These differences are demonstrated visually in Fig.2. The advantage of this over using PCA on all the data and then training a classifier is that we are not losing information about how the data in different classes orient themselves with respect to each other using the original features. However, this can also be a drawback since the additional dimensions may be unnecessary and mask important features of the data.

4) Nearest Neighbours Classifier: The intuition behind K-nearest neighbour classification is to classify spectral frames based on the vectors in the training data that are closest to it in an ℓ^1 norm sense. Let us consider a dataset, $S = \{x_i\}$, composed of column vectors x_i, which are concatenated spectrograms of audio clips. These vectors are defined by a set of frequencies F obtained from the short-time-Fourier-transform. Each column vector is labelled with a class label, and is either music or noise. Our objective is to classify an unknown column vector y of the spectrogram of an audio clip. For each $x_i \in S$, we calculate the distance between y and x_i as follows:

$$d(y, x_i) = \sum_{f \in F} w_f |y - x_i| \tag{2}$$

This is a summation over all the features in F, with w_f as the weight for each feature. Based on this distance metric, the K-nearest neighbours are selected. Then, these neighbours are used to determine the class of y. The most common technique is to assign the majority class among the nearest neighbours to the unknown vector. We can also assign more weight to the nearest neighbours when deciding the class of the unknown vector.

B. Unsupervised Classification

Our current approach trains a Gaussian classifier on a large amount of audio data, with samples for both noise and music. It also uses this training data to train a VAR model, which imputes the missing data in the spectrogram representation of an input audio clip. However, from hearing a simple audio clip and viewing the spectrogram, we should be able to approximate the time segments containing noise. In other words, we might not require a large amount of training data to classify a spectral frame as music or noise.

An unsupervised imputation method such as SVD and nearest neighbors does not perform well because it only contains the input audio as a reference to make predictions. This is not enough context to make a prediction, and leads to predictions that sound repetitive and distorted. The trained VAR model performs much better since it contains more data and can consider different samples while making a prediction. So, we continue using a VAR model for performing imputation.

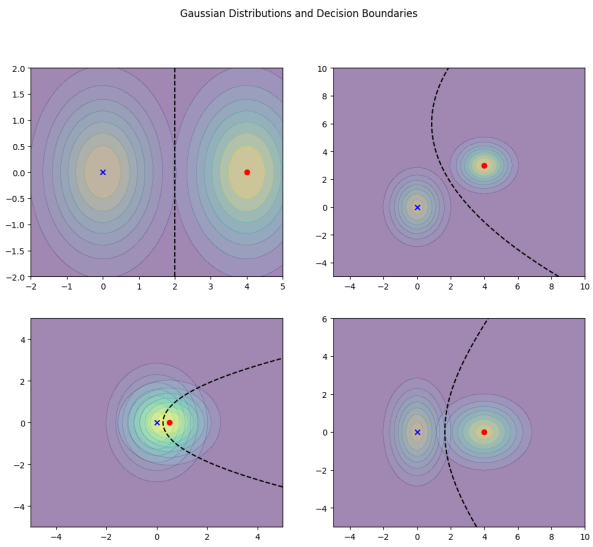

Fig. 1. Some possible decision boundaries arising from the Bayes Decision rule fitting a Gaussian distribution to each class. Boundaries of varying complexity are possible based on the complexity of the covariance matrices of the Gaussians. Top left image corresponds to Gaussians with unit covariance; all others have non-unit diagonal covariance.

1) Clustering: We cluster the spectral frames in the spectrograms into two clusters and associate one with unwanted noise and another as music. We can use a simple unsupervised classification such as K-means. However, K-means is known to struggle with outliers. Since our input audio data can contain a variety of different noise subsounds, we would also prefer soft assignments of points to clusters, which K-Means does not provide. Thus we utilize the Gaussian mixture model (GMM) to perform this clustering. The removal and imputation methods still remain the same.

A Gaussian mixture models a data distribution as a weighted sum of multiple Gaussian distributions, each of which is identified by $k \in \{1, ...K\}$, where K is the number of clusters we want to detect in our spectrogram. Each Gaussian is comprised of a mean μ, covariance Σ, and mixture probability π. The mixture probabilities for all the clusters must add up to 1, $\sum_{k=1}^{K} \pi_k = 1$. The Gaussian density function is:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{2\pi^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)} \quad (3)$$

Then, we maximize the model likelihood $\mathbb{P}(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$, using the Expectation-Maximization process. We initialize the means, covariance matrices, and priors to a random value. We can also use a simple k-means results for initialization. Next, for the expectation step, we find how much each Gaussian explains every available data point. Finally, in the maximization step, we weigh the data and re-estimate all the parameters. We alternate between these steps until convergence to obtain the final model parameters.

Due to the unsupervised nature of this approach, we must manually specify which cluster is music and which is noise. One approach to automatically detect the noise and music cluster is to check the number of spectral vectors classified as noise. Since we are dealing with audio clips with short bursts of noise, we should expect far fewer spectral vectors to be classified as noise than music. So, the cluster with lesser spectral vectors can be considered the noise cluster. Alternatively, we can use the average energy of the frames (as discussed below) in each cluster to determine which cluster contains noise frames.

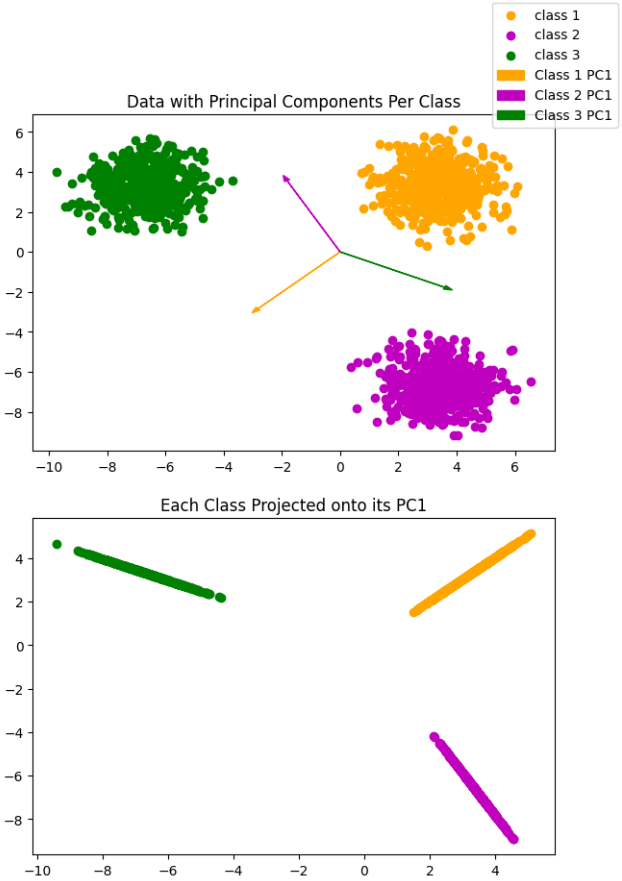

Fig. 2. Two-dimensional Gaussian classes projected onto their subspace spanned by their first principal component. Each class gets its own subspace in the original space the data lives in keeping information about the original features of the data.

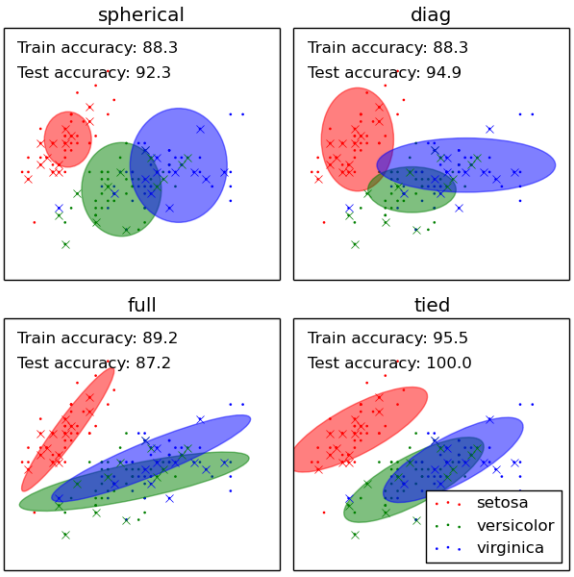

Fig. 3. A Gaussian Mixture model with 3 clusters. We obtain different Gaussian shapes due to the covariance matrix parameters based on the matrix type, which can be spherical, diagonal, full, or tied. [11]

2) Energy Analysis: Another unsupervised approach is the check the energy of spectral vectors. We observe that the energy of a spectral frame increases sharply during a segment of noise that deviates from the audio profile of the rest of the clip. This segment is especially pronounced for noise subsounds such as clapping or shattering. Based on the noise levels, we can apply an energy threshold, where all the spectral frames above a certain energy are considered as noise and all other frames are treated as music. However, this approach is difficult to generalize as this property is not necessarily universal. For example, audio clips with bursts of interleaving music and noise will not be satisfy this property and thus the frames will not be classified well.

C. Noise Removal

After classifying our input audio spectrogram, we then remove the spectral vectors classified as noise. During segments of silence, the spectral vectors have low energy. These segments appear white in the spectrogram. We ignore these frames during removal, since they are prone to misclassification by the Gaussian classifier.

To account for contextual information in the spectrogram, we apply a windowing function to the classification predictions. Due to the high granularity of the spectral vectors, there don't exist continuous segments of the predictions which are music or noise. To obtain these continuous and smooth predictions, we consider a window of variable size, and classify it as the class which is found maximum times within the window.

D. Audio Imputation

1) SVD and NMF Methods: The SVD and NMF imputation methods are simple imputation methods which we tried fitting. The SVD model [7] finds appropriate statistics to describe the entire spectrogram with the missing data, and imputes the missing part using a low rank reconstruction. This process is repeated until convergence. The NMF method [6] works similarly, where we take the NMF decomposition of the entire spectrogram, and reconstruct the missing part using the obtained W and H matrices. However, these models are additive and the only context they have is the input audio spectrogram. Thus, if the gap in the audio is large, these models are bound to fail. They work well for smaller gaps in the spectrograms. Moreover, since they utilize global statistics of the spectrograms, missing data with complex structure and unexpected samples are not imputed well.

2) Nearest Neighbor: The nearest neighbours approach [8] is also a simple process, where for each of the missing spectral vectors, we calculate its distance between it and the non-missing spectral vectors. Based on the distance metric, we find the k nearest neighbours. Then, the missing spectral vector is approximated as a weighted average of the available nearest neighbours. This is a local method that looks for similar areas within the spectrogram, and uses those statistics to perform imputation. However, this approach is again unsupervised, and cannot generalize the new audio data. For that, we consider a supervised VAR model that can learn from multiple different audio signals to make predictions.

3) Vector Autoregressive Model: We train a VAR model on dataset of classical music to obtain a matrix A that can be applied to predict the next element of a series of spectrogram frames given the past N entries. This can then be used to impute missing frames in an audio clip's spectrogram.

Given a time series of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_K \in \mathbb{R}^d$, the VAR model looks to find matrices $A_1, A_2, \ldots, A_N \in \mathbb{R}^{d \times d}$ such that for at each time step t,

$$\mathbf{x}_t = A_1\mathbf{x}_{t-1} + A_2\mathbf{x}_{t-2} + \ldots + A_N\mathbf{x}_{t-N} + \epsilon \qquad (4)$$

where ϵ is an error term.

We find these A_i matrices for a single time series (looking to minimize the error ϵ) as follows. First we construct a matrix $X \in \mathbb{R}^{dN \times (K-N)}$ (K is the length of the sequence) where the i-th column is created by stacking the $\mathbf{x}_{i-1}$ through $\mathbf{x}_{i-N}$ (starting with $i = N$). We then stack all the A_i matrices side by side to get a matrix $A \in \mathbb{R}^{d \times dN}$. Then performing the multiplication $Y = AX + \epsilon \in \mathbb{R}^{d \times (K-N)}$, the t-th column of Y contains exactly the expression for $\mathbf{x}_t$ shown above for the VAR model, and thus it is the predicted t-th time step by the model. The A that minimizes the prediction error in the least-squares sense can then be found as $A = YX^+$ (setting $\epsilon = \mathbf{0}$), which is what we want.

In our case, we construct the matrix X by concatenating the spectrograms of training music clips back to back and use 100 time lags for the model. Our computed A will then allow us to predict a frame in the spectrogram given the past 100 frames, which we can use to impute audio data.

TABLE I
CLASSIFIER PERFORMANCES

Classifier	Music Accuracy	Noise Accuracy
Gaussian	90.34%	85.1%
KNN	89.4%	84.1%
PCA	65.9%	60.5%

III. RESULTS

The accuracy of the supervised classification models are listed in table I. These models are tested on a subset of the selected audio files from the FSD50K dataset.

After testing, we found that although the KNN classifier performs almost as well as the Gaussian classifier, it significantly overfits to the dataset we are working with. As a result, the KNN model gave inaccurate predictions when applied to new input audio clips. Since we are using a small value of $k = 2$, the model is prone to overfitting. Removing the low energy frames and applying a window to the predictions also did not significantly improve the model performance. Thus, for classification and removal, we found that the Gaussian classifier works best.

We apply the Gaussian classification model to classify spectral vectors of the spectrograms of different music audios with noise. In Fig.4, we plot the spectrogram of an audio clip which contains the piano part of the song *Love of my Life* by Queen, with some applause and other noise. This audio clip contains a large segment of noise that occupies approximately 20% of the clip. Fig.7 shows the predictions made by the Gaussian classifier. Music is classified as 0, denoted by the red points. Noise is classified as 1, denoted by the blue points. To improve the accuracy of the classifications, we removed low energy frames and applied a window of size 25. Based on the plot, we can see that the classifier correctly captures the noise frames, and removes them in image 5. We can also use the unsupervised Gaussian mixture model to make classifications. These predictions can be seen in 8. Based on the plot, we can tell that the blue points labelled 1 is noise, while the red points labelled 0 is music. The imputed spectrogram is shown in Fig.6.

We also test a recording of *Für Elise* by Beethoven with bursts of claps in the middle (spectrogram shown in Fig.9). This clip is perfectly suited for our proposed approach as the noise comes in very short, but still noticeable, segments, and removing it does not affect any of the actual music. Due to the short time segments of noise, we used a window size of 5 when classifying points. The classifier's frame predictions are shown in Fig.12. Here, noise is red and music is blue. These points are removed to obtain the missing data spectrogram shown in Fig.10. The classifications made by the unsupervised GMM are also shown for comparison in Fig.13. As can be seen, the clustering if very similar to that of the supervised classifier. Based on the plot, we can tell that the blue points labelled 1 is noise, while the red points labelled 0 is music. The imputed spectrogram is shown in Fig.11.

IV. DISCUSSION

From the spectrograms and their classifications, we see that the unsupervised model works almost just as well as the supervised model. However, this is not representative of the classification ability of our models. Due to the nature of the audio clips, there is only 1 audio source for music and noise. In the love of my life clip, the music portion is mainly piano, while the noise portion is applause. Thus the GMM model is able to create accurate clusters since they are already very well defined. However, if we incorporate more musical instruments and more types of disturbances in the audio, we will notice that the unsupervised model is more prone to incorrect classifications as the difference between spectral frames of certain instruments and noise classes may not be as clearly separable. This is not an issue for the supervised model as it is trained on multiple different files of different types of music and noise enabling it to handle the more subtle differences. Thus, while the unsupervised model performs accurate classification on simple audio clips with fewer types of music and noise present, the supervised model remains the preferred choice for more complex clips.

We further looked to deep learning model such as Variational Autoencoders or Recurrent Neural Networks to improve the imputation quality; however, we these model proved difficult to train effectively due to their the large amount of computation power required given the size of our dataset.

Our approach is limited in that we assume short time bursts of noise, such as claps, applause, or a car speeding away. These short clips of noise create small gaps in the spectrogram, which can be imputed using various methods. However, for large gaps in the spectrogram, it is harder to impute since there is far lesser contextual information in the form of ground truth spectral vectors. Continuous noise, such as traffic in the background, or white noise will be difficult to remove using our approach as it overlaps with a majority of the music. Removing such noise from the spectrogram will lead to loss of useful music data that is needed to create an imputation that is pleasing to the ear. For such noise, we believe source separation methods may yield better performance as there is a consistent signal for both noise and music components.

A common issue arising in audio processing leaning on spectrograms is the problem of phase reconstruction to invert the modified spectrogram to an audio file; the methods in this paper are no exception. We opt to use the original phase of the noisy audio in inverting the imputed spectrograms. This is sensible as we work under the assumption that the desired clean audio is still present even during noisy segments, but at a lesser degree. As the phase is not exact in the imputed areas, the output audio can have white noise in the background. The Griffin-Lim phase reconstruction algorithm [10] was also considered, however this often corrupted the entire audio file, even those parts that were not imputed.

V. Conclusion

We propose a supervised and unsupervised data imputation approach in the time-frequency domain that can detect and replace short segment noise anomalies in music audio. We train various classifiers using the FSD50K dataset to classify spectral vectors of an input audio clip as music or noise achieving the best performance with the Gaussian classifier. Noise spectral frames are then removed and imputed using various imputation methods including SVD, NMF and also a VAR model trained on classical piano music to perform the imputations. With more computational power we hope to improve our imputation methods and build stronger classifiers using more of the available data. Future work will aim to reduce the white noise in the imputations resulting from sub-optimal phase reconstruction following imputation.

References

[1] Navneet Upadhyaya and Abhijit Karmakar, "Speech Enhancement using Spectral Subtraction-type Algorithms: A Comparison and Simulation Study."

[2] Daniel Stoller, Sebastian Ewert, Simon Dixon, "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation."

[3] Santiago Pascual, Antonio Bonafonte, Joan Serrà, "SEGAN: Speech Enhancement Generative Adversarial Network."

[4] Paris Smaragdis, Bhiksha Raj, Madhusudana Shashanka, "Missing Data Imputation for Time-Frequency Representations of Audio Signals."

[5] Paris Smaragdis, Bhiksha Raj, Madhusudana Shashanka, "Missing Data Imputation for Spectral Audio Signals."

[6] Jingjing Xu Yuanshan Wang Xiangnan Xu Kian-Kai Cheng Daniel Raftery and Jiyang Dong "NMF-Based Approach for Missing Values Imputation of Mass Spectrometry Metabolomics Data"

[7] Miklós Kurucz András A. Benczúr Károly Csalogány "Methods for large scale SVD with missing values"

[8] Lorenzo Beretta and Alessandro Santaniello "Nearest neighbor imputation algorithms: a critical evaluation"

[9] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, Xavier Serra "FSD50K: An Open Dataset of Human-Labeled Sound Events"

[10] D. Griffin, Jae S. Lim "Signal estimation from modified short-time Fourier transform"

[11] Scikit-Learn GMM Classifier

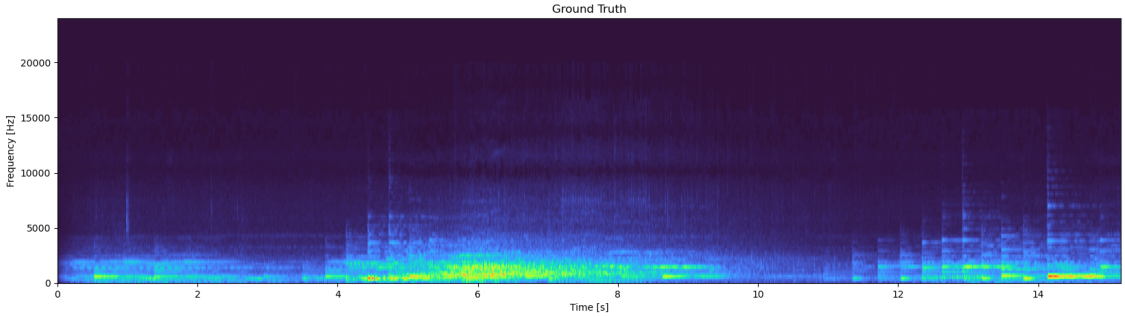

Fig. 4. Spectrogram for *Love of My Life* by Queen. We notice the applause is present between time 5 and 10s based on visual inspection. There is also some distortion at around 14 seconds.

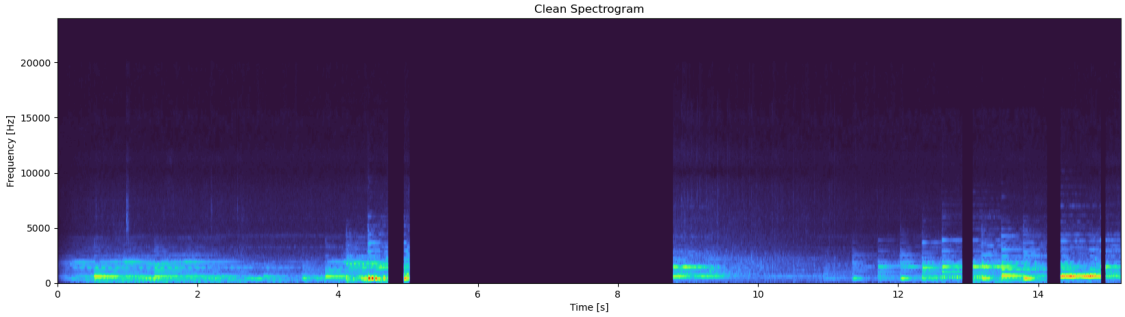

Fig. 5. *Love of my Life* spectrogram with noise frames removed.

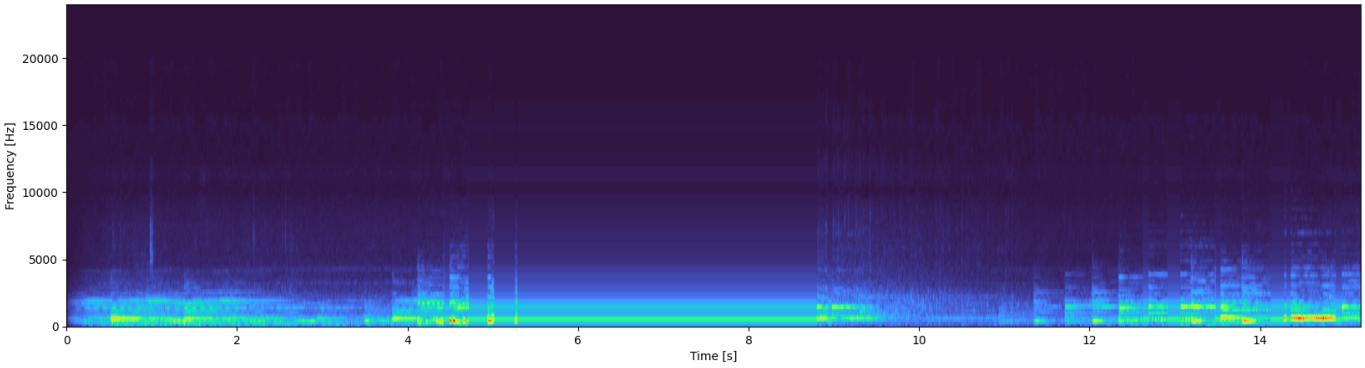

Fig. 6. *Love of my Life* spectrogram removed frames imputed using the KNN imputer. The imputation does not work very well and the resulting audio is just a lot of noise in that area due to the large gap that needs to be filled.

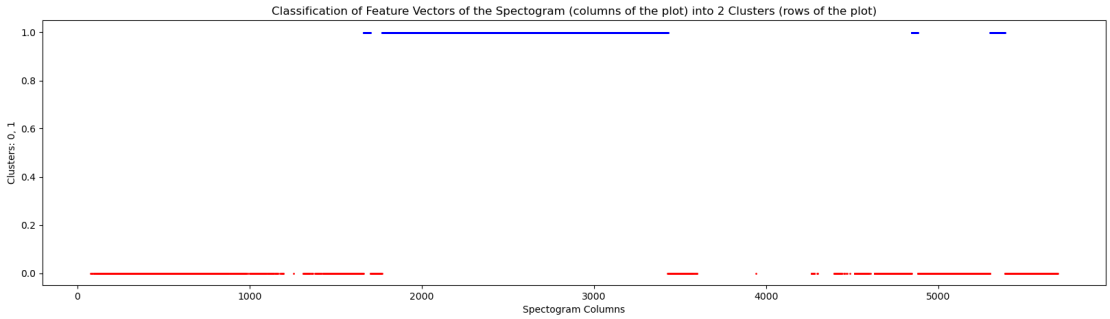

Fig. 7. Classified spectral vectors of the the *Love of my Life* spectrogram using the supervised Gaussian classifier model. Blue points are those labelled as music and red indicates disturbances in the audio. The spectral frames corresponding to the red dots are removed and imputed.

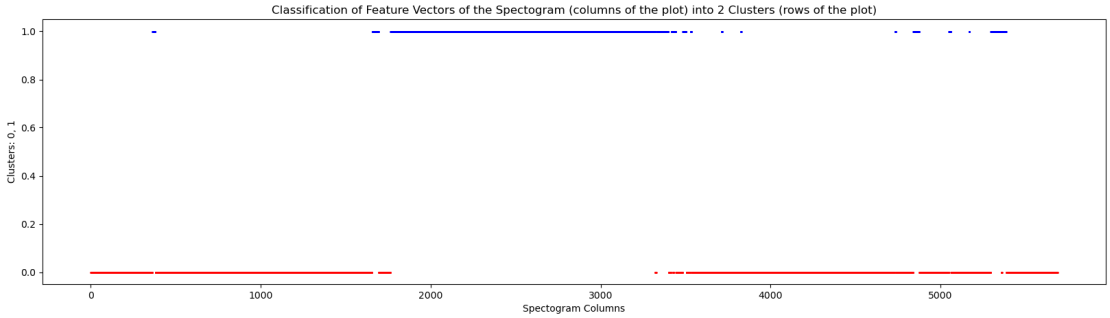

Fig. 8. Classified spectral vectors of the *Love of my Life* spectrogram using the unsupervised Gaussian mixture clustering model. This model operates knowing only the data available within the spectrogram of the given audio clip. Either by visual inspection or by analyzing the energy of the frames in each cluster, we can deduce that the blue dots correspond to music frames and the red frames are noise.

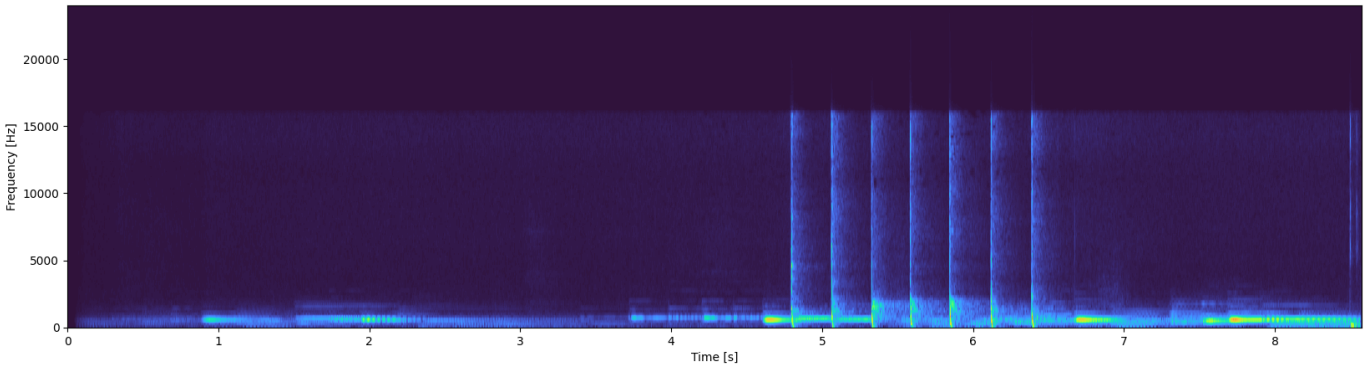

Fig. 9. We can see sudden bursts of noise present at around 5 to 6.5 seconds. We can clearly tell they are noise upon visual inspection.

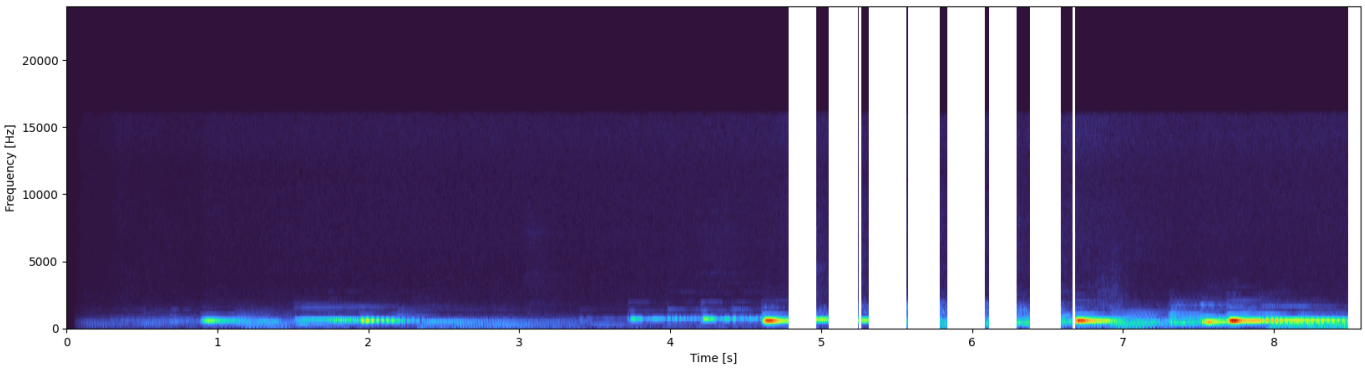

Fig. 10. Fur Elise Spectrogram with Noise Removed

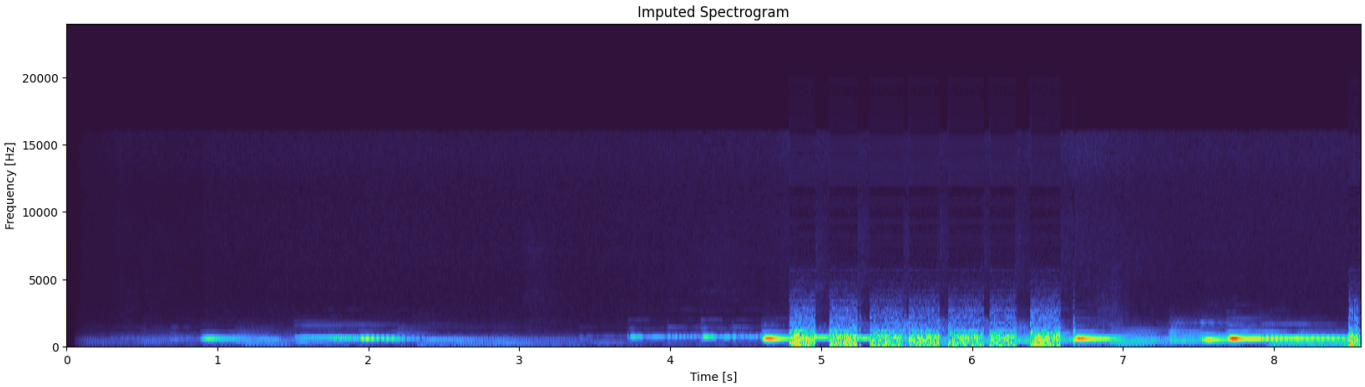

Fig. 11. Fur Elise Spectrogram with removed frames imputed using a VAR and KNN model. The missing frames are first imputed using a KNN imputer. The imputed values are then used as input to the VAR model to get new predictions to use as the imputation. This two-step process results in imputed audio segments where the music is amplified with respect to the white noise that is present in the background.

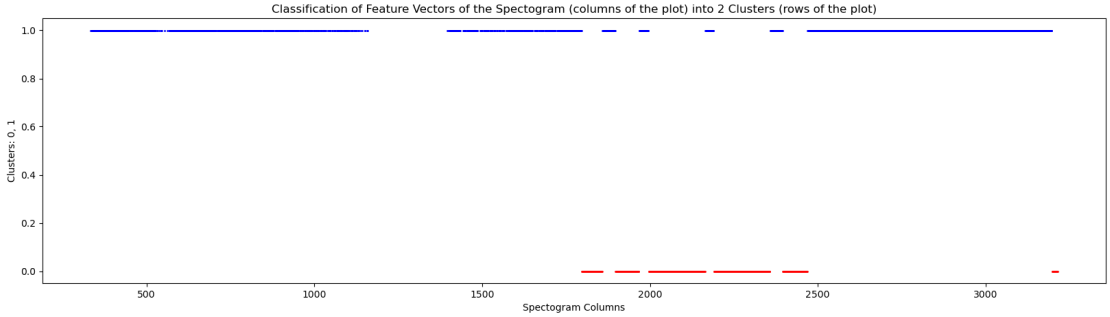

Fig. 12. Classified spectral vectors of the Fur Elise spectrogram using the supervised Gaussian model.

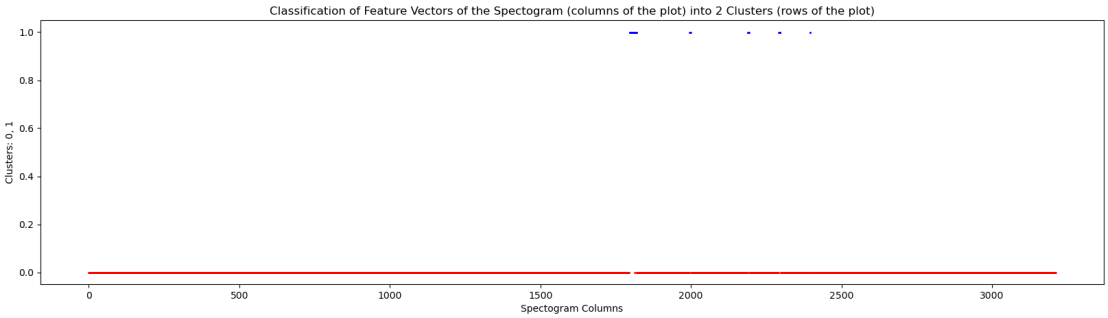

Fig. 13. Classified spectral vectors of the Fur Elise spectrogram using the unsupervised Gaussian mixture model.