Music Instruments Classification

Darshan Narayanaswamy
MSCS, 2nd year
narayanaswamy.d@northeastern.edu

Viet Nguyen
MSCS Align, 2nd year
nguyen.vi@northeastern.edu

Xiaoyu Fan
MSAI, 1st year
fan.xiaoy@northeastern.edu

I. Introduction

Classification is among the most important topics in machine learning. There are abundant applications of classification problems in real life that human beings can solve with the help of Machine learning algorithms. An interesting application for classification problems is to classify a given audio file and identify the predominant musical instrument. Music data, compared to other types of data, is more complex since it often involves multi-dimensional feature extraction. An audio sound can be represented in the form of a wave, which then can be described in terms of different characteristics: amplitude, wave length, frequency, velocity, and time-period. Different combinations of characteristics can result in different features that uniquely belong to an audio file.

In this project, we aim to explore and apply various machine learning methods for classification of audio data in music. Our main goals were to explore different features that can distinguish the instruments, learn how to extract these features in the most meaningful way, and finally learn to recognize and classify the predominant instrument for a given music audio file using suitable Machine learning methods.

The results demonstrated that recognizing the predominant instrument of a song is feasible by taking machine learning approaches. While numerous audio features can be extracted from a music piece, only a few of them largely contribute to solving our problem and demand dedicate processing to maximize their impact in providing a satisfying solution.

A. Dataset

We chose an open-sourced dataset called Instrument Recognition in Musical Audio Signals (IRMAS) [1] for our problem. The data set provides a diverse collection of more than 6705 audio files from different types of instrument. Each audio excerpt lasts about three seconds from more than 2000 distinct recordings. The instrument categories include piano, human singing voice, flute, organ, trumpet, and acoustic guitar.

For the purpose of training and validation we split our dataset in 80-20 ratio, with 5364 training examples which are validated against 1341 test dataset.

B. Data Exploration

Common audio features [2], [3] are extracted from the dataset using the Librosa library in Python. This package provides convenient methods to retrieve various audio features. The main features include:

1. Zero-crossing rate (ZCR): the number of times in which the amplitude of audio signals passes through a value of zero.

2. Spectral centroid (SC): the weighted average of all the frequencies in the sound.

3. Spectral roll-off (SR): the frequency below which a certain proportion of the whole range of frequencies is found. In other words, it is the qth percentile of the range of frequencies [4].

4. Spectral bandwidth (SB): the gap in frequency between the lowest and highest.

5. Chroma-mean feature: average representation of the entire range of frequencies in terms of 12 different pitch classes.

6. Root mean squared (RMS): the average of values of amplitude over a period of time.

7. Mel frequency cepstral coefficients (MFCC): a small set of coefficients (usually about 10-20) that compactly represents the spectral across time. This feature is among the most popular features used in speech and audio feature extraction [5]. In our project, our MFCC features include 16 coefficients expanding across 128 overlapping time frames.

Once all the features are in a tidy form, we then attempt to visualize the data set for several features.

The distribution of the mean of the audio frequency spectrum of different instruments are quite separate from each other, as seen in Figure 2. However, this would not be true in terms of the kernel density of another feature like the ZCR feature, as shown in Figure 3.

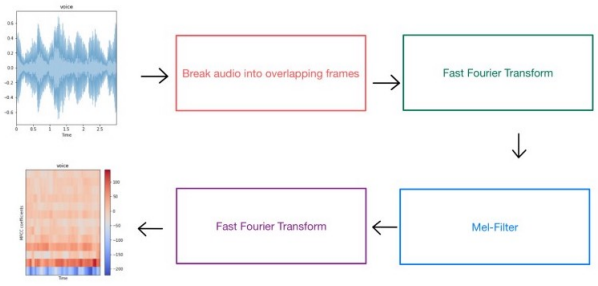


Fig. 1: Extraction steps for MFCC

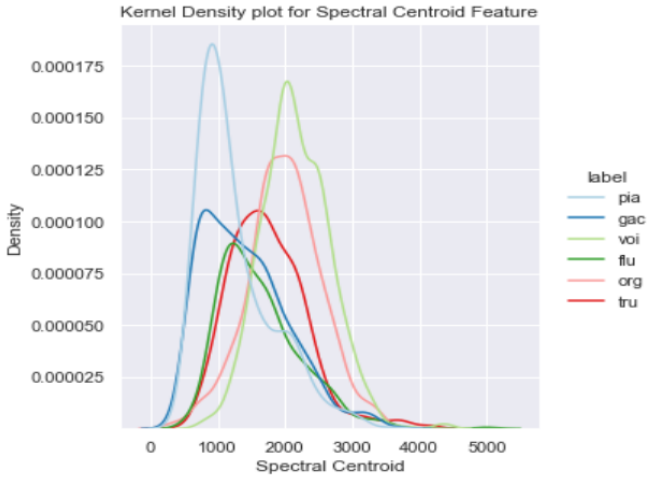


Fig. 2: Kernel density plot for Spectral Centroid Feature

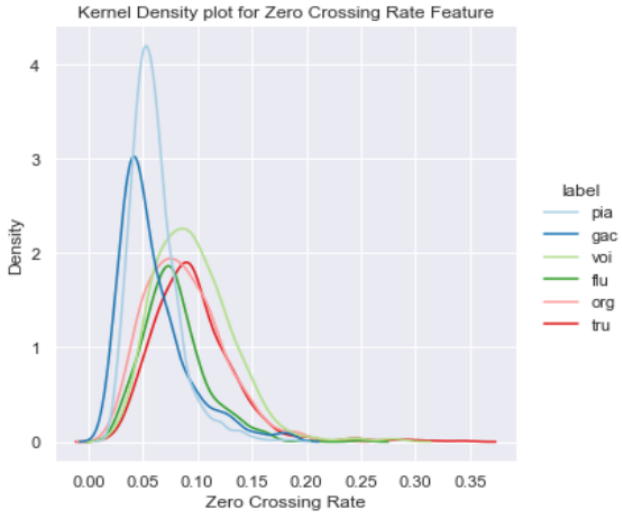


Fig. 3: Kernel density plot for Zero Crossing Rate Feature

Fig. 4: Chroma Feature by instruments

We can also look at the chroma and the RMS to further examine the distinctions across our sources of instrument. In Figure 4, trumpet has the highest chroma value while flute has the lowest value. Meanwhile, in Figure 5, the acoustic guitar has the lowest RMS value while organ has the highest value.

The data exploration demonstrates how different instruments stand out from the crowd due to their unique characteristics. Thus, it is best to include all the extracted features. However, this brings up the dimensionality problem.

Audio data consists of 6 features in the form of scalar variables and MFCC in the form of 16 cepstral coefficients for 128 overlapping time frames in the 3s audio; which is represented as an image with one channel. The 16 by 128 image matrix is flattened into a vector of 2048 feature columns along with the 6 feature columns, resulting in a total of 2054 features. Relative to the 5364 row samples in training data, the training set is bound to run into the curse of dimensionality. Hence, there is a need to reduce the higher dimensional space of features into a significantly lower dimensional set of features without losing much of the variance that captures the separation among different instruments.

Following this introduction, section II details several methods for dimensionality reduction. It also outlines the ML methods used for the classification problem, which includes SVM and CNN. The results of the recommended approach is shown in Section III. We then review the limitations and conclude our paper in Section IV.

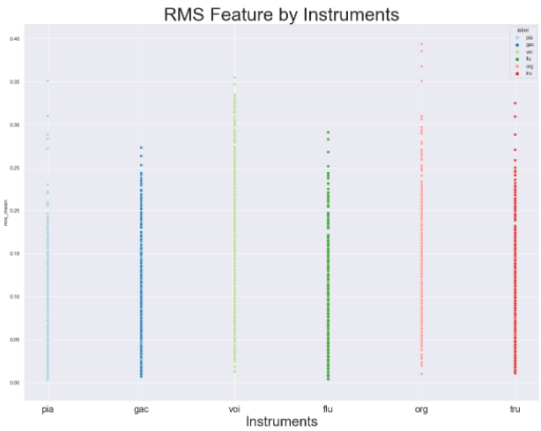


Fig. 5: RMS Feature by instruments

II. Methods

Methods of Dimensionality Reduction

A. Principal Component Analysis

Principal Component Analysis (PCA) is a linear technique used for dimensionality reduction. The maximum variance in data is captured by the eigenvectors of the covariance matrix with the highest eigenvalues; these eigenvectors are called the principal components. The principal components are linear transformations of original feature columns and are orthogonal to each other, which means the components have zero correlation.

Only a subset of these components are enough to explain the maximum variance of the underlying data. Figure 6 shows that about 90% of the variance can be explained using the first 250 principal components. This reduces our original feature space by 88%.

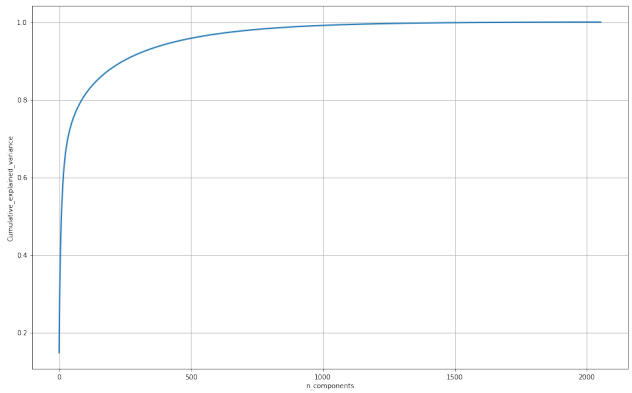


Fig. 6: PCA variance

B. Autoencoder

Autoencoders are simple neural networks used to learn feature mappings in an unsupervised manner. Autoencoders reduce the high dimensional feature space using non-linear transformations into lower dimensional feature mappings. Figure 7 shows a simple architecture of autoencoders [6], with one hidden layer. The shape of the input layer is the same as the shape of the output layer, whereas the hidden layer has a significantly lesser number of neurons.

Autoencoders are broken down into two subnetworks, encoder and a decoder. Encoder compresses the original high dimensional feature space into lower dimensional mappings. Decoder use these lower dimensional feature space to reconstruct the original higher dimensional features. The output of encoders are the reduced feature mappings that can be used as training data to machine learning models like SVM.

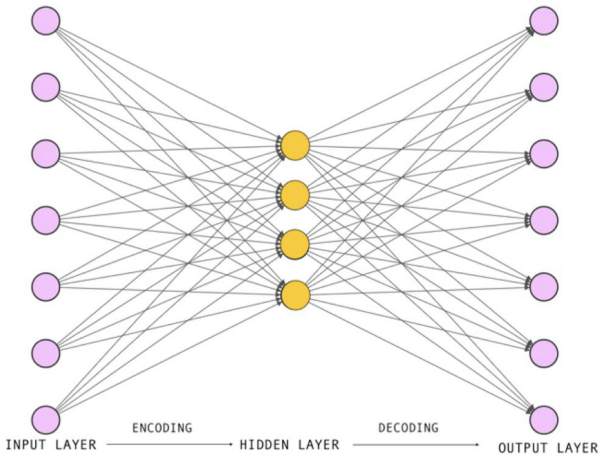


Fig. 7: Arcitecture of simple autoencoder

The number of neurons in the hidden layer defines the dimensions of our reduced feature set. There is no standard way to choose the dimension of this feature space unlike PCA, hence the choice is a hyper parameter that is chosen arbitrarily. In our simple architecture of autoencoder we chose 10% of the original feature space i.e 204 neurons with relu activation and mean squared error as a loss function. Figure 8 shows the MFCC images of different instruments(top row) and their reconstructed images(bottom row) with just 10% of the reduced feature space.

C. PCA vs Autoencoder

PCA is computationally cheaper to run than autoencoders, but PCA only learns linear mappings of the original feature space. MFCC features are represented as images and autoencoders do a good job in learning the non-linear feature maps. Figure 9 from [7] shows a very

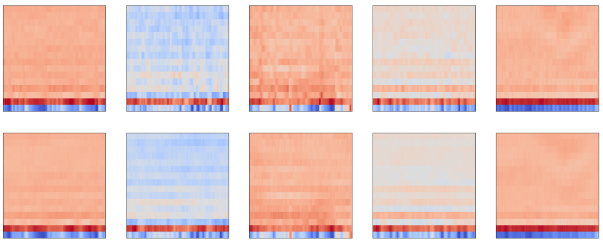


Fig. 8: MFCCs reconstructed with Autoencoder

good representation of mappings learned from PCA and autoencoders on a higher dimensional simulated curve.

D. Mean

MFCCs extracted from our 3s audio data are 16 cepstral coefficients across 128 overlapping timeframes. Since our original data spans only for 3s and the instrument frequencies and altitudes might not vary too much in 3s. A simplest way to reduce this data is to just take the mean of these 16 cepstral coefficients across entire 3s or 128 frames, which reduces our feature set from 128x16 to 1x16. Taking the mean alone does reduce our feature set significantly but we also lose the variance across time intervals, but this surprisingly gives us good results as shown in [2].

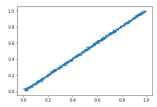
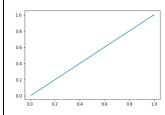
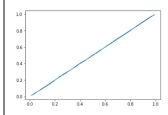
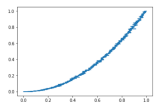
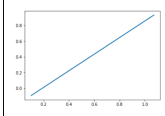
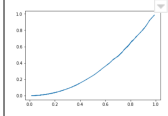
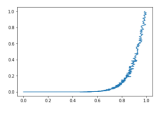
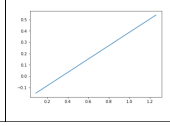
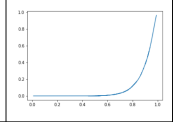
Function	Feature Space	PCA Reconstruction	Auto Encoder Reconstruction
y=mx+c			
$y=mx^2+c$			
$y=mx^8+c$			

Fig. 9: PCA vs Autoencoder

Classification Methods

E. Support Vector Machine

Support Vector Machine (SVM) is a widely used method for classification problems with both linear and non-linear datasets. Three prevalent kernels were considered, namely linear, polynomial and Radial Basis Function (RBF), as making any assumptions about the dataset is not desirable.

RBF is one of the most generalized form of kernel functions because of its resemblance to the Gaussian distribution [8]. It projects feature vectors into infinite dimensions [9] and computes the similarity between two projected vectors based on their squared Euclidean distance, defined as [10]

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \quad (1)$$

For most kernel functions, finding an optimized combination of hyperparameters can lead to a good balance between bias and variance, and thus increase the accuracy and robustness of the model [11]. The traditional tuning method Grid Search evaluates on every single combination of predefined parameters in an exhaustive way [12] which, while effective, is computationally expensive and relies on prior human knowledge. An alternative tuning strategy that does not suffer from the curse of dimensionality, which is another limitation of grid search, is Random Search. It randomly draws samples from the search space to evaluate and therefore can outperform grid search, especially in the settings where a small number of parameters greatly affect the model performance [11] [13]. Our project adopted random search for hyperparameter tuning.

For polynomial kernel, we tuned parameters degree d and gamma γ. A larger degree of polynomial usually yields an overfitting model [14]. Gamma is an influence parameter that decides the weight of a single training sample. A larger gamma means a smaller radius of other examples that have influence and hence a less smooth model.

For RBF kernel, we tuned parameters C and gamma γ. C determines how many examples are allowed to be misclassified and a larger C leads to a higher demand of classification correctness and a less smooth model. The importance of gamma is the same as the one in polynomial kernel [15].

F. Convolutional Neural Networks

Convolutional neural networks (CNN) is a class of neural networks which works well with spatially local input features with the help of convolving smaller kernels onto the original data. CNNs are well suited to select these spatially local features like edges on image or equivalently frequency/amplitude shifts in audio data. MFCCs capture the shift in frequencies on mel-scale across different time frames, and these local shifts shown as patterns in the 128x16 mfcc image can be trained on a CNN model. Most of the state of the art results on

audio data today are obtained by training MFCCs with CNN [16].

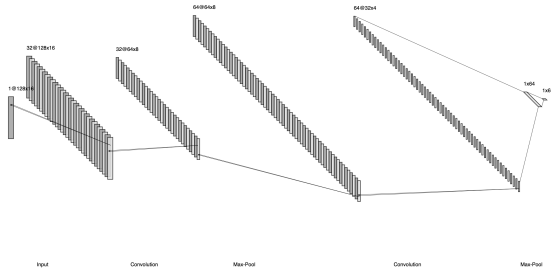


Fig. 10: Architecture of CNN

Architecture Our CNN model architecture was kept relatively shallow with two Conv layers and two fully connected layers as shown in Figure 10. Architecture of the model was limited to two Conv layers because an introduction of a third Conv layer resulted in numerical instability and weights underflow resulting in loss function to return Nan after some iterations of training. We also noticed that the validation loss curve was deviating significantly from the training loss curve and introduction of dropout layers as a form of regularization helped in avoiding overfitting for the training data. Relu was the preferred activation function as it helps us avoid vanishing gradient problem. Adam optimizer (Adaptive Moment Estimation) was chosen as the gradient descent algorithm after experimenting with various optimizers. Adam optimizer adapts a varying learning rate depending on the parameters and resulted in a faster convergence of loss function. Figure 11 shows the noisy loss of training the CNN model and 12 shows the change in accuracy vs training epochs.

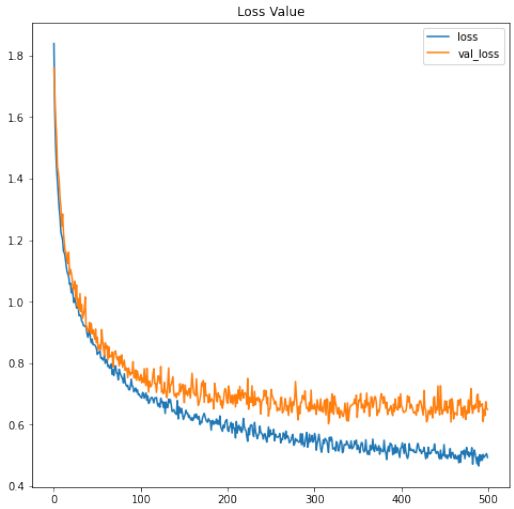


Fig. 11: CNN Loss vs epochs

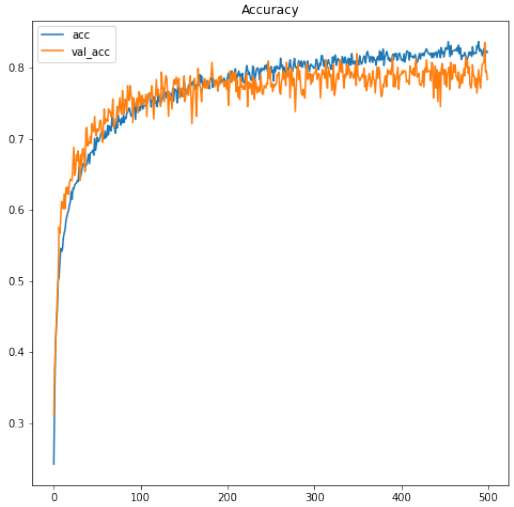


Fig. 12: CNN accuracy vs epochs

III. Results

A. Evaluation criteria

As the project's problem focuses on classifying an instrument into the right category and neglects the difference between false positive and false negative, **accuracy** was chosen as the evaluation metric of our models' performance and was used both in the hyperparameter tuning and the final results reporting. We also reported **confusion matrix** in order to judge whether the not so perfectly balanced dataset can be further improved.

B. Dimensionality reduction

All dimensionality reduction methods boosted the accuracy as Table I shows, while taking the mean of all time spans for each feature offered the most significant enhancement. Surprisingly, PCA and autoencoders only promoted the performance slightly.

Training dataset	Accuracy
Original	56%
PCA reduced	65%
Autoencoders reduced	63%
Taking mean	81%

TABLE I: Accuracy of SVM with RBF kernel running on the original and reduced datasets

C. SVM

Table II shows that SVM with RBF kernel achieved the highest accuracy among all models. Polynomial gave

a less accurate model and linear kernel performs the worst.

Model	Accuracy
SVM w Linear	62%
SVM w Polynomial	74%
SVM w RBF	81%

TABLE II: Accuracy of SVM with different kernels

Confusion matrices shown in figures 13-15 exhibit a consistent distribution of misclassified examples among all classes. Therefore, we concluded that further accuracy increment strategies for imbalanced classes would not enhance the performance of our SVM models much.

D. CNN

Accuracy achieved on training MFCCs with CNNs are shown in the confusion matrix Figure 16 and the overall accuracy achieved was 78%. CNNs achieve state of the art accuracies without a need to perform any kind of feature engineering unlike SVM and other traditional machine learning models.

IV. Discussion

A. Limitations and further improvements

Our models were trained with songs of 6 instruments only so they are not capable of recognizing instruments outside of these 6 classes, which could be an important feature to have when deployed to a real-world setting. One way to handle missing classes nicely is to add songs predominated by instruments other than these 6 instruments into the training set and annotate them with a new class - *unknown*. In the inference stage, any song that does not fall into these 6 categories should be ideally classified as *unknown*.

Another limitation is our models can only recognize one single predominant instrument in a piece of music. When a song contains multiple instruments that a human would consider equally predominant, particularly in terms of loudness, our models would merely pick the class with the highest probably and ignores others with comparably high accuracy, which could also be a factor hindering the accuracy. SVM is originally designed for binary classifications by drawing decision boundaries between two classes. However, multiclass classification by SVM can be achieved through one-against-one method, which boils the problem down to a group of binary classification problems, or one-against-all approach, which compares each class against all the rest [17]. CNN, on the other hand, can support multiclass classification by adding a softmax activation layer at the end, providing a probabilistic interpretation of the model and allowing us to obtain the top-N classes with accuracy above a threshold [18].

Fig. 13: Confusion matrix of SVM with linear kernel

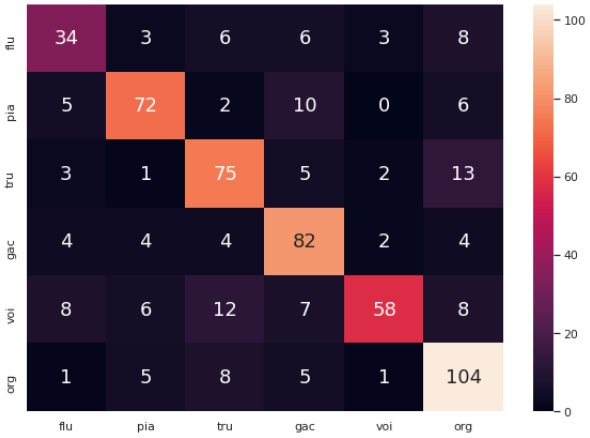


Fig. 14: Confusion matrix of SVM with polynomial kernel

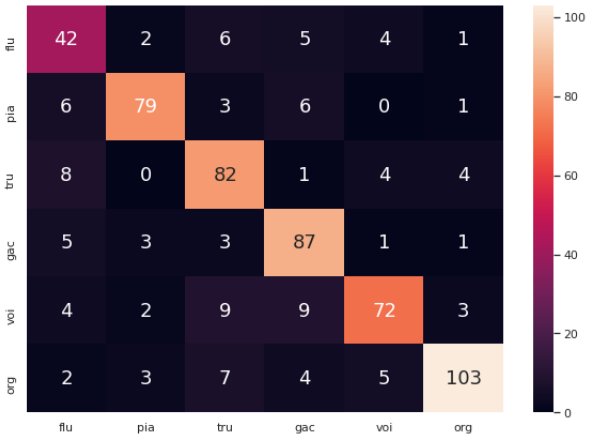


Fig. 15: Confusion matrix of SVM with RBF kernel

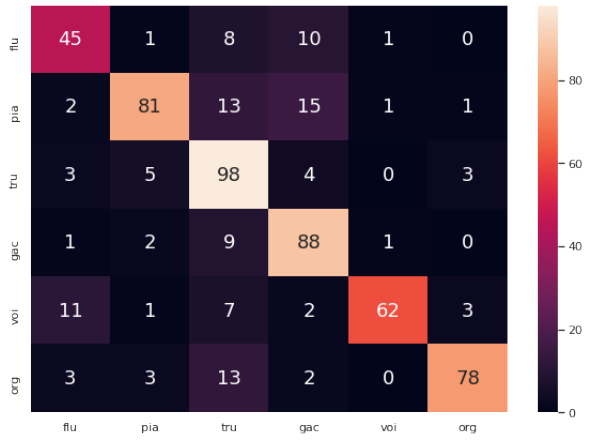


Fig. 16: Confusion matrix of CNN

It is also worth noting that all of our training samples are 3-second long. The reasoning behind is songs vary in duration and their predominant instruments can also vary at different point of time. A limitation caused by this split is - When fed with songs lasting more than 3 seconds in production, the models may not perform as well as the results we achieved in this project, as extended input data could indicate a mix of predominant instruments and introduce more noise to the classification problem. In practice, we could work around it by segmenting the input data into 3-second pieces, classify on each segment and combine the results,

To further boost the classification accuracy, fine tuning a pretrained imagenet model such as Alexnet or VGG-16 on the extracted MFCCs achieves state of the art results in other audio classification problems like Urbon-Sound8K as shown in [16], exploring similar approach of transfer learning on our dataset would be a step in the right direction.

B. Lessons learned

An important lesson we learned is whether we normalize our data makes a huge impact on the accuracy [19]. Compared with our preliminary experiments in which we failed to normalize our data, the accuracy of both SVM and CNN improved a lot this time.

V. References

[1] J. J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals." in *ISMIR*. Citeseer, 2012, pp. 559–564.

[2] K. Racharla, V. Kumar, C. B. Jayant, A. Khairkar, and P. Harish, "Predominant musical instrument classification based on spectral features," in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE, 2020, pp. 617–622.

[3] J. D. Deng, C. Simmermacher, and S. Cranefield, "A study on feature analysis for musical instrument classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 429–438, 2008.

[4] R. Rameshan, C. Arora, and S. D. Roy, *Computer Vision, Pattern Recognition, Image Processing, and Graphics: 6th National Conference, NCVPRIPG 2017, Mandi, India, December 16-19, 2017, Revised Selected Papers*. Springer, 2018, vol. 841.

[5] S. A. Alim and N. K. A. Rashid, *Some commonly used speech feature extraction algorithms*. IntechOpen, 2018.

[6] N. Patel, S. Patel, and S. H. Mankad, "Impact of autoencoder based compact representation on emotion detection from audio," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2021.

[7] U. Muaz, "Autoencoders vs pca: when to use whichnbsp;?" Jul 2019. [Online]. Available: https://towardsdatascience.com/autoencoders-vs-pca-when-to-use-which-73de063f5d7

[8] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear svm," *Journal of Machine Learning Research*, vol. 11, no. 48, pp. 1471–1490, 2010. [Online]. Available: http://jmlr.org/papers/v11/chang10a.html

[9] A. Shashua, "Introduction to machine learning: Class notes 67577," *CoRR*, vol. abs/0904.3664, 2009. [Online]. Available: http://arxiv.org/abs/0904.3664

[10] J.-P. Vert, K. Tsuda, and B. Schölkopf, "A primer on kernel methods," in *in Kernel Methods in Computational*. MIT Press, 2004, pp. 35–70.

[11] L. Villalobos-Arias, C. Quesada-López, J. Guevara-Coto, A. Martínez, and M. Jenkins, "Evaluating hyper-parameter tuning using random search in support vector machines for software effort estimation," in *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 31–40. [Online]. Available: https://doi.org/10.1145/3416508.3417121

[12] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, Tech. Rep., 2003. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers.html

[13] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[14] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, "Support vector machines and kernels for computational biology," *PLoS computational biology*, vol. 4, p. e1000173, 11 2008.

[15] V. Cherkassky and Y. Ma, "Practical selection of svm parameters and noise estimation for svm regression," *Neural networks*, vol. 17, no. 1, pp. 113–126, 2004.

[16] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking cnn models for audio classification," *arXiv preprint arXiv:2007.11154*, 2020.

[17] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[18] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[19] U. Jaitley, "Why data normalization is necessary for machine learning models," Oct 2018. [Online]. Available: https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029

VI. Appendix

Jupyter Notebook Links

- Data Exploration: https://colab.research.google.com/drive/1SK-n1mZDeeaALL2oMjZlGz-6Rjp1kuwm?usp=sharing
- PCA: https://colab.research.google.com/drive/1_NHDh923Tou33W5B0tf7_Ur-mLSvnINc?usp=sharing
- Autoencoder: https://colab.research.google.com/drive/1yNMejB64nx-HQPc4fhlBIdcbgVoSyZ0n?usp=sharing
- SVM: https://colab.research.google.com/drive/1MP5Uhu0_sYb2Ogys6ukZBpYWFjKF8zoi?usp=sharing
- CNN: https://colab.research.google.com/drive/1HTEPwkxW4PkX2rZ8rxNMD0wsoAurfYzM?usp=sharing

Documents

- Proposal Link: https://docs.google.com/document/d/1jM53f4qwFqc9dw4vQbnE8U5WJ4vwbU3aHjSeeL3_UEk/edit?usp=sharing

VII. Statement of contributions

Viet Nguyen: Data Pre-processing, Features Exploration and Data Visualization.
Darshan Narayanaswamy: Feature Selection(PCA and autoencoders) and CNN
Xiaoyu Fan: Support Vector Machines and Discussions