CS230

Music and Instrument Classification using Deep Learning Technics

Lara Haidar-Ahmad*
Department of Computer Science
Stanford University
larahdr@stanford.edu

Abstract

This paper presents our implementation of a multi-class classifier that identifies instruments in music streams. Our model consists of a CNN which's input is an audio stream that we pre-process to extract the mel-spectogram, and outputs the dominance or non-dominance of pre-selected instruments. We focus our study on 3 instruments, and thus classify audio streams into one of 4 classes: "Piano", "Drums", "Flute" or "Other". We obtained a precision of 70%, a recall of 65%, and a F1-score of 64%. As future work, we aim to implement and compare the results of more deep learning model architectures such as RNN, RCNN, CRNN, in addition to adding more instruments.

1 Introduction

Music is part of our daily life; instrument classification is a highly valuable task that could potentially enable the extraction of valuable information that in turn could contribute to tasks like music classification by genre and music search by instrument.

This project consists of implementing a multi-class classifier that inputs an audio stream and outputs the dominance or non-dominance of some pre-selected instruments, by labeling the audio stream with the class corresponding to the dominant instrument (if any). Due to the restricted time of the project, we scoped down the number of instruments to detect three instruments: piano, drums and flute. We will thus classify audio streams in one of 4 classes: "Piano", "Drums", "Flute" or "Other".

Our model architecture consists of a convolutional neural network which performs the classification. Our data consists in audio streams which we pre-process to extract the mel-spectogram. The input of the model is thus the mel-spectogram, and the output is an index corresponding to the predicted class.

This paper lays out our work as follows; in Section 2 we will discuss the related work and other state of the art. In Section 3 we will present the data used in our work. Section 4 will lay out the model and describe in detail the chosen architecture. In Section 5 we will outline the results. Finally, in Section 6 we will list our conclusions and discuss any future work.

The implementation of the model can be found on https://github.com/lara-hdr/music-classifier.

*Software Engineer at Microsoft. Github: https://github.com/lara-hdr. Alternative Email Address:haidar.lara@gmail.com

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

Music classification is a fairly popular task that has been studied previously by many machine learning researchers. Multiple research papers initially introduced SVMs and gaussian mixture models to tackle this problem; such as [1] and [2]. [1] is a study that classified 8 instruments using 2 gaussian mixture models and Support Vector Machines, and was able to acquire a 30% error rate using SVMs.

[2] used gaussian and k-nn classifier to classify instruments following a set of 43 features. They show the results for the individual instruments and also take into account a handmade taxonomy to try to assign a weight to the miss-classifications.

[3] is a work by Dresden University that uses hidden markov models to classify audio streams among 4 instruments on a small dataset (600 recordings).

The above techniques showed acceptable results, but use old algorithms that require hand picked features. More recent researches are turning into deep learning for this task, which removes the manual feature extraction process.

A research from Uppsala University [4] analyzed and compared the behavior of different features of the sound, on a model learning from the frequency spectrum of the audio samples.

Also, in paper [5], a similar study was conducted to identify 11 instruments. This latter study used a ConvNet with an architecture similar to AlexNet [6] and VGGNet [7], and achieved a F1 measure of 0.602 for micro and 0.503 for macro. Since our model and data is similar to this paper, we will compare our results in the final deliverable with the results of this paper.

Our literature study showed that most research in this domain used "clean" data with isolated sounds, consisting of a single instrument in each audio stream; our data however is closer to real world data. Since it is extracted from YouTube, each streams can have any kind of background noise and is not recorded under the same conditions, which adds complexity to the classification task.

3 Dataset and Features

AudioSet by Google [8] provides human labeled data, consisting of a set of 10 second clips from YouTube, labeled with the audio instruments they contain (and any other sound label). Given that the data is 'from the real world', this dataset might present additional processing challenges but also offers a more diverse set of audio streams that include background noise and involve multiple instruments along with natural sounds. The dataset covers a big number of audio streams, but given the time constraints of downloading big amounts of data and other faced challenges (http error 429 while sending a high number of requests to YouTube), we restricted our work to a total 2,400 samples for each class; composing a total of 9,600 samples using the same distribution and divided into 8000 samples for training, 800 samples for validation and 800 samples for the evaluation.

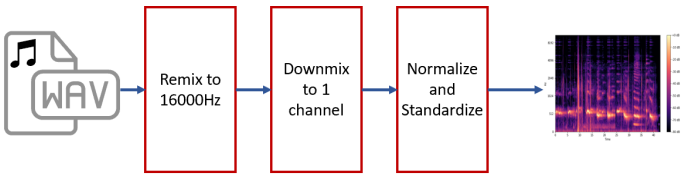

Figure 1. Audio Sample Pre-Processing Pipeline

We pre-processed the data, as shown in Figure 1, and set the frequency of all audio samples to 16000 Hz. We also pre-processed the streams by down mixing them to use 1 channel. Finally we padded shorter samples to 10 seconds followed by a normalization and a standardization steps. To represent the data, we use mel-spectograms, which are the input of our model. We generate such mel-spectograms using a sample rate of 16000Hz, a window size of 400, a fft size of 400, a hop size of 200 and 128 filter banks.

After our first deliverable, we noticed that we incurred into overfitting over the training data; to avoid that, we leveraged data augmentation techniques, and integrated random white noise to the audio samples of the training set before extracting the melspectograms.

4 Methods

Our model consists of a feed forward convolutional neural network. As mentioned previously, we pre-process the audio streams to extract the mel-spectrogram that is used as input of the model; the model is thus a model for image classification on the mel-spectrogram.

Our final model, shown in Figure 2, is a 7 layer CNN, in which each convolution has a kernel size of 2, a stride of 1 and 0 padding. The number of filters are 8, 8, 16, 16, 32, 32, 64 and 64 in increasing order from the shallow to the deep layers of our network. The maxpool nodes have a kernel size of 2. Finally the fully connected layer has 4 filters (number of classes). We did not add a softmax node at the end of the model since our loss function is cross-entropy, which already includes a softmax in PyTorch. We chose our loss function to be cross entropy since it has proven to work well with multiclass classifiers.

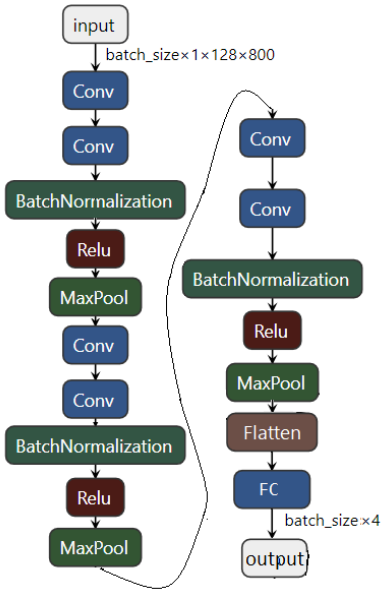

Figure 2. Model Architecture

To initialize the parameters we used He initialization which works better with ReLU activations.

5 Experiments/Results/Discussion

We trained on 8,000 samples, with batches of size 128, 15 epochs, and a learning rate of 10e-3. Before choosing these values, we tried multiple hyper parameters values; for the learning rate we landed on 10e-3 after multiple tunnng steps. For the mini-batch size, we chose 128, which was the highest value we could use without running into memory size issues. We used cross entropy as the loss function with Adam optimizer. Figure 3 presents the loss over the epochs for training and validation. We forced an early stop resulting on a total of 15 epochs after which we corroborated the model's trend to slowly start to overfit the training data. To verify that we did not overfit we calculated the evaluation metrics for the training data and compared them with the results for the validation dataset.

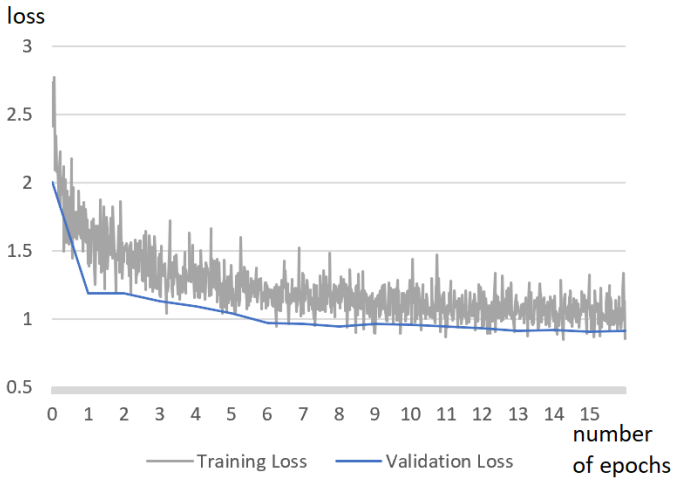

Figure 3. Training and Validation Loss over the Epochs

We computed the confusion matrix for the test dataset, which is shown in Table 1. We can observe that the biggest errors are between the classes "Drums" and "Other" where the model predicts "Other" for a sizeable amount of "Drums" samples (105 i.e: 52.5% of the drums samples).

actual \ predicted	Piano	Drums	Flute	Other
Piano	134	2	40	24
Drums	19	74	2	105
Flute	30	0	160	10
Other	41	5	6	148

Table 1. Confusion Matrix for the 4 Classes

We computed the precision, recall and metrivs F1-score for the datasets. It is worth noting that during training we used f1 as the optimizing metric. Table 2 displays the metrics for the test dataset. We attained an average precision of 70% , an average recall of 65%, and an average F1-score of 64%. We compare our results to a similar study in [5]. Their CNN attains a micro precision of 66%, a micro recall of 56%, and a micro F1-score of 54.1%. We obtain higher values for these metrics; however we should note that this study classifies audio streams into 11 classes which is a more difficult task than classification with 4 classes. On the other hand, our model was trained on more diverse data, that was extracted from YouTube, which can be more challenging to classify since it can contain any kind background noises and data closer to 'real world data'.

	Piano	Drums	Flute	Other	Total
Precision (%)	59.82	91.36	76.82	51.57	69.92
Recall (%)	67.00	37.00	80.00	74.00	64.50
F1-score (%)	63.21	52.67	78.43	60.78	63.77

Table 2. Precision, Recall and F1-score for the 4 classes and Total Data

We conducted an error analysis on 30 samples for each class to find out the model's causes of errors, and recurred to 5 annotators for some of the errors. Table 3 shows the main types of errors per class. The first column annotation "Other sounds" corresponds to audio samples where other sounds (asides from instruments) are dominant in the audio stream (mainly people talking), and/or when the targeted instruments only appears in the background. The "Unlabeled instruments" annotation corresponds to

the audio samples where both the predicted instrument and the labeled instrument appear without exhibiting a clear dominance (even though we made sure that the audio streams were annotated by at most one instrument, some were missing labels and had more than one dominant instrument in the stream). The "Mislabeled" annotation corresponds to audio samples where all the members of a set of 5 human beings would also classify the audio stream in concordance with the model. The annotation "Sounds like predicted class" corresponds to audio streams where at least 1 members of a representative set of 5 humans also miss-classified the audio sample with the predicted label, while the rest classified it with the tagged label. (eg. a motorcycle engine that should have been predicted "other" is predicted as "drums"). We can see that the annotation "Unlabeled instruments" represents the biggest portion of error types with an average of 40% of the errors, and as high as 76.67% of the errors for the class "Other". The majority of errors for this class are attributed to audio samples that contain music but is not labeled by any of the instruments. The second cause of error corresponds to the annotation "Sounds like predicted class", with around 43% of the errors for piano and 17% for the classes piano and flutes. By analysing the errors of these two classes, we noticed that many samples labeled as "piano" contained high pitched audio that could be attributed to electric keyboards (rather than classical piano), which tends to be confused with a flute (and vice versa) by both the model and at least 1 member of our set of human beings.

	Other Sounds (%)	Unlabled instrument (%)	Mislabeled (%)	Sounds like predicted class (%)	Other (%)
Piano	10.00	16.67	-	43.33	30.0
Drums	23.33	36.67	10.00	-	30.00
Flute	13.33	30.00	13.33	16.67	26.67
Other	-	76.67	-	10.00	13.33

Table 3. Error Analysis per class

For better results, we should clean the data and label the unlabeled instruments. Adding data for drums class (where other instruments don't appear) could also potentially improve our model. Other improvements could be by splitting and distinguishing between sub-classes if the data is available (eg. splitting piano into "classical piano" and keyboard)

6 Conclusion/Future Work

In this paper we were able to achieve a precision of 70%, a recall of 65% and an F1 of 64% for our multi-class classifier. With more time we would have tried more architectures like RNN that proved to be efficient with audio, or even RCNN or CRNN architectures. In addition, we would have cleaned the data and labeled the unlabeled instruments in the streams, and added more data samples for the class "Drums" (where other instruments do not appear), with the objective of making the model learn the features for the drum sounds. We would also have liked to support more instruments, and split the current classes into sub-classes (e.g. split piano into "keyboard" and "classical piano", etc.), which could help with cases where the model confuses piano and flute. Finally, expanding our target space by supporting the identification of more than one instrument in a stream would be a desirable improvement on our current model.

References

[1] Marques, J., Moreno, P. J. (1999). A study of musical instrument classification using gaussian mixture models and support vector machines. Cambridge Research Laboratory Technical Report Series CRL, 4, 143.

[2] Eronen, A., Klapuri, A. (2000, June). Musical instrument recognition using cepstral coefficients and temporal features. In 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100) (Vol. 2, pp. II753-II756). IEEE.

[3] Eichner, M., Wolff, M., Hoffmann, R. (2006). Instrument classification using hidden Markov models. system, 1(2), 3.

[4] Toghiani-Rizi, B., Windmark, M. (2017). Musical Instrument Recognition Using Their Distinctive Characteristics in Artificial Neural Networks. arXiv preprint arXiv:1705.04971.

[5] Han, Y., Kim, J., Lee, K., Han, Y., Kim, J., Lee, K. (2017). Deep convolutional neural networks for predominant instrument recognition in polyphonic music. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 25(1), 208-221.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[8] Google (2019), AudioSet. https://research.google.com/audioset

[9] PyTorch (2019), https://pytorch.org/

[10] TorchAudio (2019). https://pytorch.org/audio/

[11] AudTorch (2019). https://audtorch.readthedocs.io/en/0.4.1/