The Lick Machine
CNN Audio Pattern Recognition

Final Report

Maddy Walkington (260 986 638)
Louis Bouchard (261 053 689)
Simon Pino-Buisson (261 051 516)
Simon Lavoie (261 051 325)

December 15th, 2024

Abstract

A convolutional neural network (CNN) and hybrid convolutional/recurrent neural network (CNN + RNN) were trained to recognize a distinctive 5-second melody common in jazz improvisation called *the lick*. By augmenting our dataset to offer higher variance in the melody's key, tempo and timbre, then training the models on the resulting chromagram and Mel-Spectrogram (as well as its derivatives), the CNN was able to achieve a training accuracy of 97.3% and testing accuracy of 91.4% on audio files containing single isolated instruments, while achieving 93.6% and 85.9% respective accuracies on audio files containing actual song recordings (which comprised of multiple instruments and more complex audio signals). The hybrid CNN + RNN model achieved 99.6% training accuracy and 94.3% testing accuracy on the single-instrument files, as well as 96.6% training accuracy and 88.9% testing accuracy on the files containing actual song recordings. These accuracies indicate that both models were successful in recognizing *the lick* despite being limited by a relatively small dataset (1132 audio files).

1 Introduction

A popular inside-joke within jazz music circles centers around *the lick*. The *lick* is a short musical passage which is often played during improvisation. The phenomenon began with a simple observation: "I've heard this passage before", and became a challenge: "How can I sneak *the lick* into this solo?". Today, when *the lick* is played during a set, the neurons of those in the know immediately fire, while going under the radar as a regular melody to those who aren't.

This project presents a comparison between a convolutional neural network (hereby referred to as a CNN) and a hybrid convolutional-recurrent neural network (CNN-RNN) trained to recognize this motif in a piece of music, making it also part of this inside joke. For the model to succeed in recognizing *the lick* in an audio sample, it would have to be invariant to the music's key, tempo, timbre, volume (see Appendix A) and accompaniment of additional instruments.

2 Literature Review

2.1 Music in Machine Learning

As recent advancements in machine learning have evolved technology's ability to model complex datasets, the potential uses of such models have skyrocketed. Several fields were able to take advantage of this newfound technology and adapt it for their own purposes, such as musicology (the broad study of music); machine learning models could help with copyright detection, automate music sheet creation from sound files, improve music recommendations, build advanced search tools for music libraries, and more!

For example, models trained for genre/mood classification of audio data can help automate an important aspect of music consumption in the digital space (on apps such as Spotify and Apple Music). Recently, Rani et al. (2022) [1] performed mood classification of music by converting the audio into its time-frequency spectrum and using a convolutional neural network to analyze the resulting image, with a whopping 90% accuracy.

Additionally, Anand et al. (2019) [2] aimed to identify the underlying *raga* in a musical piece. In Indian classical music, the *raga* is a sequence of notes that forms the melodic framework for the piece. This motif defines the structure on which any additional improvisation is based. A convolutional neural network was trained to identify the *raga* based off the variation in pitch throughout the piece, achieving 96.7% and 85.6% accuracy on two different models. Since this paper involved identifying melodic sequences which serve as the substructure for improvisation in the piece, it is very relevant to

our own project to understand their methodology. Jazz is a notoriously improvisational genre, where melodies such as *the lick* serve as the motif for a musician's spontaneous choices. Our model will have to detect such a pattern hidden under layers of additional melodies and improvisational flourishes.

Minkyu et al. (2018) [3] outline the standard procedure for vectorizing audio samples. Namely, by normalizing the audio file's sample rate to a constant 16 kHz and performing a short-time Fourier Transform (STFT) for each sample, thus creating a spectrogram which yields a time-frequency representation of the audio. It is also preferred to convert the spectrograms into mel-scale spectrograms, which rescale the frequency axis to be more representative of the frequency scale of human hearing, as shown in Doshi et al. (2021) [4]. Since music is played on instruments designed to make sounds in the scale of human hearing, these pre-processing techniques will allow the desired features (notes played) to be emphasized.

2.2 CNNs versus Transformers

With the help of the previously mentioned articles, it has been established that both transformers and CNNs are well-suited for the classification of audio samples. A thorough audio classification survey carried out by Zaman et al. (2023) [5], which compared the performance of over 100 models spanning eight different implementation methods on various different audio datasets further demonstrates this point. Within the articles in the paper, a variety of transformers and CNNs, as well as some RNN models, were tasked with classifying five-second audio samples of ambient sounds (using for example the ESC-50 dataset), which closely resembles our task of identifying a particular melodic pattern within a musical piece. It was found that both CNNs and transformers could perform extremely well; both types of models, as well as the CNN-RNN hybrid models, were often able to surpass the 90% accuracy threshold. Upon further investigation of the matter, Xu et al. (2021) [6] found that transformers require a bigger dataset than CNNs to achieve similar accuracy, which can be mitigated by using techniques out of the scope of this project. Moreover, CNNs are known for their abilities to detect local patterns with a high degree of precision. Given that *the lick* is a motif that generally lasts between one and two seconds (much smaller than the samples of five seconds in the ESC-50 dataset) we believe that the long-range global context detection that transformers offer would be of lesser importance. Considering this new information, as well as the comparable performance of CNNs and hybrid CNN-RNN models outlined in Zaman et al. (2023) [5], this project opted to make use of these two types of models rather than transformers.

3 Methodology

3.1 Data Formatting

Andy Chamberlain and Tyler Weir's The Lick dataset [7] was used to train this model. This dataset comprises 18 thousand unique audio snippets, 1132 of which were found to suit the project's needs and were used as input data. Roughly half of them contain examples of *the lick* with varying key, tempo, timbre, volume and accompaniments (which helps increase the model's adaptability), while the other half contains a different melody to help train the model to have falsifiable results. Within these two categories, there are two subcategories: Solo (single midi instruments playing in isolation) and External (music samples from actual concert/song recordings).

It was found that 99% of the audio files were under 5 seconds in length, so each audio file was either padded or trimmed to 5 seconds to ensure consistency in the data.

We used various methods of data augmentation on the input files to ensure a robust and flexible model. Despite the uniqueness of each audio file, the snippets of music they contained were, for the

most part, generated using very similar midi sounds. This caused issues when initially training the model as the original dataset did not present a large enough variance in timbre, tempo and key to train a flexible model. To prevent overfitting and help the model generalize well to more complex audio inputs, such as actual song recordings, the 1132 input files were transformed to offer a higher data variance; they were randomly pitch-shifted by 0 to 5 semitones, their tempo was randomly stretched, and pink noise (white noise with a $1/frequency$ filter, which means larger amplitudes in the lower frequencies) was added. Once these transformations were applied, the audio files were vectorized and a random translation of 0 to 10% was then applied to further increase the data's variance. Thus, the models' performances were quickly improved after training on this augmented dataset as they were more invariant to properties such as key, tempo, and timbre.

The audio samples from the dataset were vectorized using the *python* library Librosa [8] to extract each file's Mel-Spectrogram, which is a visualization of the audio's frequency density over time, and chromagram, which is a similar visualization of the audio's pitch class density over time (see Appendix C). These two spectra, as well as the Mel-Spectrogram's two first derivatives(its Δ and $\Delta\Delta$ plots), give the model 4 greyscale images through which it can interpret each audio files (see Figure 1).

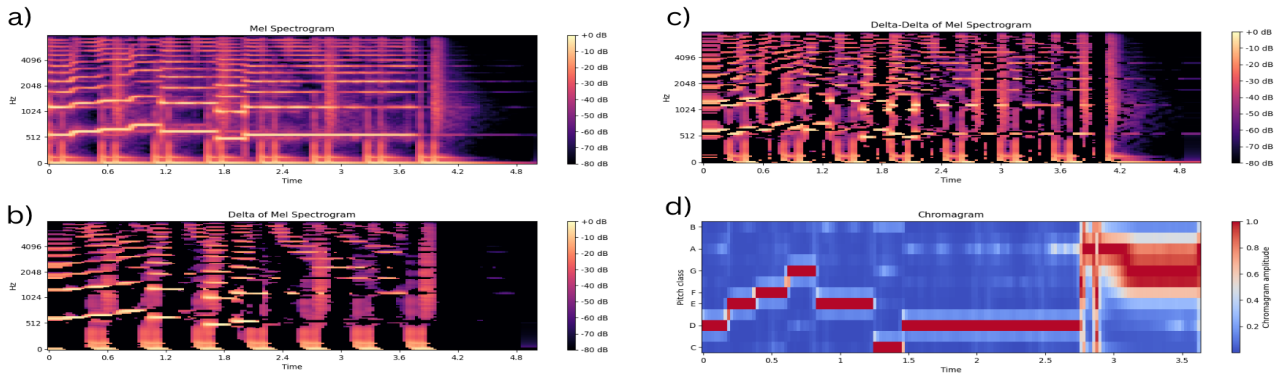

Figure 1: Mel-Spectrogram (a) for a sample audio file containing *the lick*, as well as its Δ spectrogram (b), $\Delta\Delta$ spectrogram (c), and chromagram (c).

3.2 Model Architectures

In the interest of evaluating the abilities of different types of machine learning models, we created a CNN and a hybrid CNN + RNN model to compare their respective performance. Both models have similar architecture, the only difference being that the latter has an additional recursive layer between the convolutional and fully connected layers. This recursive layer is an LSTM (Long Short-Term Memory) type of recurrent neural network, which excel at recognizing dependencies in sequential data over several time-steps and mitigating the vanishinig gradient problem [9]. Since *the lick* is a sequential pattern of variable length, we expected the hybrid model to make use of these memory-based skills and perform better at processing and recognizing this melody than the simple CNN model.

The rest of the architecture, shown in Figure 2, is composed of five convolutional layers, each separated by RELU activation and a batch normalization layer. Four of these have a pooling layer and three of them have a 10% dropout rate. The hybrid model then includes the recursive LSTM layer of depth 2. Following these layers, which perform the feature extraction, three fully connected layers perfom the classification task. Since this is a simple case of binary classification (*lick/no lick*), the output layer has size 1 to which we apply a sigmoid function.

During the training of both models, binary cross-entropy quantified the loss and ADAM was used as an optimizer during gradient descent, with a learning rate of 10^{-4}, $\beta = 0.9$ and 0.999, and L2

regularization of 10^{-3} to mitigate overfitting. A learning rate scheduler was also used to reduce its value as improvment stagnated.

Layer	Output channels	Kernel size	Specifications
Conv	24	9	Padding = 5
MaxPool	24	2	Stride = 2
Dropout			10%
Conv	48	3	
AvgPool	48	2	
Conv	82	3	
AvgPool	82	2	
Dropout			10%
Conv	164	3	
AvgPool	82	2	
Dropout			10%
Conv	132	3	
LSTM	3168		Depth = 2
FCL	256		
FCL	164		
Dropout			10%
FCL	1		

Figure 2: Visual representation of the hybrid CNN + RNN architecture layers and their specifications.

4 Results

After training both models for 50 epochs on batches of size 64 of the mixed solo and external audio samples, we tested their accuracy on the solo and external audio samples separately (see Table 1).

Test	CNN	CNN + RNN (Hybrid)
Solo Train	97.3	99.6
Solo Test	91.4	94.3
External Train	93.6	96.6
External Test	85.9	88.9

Table 1: Comparison of the accuracy of both models when training on the solo + external dataset

5 Discussion

5.1 Interpretation of the Results

As we can see from the results in Table 1, both models managed to successfully classify more than 85% of the data set. The solo data was always better classified; this is to be expected since these audio files contained very clear notes played sequentially from a single instrument, rather than multiple

instruments playing at the same time. The models would obviously have an easier time recognizing *the lick* without have multiple other melodies playing over it. This would also explain why both models generalize better with these solo files than with the external data when measuring their testing accuracy.

Furthermore, the hybrid model performed slightly better than the CNN model, but took around three times longer to train. Its higher accuracy is not surprising when considering its sequential pattern recognition skills (as explained in Section 3.2).

In addition, the combination of both data types in one dataset during the training of the models greatly improved the generalization potential at the cost of slightly reducing the accuracy from training on only one type. Training on them separately made both models very good at classifying either solo or external, but not both simultaneously (see Appendix B).

5.2 Further Improvements

The main limitation of this project lies within the small size of the dataset, especially regarding the positive external data points. In order to get enough data to accurately classify real music audio samples, we were forced to reduce the amount of data points in the other categories (solo positive and negative, and negative external) so that they would have comparable sizes and not skew the model. This led to the entire training dataset having less than 1000 data points. Both models perform remarkably well despite this limitation, but a larger amount of data would help the models train more robustly and generalize better during testing to significantly improve their performance.

Apart from increasing the raw dataset size, the quality and size of the available data could be further improved by utilizing more advanced data augmentation techniques. To offer a higher variance and prevent overfitting, certain transformations were already used (as explained in Section 3.1). However, other techniques, such as frequency filters/masks, gain adjustments, and certain effects like reverb or distortion, could help us attain a more robust model.

Both models could also be improved by performing more refined hyperparamter tuning. A grid search to find the optimal values for such parameters, such as the learning rate or the regularization weight decay, could greatly increase the performance of the model without compromising its balance between expressivity and ability to ogeneralize

Although it already does very well, especially considering the limited amount of data, our model could benefit from a few modifications. We know from the training on each dataset that the models are already expressive enough to classify the data points. However, a grid search to find the optimal values for hyperparameters such as the learning rate, the betas or the regularization weight decay.

6 Conclusion

Finally, we can say the goal was completed, since our model managed to accurately classify close to 90% of real music samples. With more real music audio samples of *The lick*, we could hope for an accuracy closer to 90%-95% like for the models trained on the isolated sets. Although, this may seam like a silly idea, a project like this acts as a stepping stone for greater models that could help with copyright issues, music recommendation or even music generation.

7 Reflection

7.1 Contributions

Most of the model architecture and data processing sequence was created by Louis Bouchard and Simon Lavoie. Simon Lavoie also edited the video presentation associated with this project. Maddy Walkington and Simon Pino-Buisson mostly worked on research and report redaction. All work was reviewed and discussed with the entire team.

7.2 Self-Assessment

Working on this project allowed us to explore areas that had not been fully visited during class, which gave us a better appreciation and understanding of the material. It was a fulfilling experiment that we are proud of. We are more than pleased with the results, especially considering the dataset size issues we encountered. Given more time and resources, we would have expanded our dataset and explored different model architectures, such as transformers and maybe auto encoders. A repository of the source code can be found here.

References

[1] S. Rani, M. Kaushik, and V. Yadav, "Identifying mood in music using deep learning," in *Artificial Intelligence and Technologies*, R. R. Raje, F. Hussain, and R. J. Kannan, Eds. Singapore: Springer Singapore, 2022, pp. 571–578.

[2] A. Anand, "Raga identification using convolutional neural network," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, 2019, pp. 1–6.

[3] D. P. H. K. Y. O. J. P. J.-S. J. G.-J. Lim Minkyu, Lee, "Convolutional neural network based audio event classification," *KoreaScience*, 2018.

[4] K. Doshi, "Audio deep learning made simple - why mel spectrograms perform better," https://ketanhdoshi.github.io/Audio-Mel/, Feb 2021, accessed: Nov 22, 2024.

[5] K. Zaman, M. Sah, C. Direkoglu, and M. Unoki, "A survey of audio classification using deep learning," *IEEE Access*, vol. 11, pp. 106 620–106 649, 2023.

[6] P. Xu, D. Kumar, W. Yang, W. Zi, K. Tang, C. Huang, J. C. K. Cheung, S. J. D. Prince, and Y. Cao, "Optimizing deeper transformers on small datasets," 2021. [Online]. Available: https://arxiv.org/abs/2012.15355

[7] A. Chamberlain, "The lick," 2022, accesed: 2024-11-05. [Online]. Available: https://www.kaggle.com/datasets/andychamberlain/the-lick/data

[8] B. M. et al., "Librosa documentation," 2024, accessed: 2024-11-05. [Online]. Available: https://librosa.org/doc/latest/index.html

[9] GeeksforGeeks, "What is lstm - long short term memory?" Jun 2024. [Online]. Available: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

[10] D. O'Shaughnessy, *Speech Communication: Human and Machine*. IEEE, 2000.

A Music Terminology

- **Key**: This is probably the most familiar term to those not familiar with music terminology. The key of a piece of music is its tonal center. The note that "sounds like home". Analogously to a coordinate system, given some arbitrary origin, changing the key for a melody would preserve the relative distances between its notes, while moving the ensemble to or away from the origin. The model must recognize *the lick* by the relative distances between its notes, regardless of the key.
- **Tempo**: This is simply the speed at which the music is being played. Simply put, the model should recognize *the lick* regardless of how fast or slow it is being played.
- **Timbre**: This term refers to the overall "sound quality" of a piece of music. This could encompass the notion of high-definition/poor quality when talking about compression of sound files, but is a more general term that encompasses how music sounds. Pianos produce a different sound than trombones do, even if they play the same note, at the same pitch, and the same loudness. In terms of sound data, timbre could be viewed as the fine-grain details within the waveform that aren't its amplitude or frequency. The model should recognize *the lick* regardless of the instrument playing it.
- **Volume**: This is simply stating that our model should recognize *the lick* regardless of how loud or quiet the lick is being played. In terms of the data, this means it must be invariant to waveform amplitude.

B Complementary results

	Solo Dataset		External Dataset	
Test	CNN	CNN + RNN (Hybrid)	CNN	CNN + RNN (Hybrid)
Solo Train	100	99.8	56.3	59.1
Solo Test	95.7	97.9	52.1	58.6
External Train	45.3	56.2	97.8	96.5
External Test	50.4	58.4	89.3	88.2

Table 2: Comparison of the results of both models when training on the solo and external datasets separately

C Mel-Spectrogram

A Mel-Spectrogram is a visual representation of the frequency versus time density plot of the audio. This is produced by performing a sequence of Short-Time Fourrier Transforms (STFT) on equal-length time window divisions of the whole sample. The result is a spectrogram, which is then rescaled to the Mel-scale, which emphasizes the frequency range of human hearing. This is done by converting the frequency f, in units of Hz, into a "mel" m by the formula [10]

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

Valuable information about variations in the audio over time can still be hidden within the Mel-Spectrogram, so these are extracted by analyzing the differences in amplitude of the frequencies between two consecutive time frames (abbreviated as Δ). Thus, for some time frame t and some frequency f_t, its delta is given by

$$\Delta_t = A(f_t) - A(f_{t-1})$$

The second difference, known as the $\Delta\Delta$ is given by

$$\Delta\Delta_t = \Delta_t - \Delta_{t-1}$$

These provide computationally trivial methods of approximating the first and second derivative of a signal, which could help identify note changes when trying to recognize *the lick*.