CS230

High Fidelity Song Identification via Audio Decomposition and Fingerprint Reconstruction by CNN and LSTM Networks

Simon Tao, Yosheb Getachew
Department of Computer Science
Stanford University
`stao18@stanford.edu, yoshebg@stanford.edu`

Abstract

The task of generic audio input identification is achieved through numerous robust techniques, one of which is audio source separation, which isolates a particular source from a confluence of audio samplings. While these techniques accomplish consistently high performance levels on manufactured audio, inexorable discrepancies arise when these algorithms are implemented to interpret human audio, due to its unpredictable composition and innate lack of regularity. Currently, softwares such as Shazam and other audio analysis algorithms have attained relatively successful results on processed audio, but improvements and developments of these mechanisms to recognize unfiltered human audio are absent from the field. Within this project we consider the identification of musical samplings, which arises in the familiar conundrum where an individual cannot distinguish the identity of a sampling of a song but would like to. We propose a mechanism for robust audio recognition and filtration, by programming a signal analysis and identification model that transforms distorted human audio into signal frequencies, then deploys a recurrent neural network LSTM unit to process the continuously sequenced frequency points and associated pitch points, subsequently applying softmax classification to identify the correct song.

1 Introduction

Computer science offers an innovative and efficient medium to expeditiously solving high density analysis problems that require high complexity computations or large breadth data searches. This resolves many contemporary problems, one of which can be abstractly generalized as input identification, where a computing system registers an input and decodes an associated output for it.

Audio input identification is a particular subdomain of input identification, which can manifest in many distinguishable applications, such as music recognition. We encounter this dilemma in the simplest of manners, such as when an individual hears a song that they vaguely recollect, or discovers a new melody that they are unfamiliar with but would like learn, but unfortunately metadata such as the title, genre, artist, and album are not accessible. This can amount in frustrations due to the elusiveness in determining the nature of the song that the previous impediments conceal, a sentiment that many empathize with. This problem has motivated many previous enterprises to construct an efficient and confident search engine that can facilitate immediate music recognition have culminated in the development of pitch estimation architectures and audio search services.

However, these structures do not necessarily encode components that accommodate all the practical idiosyncrasies that are to be anticipated when a human interfaces with a computational model. For example, slight deviations in pitch are common, and it is a rarity to replicate the intricate modulations in pitch that are characteristic in most musical compositions. Most individuals cannot sing in perfect pitch either, which may introduce error in the analysis of the recorded sample from a user. Computers may also misattribute a song if the incorrect cadence or tempo inputted by a human user, dependent upon the sequencing method of the architecture. Other disturbances, such as background interference, can corrupt the sampling itself [1], and an inexperienced software with implicitly homogeneous development may be susceptible to inaccuracy when translating audio of diverse accents and registers.

Therefore, to augment these preexisting systems and optimize the inevitable discrepancies, we introduce a novel mechanism that encodes the ability to adapt to differing human registers, accents, pronunciations, and other vocal distinctions by programming learning capabilities with heuristically defined hyperparameterizations within a deep learning neural network algorithm. This algorithm will be differentiable from comparable softwares such as Shazam by employing deep neural networks to achieve high fidelity interpretation of vocal audio that current computational models do not process extremely well. While the current conventional structure of audio translation accomplishes high performance when registering audio projected from equipment such as speakers with an abundance of parameters such as pitches, lyrics, etc., our model transforms distorted human audio into signal frequencies using input in the format of a waveform. We then deploy a recurrent neural network LSTM unit to process the continuously sequenced frequency points and associated pitch points, subsequently applying softmax classification to output the predicted identify the correct song.

2 Related work

As previously referenced, there exist other programming architectures within the literature that address the generic format of this problem, albeit with different specifications to address particular niches. One of these is Shazam, a popularized identification software. Shazam emphasizes recognition of music, television advertisements, movies, and other multimedia modes, by employing a query-by-example search service. However, this imposes limitations, as it necessitates assembling an enormous repository of music to refer to, which is a universal conundrum we acknowledge will manifest in our proposal as well. Shazam originally implemented temporally aligned combinatorial hashing to surmount sample distortion due to mobile phone recording resolution, but the software has now evolved and employs a variational recurrent neural network, which differs from the LSTM architecture we constructed, as an LSTM unit is capable of learning long-term dependencies[1].

Experimentation on the robustness of Shazam has elucidated that while "the existing solutions tackle variations in some properties such as background sound and white noise[...] the identification of samples containing large variations in key, tempo, ornamentation, and harmonization remains largely unsolved[4]." Using MIDI waveform inputs, Xiao demonstrated that the algorithm is not built to recognize and perform identification on musical structure. This weakness of Shazam is where we intend to utilize our model. While Shazam is the contemporary state-of-the-art recognition system, it cannot recignize unfiltered monophonic human audio, and we hope that a proof of concept of our methodology may inspire evolution of the model to remedy the current constraints.

Other architectures resolve elemental pieces of the problem, such as pitch estimation of a monophonic source. This problem is of concern since the human voice is categorized as monophonic, and this domain was largely neglected until the early 1990s, when methods of recognizing human pitch over a three-octave range were established and implemented in hardware and software configurations. These techniques surmounted existing procedures such as FFT and autocorrelation, by suggesting that the fundamental period measurement of an acoustic waveform can be exploited to achieve higher accuracy with more computational efficiency[2]. Using filtering banks attuned to extract the fundamental frequency, Kuhn exhibited that the input waveform can easily be transformed to a sinusoidal form, where the period of the transformed sinusoid is associated to the pitch of the original input waveform. This approach was extremely clever, as it required drastically less dense algorithms while conserving essential computing power. The methodology was unprecedented at the time, which enabled future innovations to originate from this concept.

This idea was advanced in 2018 by a group at NYU, who utilized this insight to engineer the CREPE architecture, which invokes a deep convolutional neural network operating on the time-domain waveform input, which according to the results, "obtains state-of-the-art results, outperforming heuristic approaches[...][and is] more robust to noise too[6]." We utilize this structure as a foundational basis of our algorithm, by integrating the pitch estimation architecture CREPE with our waveform sequencing technique, modeled after DNA sequencing.

3 Dataset and Features

We procured multiple datasets as well as generating our own throughout the progression of our project, as was necessitated by our advancements within each milestone. We used two training sets, the first of which to test our implementation of the preceding architectures CREPE. In accordance with the recommendation by the documentation on CREPE, we installed the MIR-1K dataset with one thousand training examples to confirm that our preliminary scaffolding was correct. This validated the CREPE extraction and sequencing of pitch estimations, which we then transformed into frequency sequencing using complementary preprocessing functions.

For the secondary milestone, we compiled our own dataset by collecting 10 popular musical pieces and converting them into audio waveform format, with a frequency list mapping across the time-domain. The total duration of the dataset audio is 41 min, with a data size of 442MB. The dataset was compiled with a heuristic argument of achieving robust data variation, by selecting songs produced by artists with varying characteristic registers and acoustic styles, so as to maximize the networks familiarity with a large spectrum of musical input. This dataset was implemented as our training dataset for the first iteration of our algorithm. The 10 waveform examples were discretized after data augmentation into 10 second intervals, as a hyperparameter we tuned to optimize the estimation capabilities of the CREPE pitch tracking component of the model. We subsequently ran the model on the approximately 246 examples partitioned from the original 10 songs, with promising results.

We then developed our own validation set with 10 examples, by attempting to record approximately 10-30 second increments of our own attempts to sing the songs from the aforementioned training datasets. These audio samples were acquired via recording from a mobile device, in an attempt to address the low resolution mobile device recording issue Shazam also confronted. After uploading the compiled dataset, we performed data augmentation to segment the audio waveforms into 10 second increments with an offsetting of 3 seconds to distinguish the validation intervals from the training intervals. This produced 19 validation examples.

The test set was composed of 2 second duration segments of the recording from the upload implemented in the validation set. Our model was only executed on 10 percent of these partitions, and thus the test set contained 9 examples.

4 Methods

To achieve our goal, our project was divided into two major components. The first component consisted of converting all wavefiles into lists of musical pitches, succeeded by training a Recurrent Neural Network on these sequences of pitches to learn parameters that distinguish one song from another.

To simplify the enormity of these ambitions, we begin with the assumption that the user has perfect pitch. As the first half of the model executes a frequency-to-pitch translation, our assumption permits us to prove that the model works before we extend to analyzing relative pitch differences (to compensate for relative pitch).

We initialized our deep network by importing the GitHub repository containing the CREPE convolutional neural network environment, as well as importing miscellaneous Python modules to perform mathematical operations and complement analysis with data visualization. Afterwards, the dataset is introduced to the framework and parsed as to access its various features distinctly, such as filenames and waveforms. We subsequently develop an architecture basis by programming the waveform transformation functions that map generated frequency data points to the pitch domain.

To perform the frequency-to-pitch conversion, we first run CREPE, which translates raw audio waveforms into numerical frequencies, from which a simple formula converts the frequencies to pitches. Using the standard A4 = 440Hz model, we obtained the lowest note, C0 = A4 $* \ 2^{-4.75} = 16.35$ Hz. From there, dependent on the tuning of musical notes, we can find the pitch correlated to a given frequency by the following formula: pitch = $log_2(\frac{frequency}{C_0})$. For our training and test sets, all of the waveforms and pitches are calculated on 10-ms increments.

After getting the list of pitches for each song, our next goal was to build the Recurrent Neural Network that performs an NLP task. This portion is similar to DNA sequencing, as we are trying to train a model that can recognize some sequence of pitches correspond to some song. As there was relatively little data on acapella performances for the pop songs, we performed data augmentation to acquire more clips for each of our songs, as explained above. One of the hyperparameters we played around with was how long should each snippet be, as the longer the snippets are, the closer it can resemble the actual song, but we also were wary of the model overfitting the data if the snippets were too long as, realistically, people usually won't record themselves for too long when they try to identify a song. We initially tested our model using 5-second clips, but as we played around, we settled for the snippet length to be 10 seconds as it yielded higher test accuracy, and any snippets that didn't fill 10 seconds were padded with 0's.

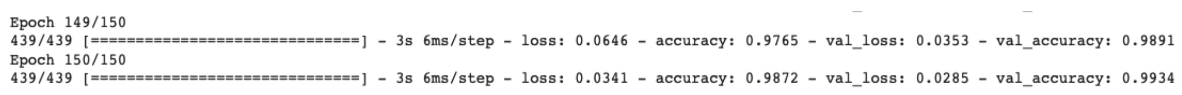
For our model, we used Keras to build an RNN to learn the parameters. The clips are first sent through an Embedding layer of size 12, as there are only 12 unique pitches in a standard musical scale. The input length was set to be the number of seconds each clip is. From there, we ran the clips through an LSTM layer, along with a Dropout layer of 0.5, in order to find time-dependent patterns that exist within songs. The model then passes the clips through two Dense layers with relu activation in order to recognize more local sequential patterns, and finally an output Dense layer that performs softmax activation to output a one-hot vector that corresponds to which song index it thinks song is. As we only used 10 different songs currently, the output dimension is 10x1. The loss function used was the cross-entropy loss function, where $L = -\Sigma_{c=1}^{M}(y_c \log \hat{y_c})$, where M is the number of different songs, which represent the number of classes. All in all, using this model, 583,594 parameters were trained.

5 Experiments/Results/Discussion

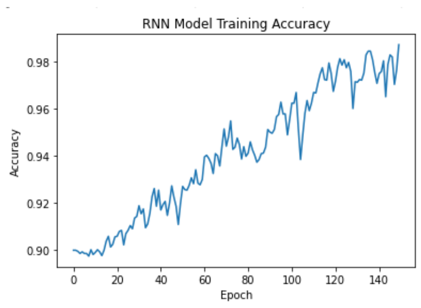
The adjustable hyperparameters we selected were represented by the duration of segmented dataset examples, the temporal translation of the segmented examples from their start in the validation set to ensure variability from the training set, and the dropout probability within the LSTM unit. We converged upon the numerical value of 10 seconds for the training and validation set example interval length, after originally encoding 5 seconds as the length. After multiple iterations of optimizing the algorithm we found that a 10 second duration maximized the accuracy in conjunction with the supplementally optimized hyperparameters. Similarly, the test set segment duration was determined after attempting to execute the model on 2 second example partitions, then on 4 second examples, which exhibited the highest confidence. We initially established the offset of the validation examples from the beginning of the uploaded unsegmented dataset examples to be 3 seconds, which demonstrated consistently superior performance, so we did not modify this hyperparameter. The dropout probability was standardized at 50 percent, in concordance with convention in the field.

The primary metric of our project was accuracy, as the objective of the model was to optimize identification accuracy of an input audio waveform. Our accuracy was computed via the binary cross-entropy loss function, as listed previous section.

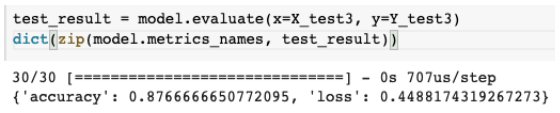
The 246 example dataset training accuracy was approxiamtely 99 percent, depicted below.



```
Epoch 149/150
439/439 [==============================] - 3s 6ms/step - loss: 0.0646 - accuracy: 0.9765 - val_loss: 0.0353 - val_accuracy: 0.9891
Epoch 150/150
439/439 [==============================] - 3s 6ms/step - loss: 0.0341 - accuracy: 0.9872 - val_loss: 0.0285 - val_accuracy: 0.9934
```



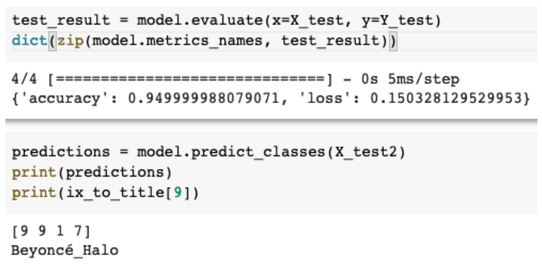
We observed high accuracy in the training set and validation set model summaries of the self-generated datasets, with a nontrivial decrement of approximately 10 percent accuracy from the training and validation set accuracy of 98 percent to 88 percent. From this we surmised that the algorithm was overfitting on the training and validation sequences



```
test_result = model.evaluate(x=X_test3, y=Y_test3)
dict(zip(model.metrics_names, test_result))

30/30 [==============================] - 0s 707us/step
{'accuracy': 0.8766666650772095, 'loss': 0.4488174319267273}
```

To introduce more variation to the model to ensure rigorous network learning, we assembled a dataset from an external party with a significantly differing register, acoustic range, and pitch capability as the subject of our validation and test datasets, and evaluated their attempt at replicating the song Halo by Beyonce, which the model successfully interpreted.



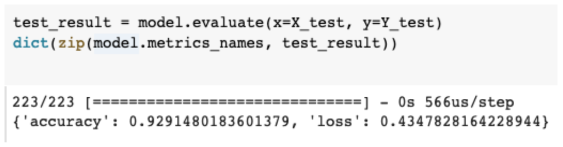
```
test_result = model.evaluate(x=X_test, y=Y_test)
dict(zip(model.metrics_names, test_result))

4/4 [==============================] - 0s 5ms/step
{'accuracy': 0.949999988079071, 'loss': 0.150328129529953}

predictions = model.predict_classes(X_test2)
print(predictions)
print(ix_to_title[9])

[9 9 1 7]
Beyoncé_Halo
```

One observed flaw when testing the 5 second segment model with a manually recorded sampling of Beyonce's Halo was the inordinately high probability attributed to the identity of the song as Ben E. King's Stand By Me. Both songs are performed in the A major key, which indicates there is a tonal congruence between the two pieces. We hypothesizes this behavior of the model was due to underfitting, so to resolve this issue we extended the clip lengths to 10 seconds, to enable longer sequence learning by the model for distinct songs and optimistically reduce overlap between similar songs. Increasing the duration of the intervals during data augmentation resulted in higher training and validation accuracy, stabilizing around 98 percent. However, test accuracy declined to aproximately 93 percent. With this implementation, the model was able to predict Beyonce's Halo with greater fidelity than Stand By Me. The cumulative model performed substantially well, attaining an accuracy of approximately 90 percent.



```
test_result = model.evaluate(x=X_test, y=Y_test)
dict(zip(model.metrics_names, test_result))

223/223 [==============================] - 0s 566us/step
{'accuracy': 0.9291480183601379, 'loss': 0.4347828164228944}
```

6 Conclusion/Future Work

In conclusion, the model achieved our objective of audio sample identification using a recurrent neural network architecture with high fidelity and learning capabilities, equipping it to interpret diverse variations of audio input. The LSTM algorithm performed with impressive results, fulfilling our initial goals of engineering a proof of concept of the principles elaborted upon in this paper.

This project has many potential future developments in order to make it more accessable. For starters, implementing an algorithm that analyzes songs by the relative distances of the pitches, rather than the absolute pitch, would greatly enhance the scope of feasible users, as not everyone would be singing on pitch. Another potential idea to implement that would simply be to scale down all songs to the same musical key, which would enable us to analyze the sequences easier without needing to account for relative distances between pitches, but this might lead to confounding effects given the similar patterns across many pop songs. Furthermore, we can potentially also try to take lyrics into account for other genres such as rap, where melody cannot be reliably used as the distinguishing factor to differentiate between songs, but rather the rhythm and words. Of course, as our current database only consists of 10 unique songs, we also would need to gather a lot more songs in order for our algorithm to become more powerful. Overall, nevertheless, our current model stands as a starting point to a new method of music recognition, and could be expanded upon greatly in the future.

7 Contributions

Most of the project was split evenly between research and code between the two of us. Yosh did a lot of initial research and discovered the crepe model (as well as its advantages and disadvantages), while Simon figured out how to implement crepe into our algorithm. Most of the programming and model building was carried out by Simon, while Yosh carried out extensive research into the literature and theory behind the models , as well as gathering the training and test data which we used to test our model. We both contributed to all of the writeups, and overall, it was an amazing collaboration.

References

[1] Avery Wang. 2006. The Shazam music recognition service. Commun. ACM 49, 8 (August 2006), 44–48. DOI:https://doi.org/10.1145/1145287.1145312

[2] Kuhn, William B. “A Real-Time Pitch Recognition Algorithm for Music Applications.” Computer Music Journal, vol. 14, no. 3, 1990, pp. 60–71. JSTOR, www.jstor.org/stable/3679960. Accessed 24 Apr. 2020.

[3] Zhang, Bin, Changqin Quan, and Fuji Ren. "Study on CNN in the recognition of emotion in audio and images." 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS). IEEE, 2016.

[4] Xiao, Fangjian Flora. Experiments with the Shazam music identification algorithm. Diss. 2018.

[5] T. Gagnon, S. Larouche and R. Lefebvre, "A neural network approach for preclassification in musical chords recognition," The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003, Pacific Grove, CA, USA, 2003, pp. 2106-2109 Vol.2.

[6] J. W. Kim, J. Salamon, P. Li and J. P. Bello, "Crepe: A Convolutional Representation for Pitch Estimation," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, 2018, pp. 161-165, doi: 10.1109/ICASSP.2018.8461329.

[7] MIR-1K Public Dataset. http://mirlab.org/dataSet/public/.

[8] Emilia Gomez, Merlijn Blaauw, Jordi Bonada, Pritish Chandna, Helena Cuesta, "Deep Learning for Singing Processing: Achievements, Challenges and Impact on Singers and Listeners," 2018. https://arxiv.org/pdf/1807.03046.pdf

[9]Zain Nasrullah, Yue Zhao, "Music Artist Classification with Convolutional Recurrent Neural Networks," 2019 July, doi:10.1109/IJCNN.2019.8851988.

[10] Alipanahi, Babak, et al. "Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning." Nature biotechnology 33.8 (2015): 831-838.

[11] Quang, Daniel, and Xiaohui Xie. "DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences." Nucleic acids research 44.11 (2016): e107-e107.

Libraries

[1] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830.

[2] Chollet, F., others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[3] Kim, Jong Wook, et al. "CREPE: A convolutional representation for pitch estimation." 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018.

[4] Hunter, John D. "Matplotlib: A 2D graphics environment." Computing in science engineering 9.3 (2007): 90-95.

[5] Virtanen, Pauli, et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python." Nature methods 17.3 (2020): 261-272.

[6] McKinney, Wes. "pandas: a foundational Python library for data analysis and statistics." Python for High Performance and Scientific Computing 14.9 (2011).

[7] Abadi, Martín, et al. "Tensorflow: A system for large-scale machine learning." 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016.