Detecting and Classifying Musical Instruments with Convolutional Neural Networks

Dominick Sovana Hing, Connor Joseph Settle
Department of Computer Science
Stanford University
Course: CS230: Deep Learning
Project Category: Audio
dhing@stanford.edu, csettle@stanford.edu

Abstract

This paper presents an implementation of a multi-class musical instrument classifier that examines an audio segment and determines which musical instruments are present. Our approach was to train a single-class classifier, then use transfer learning to train a model that can identify multiple instruments within the same audio clip. We discuss the performance of both our single-class and multi-class models and identify some of the challenges associated with the convolutional approach. We concluded that despite the challenges, CNNs remain a promising approach to this problem.

1 Problem Statement

Since the advent of audio-recording technology, digitized music clips have become commonplace, in the form of CDs, music streaming applications, YouTube videos, and more. It is common for musical listeners and performers to need to identify which kinds of musical instruments are used in a song, as well as when those instruments are playing. Instrument classification not only has the potential to benefit musical instrument learners, but also could have more widespread implications towards widespread organization of musical audio clips based on instrument.

In this paper, we describe the implementation and performance of an instrument detection convolutional neural network that is able to identify musical instruments within an audio clip. Our model is based off of the architecture used in a former CS230 project [1] on musical instrument classification, but is modified for a slightly different use case. Our neural network consists primarily of four convolutional layers, as well as two fully-connected layers. Our convolutional neural network could be used to service an application that accepts an arbitrary audio clip and tells the user which instrument is the most dominant at any 3-second interval of the clip.

This paper is formatted as follows. First, we will review existing techniques used in prior research papers on related topics. Then we will describe the dataset from which we trained our models, as well as the preprocessing and augmentation that was required. Next we will discuss our model architecture, learning strategy, and techniques we used to optimize performance (hyperparameter tuning and variance reduction). Finally, we will discuss our results.

2 Literature Review

[1] is a past cs230 project which used used Google's AudioSet, but only split the musical instruments into 4 categories. These four categories each represent a single instrument from 4 different instrument

classes (piano, flute, drums, and "other"). Additionally, this paper does not attempt to identify the times at which instruments are playing in audio clips, nor did it attempt to detect overlapping instruments.

https://cs230.stanford.edu/projects_fall_2019/reports/26225883.pdf

[2] uses a small dataset of 16 CD's, and classified into a larger number of categories than the previous paper. Two different models were tested, one using SVM's and the other with Gaussian Mixture Models. This study did not attempt to classify multiple simultaneous instruments being played.

http://www.bitsavers.org/pdf/dec/tech_reports/CRL-99-4.pdf

[3] attempts to model the human ear using a set of linear filters in order to identify different sources of sound in a more stochastic environment, including when multiple sounds are layered over one another.

https://www.researchgate.net/publication/200045103_Computational_auditory_scene_analysis

[4] uses CNN's to classify different individual musical instruments. The input to their CNN was a combination of spectrograms, along with a set of recurrence plots, which map how the phase of a sound frequency changes over time.

https://arxiv.org/ftp/arxiv/papers/1512/1512.07370.pdf

The preceding references gave us a broad overview of different approaches to the problem of musical instrument classification. It should be noted that [3] does not address specifically the problem of musical instrumentation, but focuses more on computational methods for auditory perception.

It should also be noted that these references do not specifically address the problem of multi-instrument classification, which is the problem of trying to identify all of the instruments present, whether or not concurrent instruments are playing within the audio clip. Nevertheless, these references provided us with a broad overview of numerous strategies and approaches that then influenced the approach to our specific problem.

3 Dataset and Preprocessing

To train our classifier, we used the IRMAS (instrument recognition in musical audio signals) dataset from the Universitat Pompeau Fabra in Barcelona.

https://www.upf.edu/web/mtg/irmas

This dataset consists of two parts, already pre-split into a training set and a test set. The training set consists of 6705 audio files, each of which are 3 second stereo .wav files that are annotated with the type of instrument being used. In the training dataset, each audio file is labelled with only one type of instrument. The test set consists of 2800 stereo .wav files, each of which is between 5 and 20 seconds. In this set, the labels may correspond to more than one instrument at a time. In total, there are 11 different labels, each pertaining to a different musical instrument: cello, clarinet, flute, acoustic guitar, electric guitar, organ, piano, saxophone, trumpet, violin, and human singing voice. Audio samples are taken from a wide range of musical styles (e.g. Jazz vs. Classical), contain a variety of different musical articulations, and were made using a wide range of recording techniques.

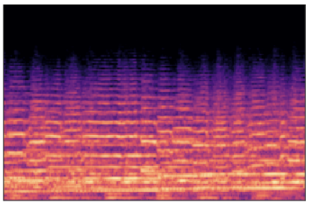

Figure 1: One channel of a decibel mel-spectrogram.

To use this dataset, we needed a significant amount of preprocessing to create an effective training set. To avoid a data mismatch problem, we split the 6705 training examples into around 700 validation examples and 6000 training examples. At this point, we have a training set of 6000 clips, each 2-channel (stereo) and 3 seconds long. We were able to augment this training set by swapping the left

and right channels, thus duplicating each training example and giving us a total of 12,000 training examples. We preprocessed these clips using a python sound processing library called librosa, to generate normalized mel-spectrogram images on a decibel scale. This outputs a set of 128×259 spectrograms, each with 2 channels, giving us a total input volume of [batch_size] $\times 128 \times 259 \times 2$.

We performed the same augmentation on our validation set, giving us 1400 test examples. Note that we made sure to split the train and validation sets before performing data augmentation. This ensures that we don't have one audio clip in the train set and its clone (with the channels switched) in the validation set, which could make the two sets dependent.

4 Learning Methods

We started our initial model architecture based off of the previous CS230 project that we researched. After some tuning and testing, we arrived at our current architecture.

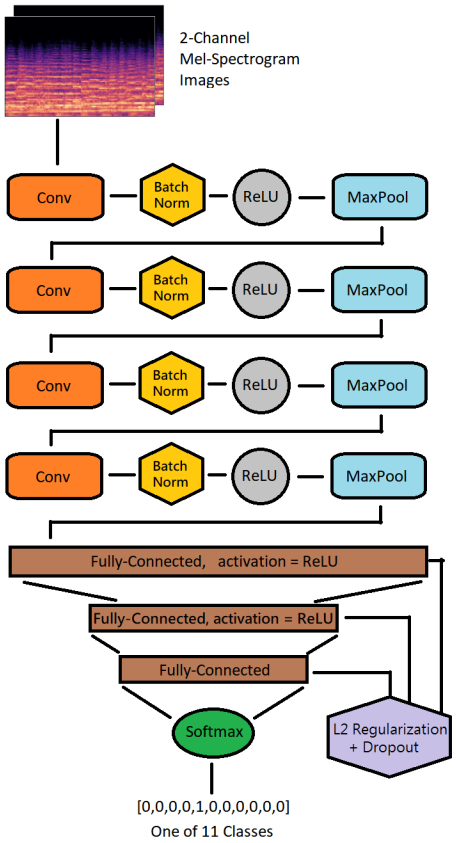

Figure 2: Current model architecture

The first conv layers uses a 4×4 kernel with 8 filters. The second conv layer uses a 3×3 kernel with 16 filters. The third conv layer uses a 2×2 kernel with 32 filters. Finally the fourth conv layer uses a 2×2 kernel with 64 filters. Every MaxPool layer uses a 2×2 kernel with stride 2. Our first Fully-Connected layer has 500 hidden units, followed by another fully-connected layer with and the second one has 11 hidden units (one for each class). Finally, we use a Softmax output node with categorical cross entropy loss and an Adam optimizer.

Our overall approach to solving the problem of multi-instrument classification was to train our convolutional neural network on the augmented IRMAS training set examples. For training, we split the augmented IRMAS train-set using a 9010 ratio: 90% for training and 10% as a validation set

to evaluate the single-class model. This meant that we ended up training our weights on just over 12,000 examples.

Once we had the convolutional layers trained at an acceptable performance, we removed the last layer and generated a new set of models, each with the the same weights and architecture, but replacing the softmax layer with a few additional hidden dense layers followed by an output with 11 sigmoid activation nodes, each corresponding to an instrument class. Each output node independently attempts to detect whether a particular instrument is present in a clip. This allows us to create a multi-class output. When training the multi-class model, we kept the pre-trained weights the same and only trained the weights for the new layers.

We felt that this transfer learning technique was appropriate to use because the amount of overlap between the problems of single and multi-class classification. This approach was also partially determined by the constraints of the IRMAS dataset. The IRMAS training set had single-class labels, while the IRMAS testing set had multi-class labels.

5 Hyperparameter Tuning & Variance Reduction

When performing our single class training, we initially excluded the Batch Normalization layers. At first, we felt that they were reducing our performance, and that batch normalization was less useful since the dataset was hand-crafted (and perhaps not as varied as a different and larger dataset we were also considering using). After some experimentation, we found that the batch normalization did improve the performance while training, both on the training subset of the IRMAS train-set and the validation subset of the IRMAS train-set.

Additionally, we found that augmenting our dataset by flipping the two channels also improved training performance. This is likely because most of the recordings did not have identical left and right channel data, which gave our model more opportunities to learn at each convolutional layer.

We also tested a range of hyperparameters to maximize performance. We altered the batch size, trying values of 16, 32, and 64, and found the best results with a batch size of 32. We also tested different training durations, measuring the accuracy on the train and validation sets every 5 epochs. The best results showed up at around 30 epochs, and performance plateaued after that point. Lastly, we experimented with different rates of dropout, from .15, .25, .35, and .5, with the best results coming from .5 and .25 based on the size of the hidden layers.

Without these optimizations, we found that our single-class training exhibited dramatic overfitting. Our validation subset would have around 55% accuracy while our training subset would have nearly perfect accuracy. By including these optimizations, we found that the training subset accuracy decreased to about 70.3% accuracy, but our validation subset accuracy increased by several points, to about 60.4% accuracy. While our model still exhibits some overfitting, we felt that we had reduced it to an acceptable level that would still make for a decent machine learning model.

6 Evaluation

Upon training our original model to the IRMAS training set, we were able to achieve a 70.3% training. accuracy and a 60.4% validation set accuracy. The results can be summarized using the two tables below, which are based off of the 10% validation split from the IRMAS training set. Thus, this data includes about $1,000$ validation set examples.

These results indicate a comparable result to the the benchmark given to us by [1], which reports (overall) a precision of 70%, recall of 65%, and an F1 of 64% for their model. This benchmark may not be an exact comparison, since [1] used Google's AudioSet for training and we used the IRMAS data. It's also worth noting that our model attempted to classify among significantly more categories than [1]. Nevertheless, it is promising that a similar neural network architecture yields similar results, even with different datasets.

After training the multi-class classifier, we obtained a 40% train set accuracy, a 30% validation set accuracy, and a 30% accuracy on the IRMAS testing data. This is noticeably lower than the accuracy we got when using the same pre-trained weights for the single-class classifier (these accuracies were computed using intersection over union, so it does make sense to compare the accuracies of the

multi-class and single-class models). Note that the IRMAS testing data contains between 1 and 5 instrument classes. We were surprised that the multi-class model performed equally well on the validation set and the test given that only the test set had multi-class labels, though we never trained on audio with multi-class labels.

	cel	cla	flu	gac	gel	org	pia	sax	tru	vio	voi
cel	29	1	3	12	4	0	11	4	0	8	0
cla	0	51	6	5	2	0	22	8	2	4	0
flu	2	10	40	8	3	6	23	0	0	0	0
gac	0	2	0	89	2	4	15	0	0	0	0
gel	0	2	2	23	68	8	36	0	2	3	0
org	0	0	2	5	7	88	36	2	2	2	0
pia	0	0	0	8	0	2	146	0	0	0	0
sax	6	4	10	10	2	0	16	59	6	4	1
tru	0	5	4	0	0	4	21	18	64	6	0
vio	14	5	9	8	12	0	11	10	1	62	0
voi	0	0	2	12	18	0	8	0	0	0	124

Confusion Matrix: The true instrument label is along the left-side and the predicted instrument label is along the top edge.

	cel	cla	flu	gac	gel	org	pia	sax	tru	vio	voi
Precision	0.57	0.64	0.51	0.49	0.58	0.79	0.42	0.58	0.83	0.7	0.99
Recall	0.4	0.51	0.43	0.79	0.47	0.61	0.94	0.5	0.52	0.47	0.76
F1-Score	0.47	0.57	0.47	0.61	0.52	0.69	0.58	0.54	0.64	0.56	0.86
Count	72	100	92	112	144	144	156	118	122	132	164

Metrics Table: The true instrument label is along the top edge, along with various per-instrument metrics.

Our model performed exceptionally well in distinguishing human voices from other instruments. Of all of the categories, the human voice had the highest precision (99%) and highest overall F1-Score. This could potentially be explained by the fact that the human voice is fundamentally different from most other kinds of instruments in terms of the amount and variety of timbres and articulations possible. On the other hand, our model was likely to conflate instruments of similar types. For instance, out of the examples labelled as violins, most of the incorrect selections came from cellos, electric guitars, and acoustic guitars (i.e. other stringed instruments). Perhaps even more apparent is that our model was most likely to conflate instruments that played in similar pitch ranges. This might explain why most instruments were most likely to be misidentified as a piano, since the piano has the widest range of all of the categories. Likewise, trumpets were most likely to be misidentified as saxophones, and flutes were likely to be misidentified as clarinets.

7 Conclusion

Despite working with more classes of instruments, our model achieved comparable performance to what we've seen in literature. Our efforts to reduce variance were successful across the board, bearing fruit with both classifiers. The mel-spectrogram representation of music provided sufficient features and information for our convolutional neural network fit to, and allowed our model to very accurately differentiate between musical instruments with very different timbres. The model had more trouble picking up on the differences between similar instruments, but potentially could be improved upon with additional data and/or time.

Perhaps the most surprising result is the performance of the multi-classifier and how well it generalized to the test set. The IRMAS train set contains examples of precisely 3 seconds in duration with 1 positive class, while the test set has examples with variable length and between 1 and 5 positive classes. We initially thought the multi-classifier would suffer in performance on the test set compared to the validation set. Instead, we achieved almost exactly the same accuracy on the validation set and the test set with the multi-classifier, indicating that our method of transfer learning was successful.

8 Contributions

The dataset parsing and preprocessing was done jointly between the two of us. Dominick was responsible for data aggregation and data augmentation techniques, as well as the associated code infrastructure. Connor was responsible for implementing the initial architecture setup and converting all of the audio clips into mel-spectrograms formats. Dominick and Connor jointly did a hyperparameter search to optimize the weights. Finally, Connor adapted the model to a multi-class classifier.

9 References

[1] https://cs230.stanford.edu/projects_fall_2019/reports/26225883.pdf

[2] J. D. Deng, C. Simmermacher and S. Cranefield, "A Study on Feature Analysis for Musical Instrument Classification," in IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 38, no. 2, pp. 429-438, April 2008, doi: 10.1109/TSMCB.2007.913394.

[3] Brown, Guy Cooke, Martin. (1994). Computational auditory scene analysis. Computer Speech and Language. 8. 297-336. 10.1006/csla.1994.1016.

[4] Park, Taejin, and Taejin Lee. "Musical instrument sound classification with deep convolutional neural network using feature fusion approach." arXiv preprint arXiv:1512.07370 (2015).

[5] Link to the github with all of our code: https://github.com/dhing1024/cs230-instrument-audio-ai