Beatomizer: Using Deep Learning to Turn Songs into Satisfying Visualizations

Jack Michaels
jackfm@stanford.edu

December 2022

1 Introduction

1.1 Description

We enjoy listening to songs in the everyday life as frequently if not more frequent than we eat. It is ingrained within us to automatically put on study music while working or listen to an exercise playlist at the gym. While we love to discover and digest new songs on the daily, we hardly question the medium through which we consume said content. Henceforce, Beatomizer was born to enhance our consumption of music. Beatomizer is a deep learning model which takes in audio and outputs a visualizing satisfying light pattern in RGB. Such a model has the immediately obvious applications in clubs or with DJs, yet it could also enhance any audio experience across the board. By pairing the RGB output from Beatomizer with venue lights, room lights, or any visual application you can fully immerse yourself in the music.

1.2 Challenges

Creating a visually **satisfying** light show is challenging for a variety of reasons. The key word being satisfying, artificial intelligence is known to lack creativity and thus will display loss minimizing, yet aesthetically boring visualizations. The sheer size of the output space is massive, allowing the algorithm to hide behind loss minimization without creating anything solidly representative of the inputted song. This can be avoided through adversarial networks like GANs [1], yet that is above the scope of this project. Furthermore, the dataset requires heavy preprocessing in order to link frames to their corresponding audio segments. This is compounded by the fact that most videos have different frame rates, all of which must be normalized to the same value. For Beatomizer, an ideal frame rate of 60 fps was chosen. Finally, due to the high complexity associated with RNN models and the large input and output space of the model, running the attention model requires significant computational resources.

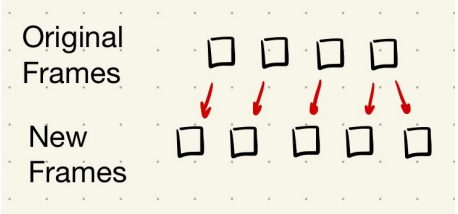

Figure 1: We can see some of the complexities associated with upscaling frame rate above. To keep time, we have to determine which frames must be duplicated and which mustn't be. We do so by decomposing into cycles, analogous to a Fourier Transform.

1.3 Related Work

Beatomizer can be considered a machine translation problem except instead of taking word embeddings and mapping them to one hot word vectors, we take slices of the audio and map them to RGB values, effectively "translating" audio into video. Machine translation between languages has been well documented [2] [3] [4] with sequential models as the primary candidate for results. Specifically, Beatomizer follows the attention model architecture posed in [2] due to the high time dependency of audio and visual data. SUFFERED FROM VANISHING GRADIENT

2 Dataset

The dataset was created using a Python scraper which ran over a human compiled and quality checked list of light shows. This created around four hundred 15 second clips which ultimately made up the dataset used in Beatomizer. Due to the unvectorized and inconsistent representation of video, heavy data preparation was required. We can see such augmentation here:

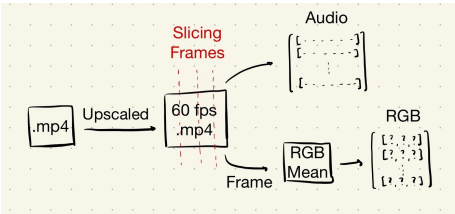

Figure 2: An image of the process Youtube .mp4 data takes to be model ready

Upsampling is immediately required to bring all videos up to a homogeneous level of content. For a sampled frame, the average of all pixel values was then taken for our label. Videos were then sliced into 15 second segments to ease the complexity of the model (15 seconds of video still corresponds to the massive input dimension of 900). Lastly, while a range of song genres were selected, songs with a pronounced beat were preferred as they would help the model learn to flash on the beat.

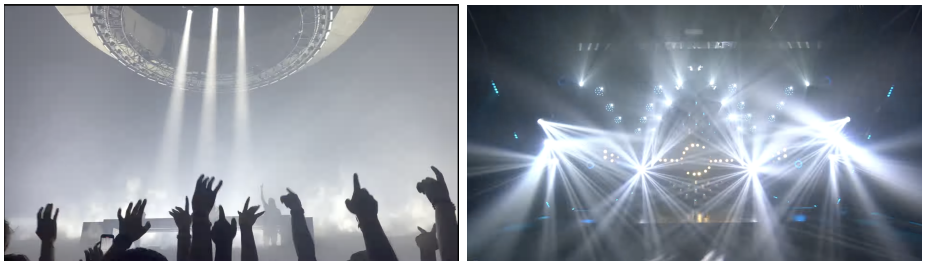

Figure 3: Examples of frames from the dataset

3 Metrics

To parse the correct RGB values, Beatomizer distills down to a regression problem and thus evaluation metrics such as "accuracy" have little meaning. As is common in many machine translation tasks [3], comparing two vectors can be accomplished through cosine similarity. Cosine similarity was used as the evaluation metric for both the baseline and attention model.
In regards to the loss function, while cosine similarity is great at instantaneously comparing two vectors, light shows are heavily dependent on the beat. Therefore, invoking a loss function which reflects not only the instantaneous discrepancy between labels but the discrepancy through time would be ideal.

4 Learning Methodology

4.1 Baseline

Considering how the data was formatted (for each desired frame there is an vector representing the corresponding interval of audio), we can imagine this input as a one dimensional image and apply a convolutional neural network to the problem. This was initially prompted due because of the high dimensionality of each input vector (roughly 350 distinct numbers). Though certain embedding frameworks use a comparable number of units [5], convolutional networks provide the additional feature that they are good at distinguishing between boundaries in images. Hypothetically, if the CNN baseline could distinguish the boundaries between beats, it could feed that information to the model thus improving accuracy. The architecture is as follows:

$$\textbf{Conv1D} \rightarrow \textbf{MaxPool} \rightarrow \textbf{Conv1D} \rightarrow \textbf{MaxPool} \rightarrow \textbf{Conv1D} \rightarrow \textbf{MaxPool} \rightarrow \textbf{Conv1D} \rightarrow \textbf{MaxPool} \rightarrow \textbf{Flatten} \rightarrow \textbf{Dense} \rightarrow \textbf{Dense} \rightarrow \textbf{Dense} \rightarrow \textbf{Dense}$$

with the idea of refining possible beat detections using the CNNs and then extrapolating those to RGB values with the dense layers.

4.2 Attention Model

While the baseline architecture utilizes CNNs to potentially parse beat patterns, it completely fails at predicting large scale time dependencies. To counteract this, methodologies mentioned in [2] are used. Beatomizer's more powerful model utilizes the sequential to sequential model equipped with attention. The model draws heavy inspiration from this attention model presented by Andrew Ng.

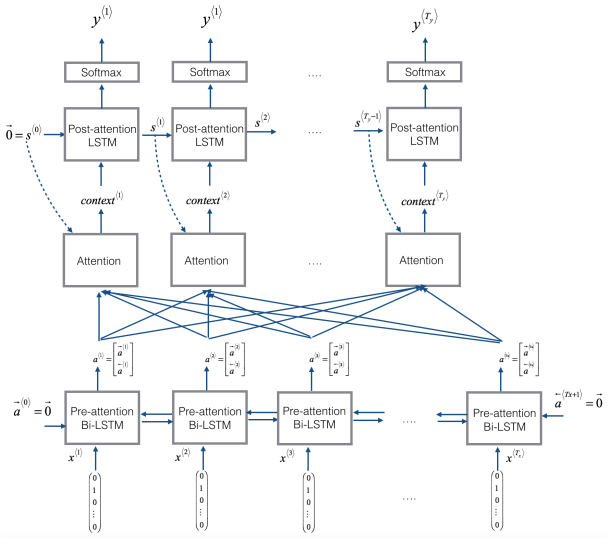

Figure 4: The inspiration for the attention model used in Beatomizer. Created by Richard Ng

A couple modifications were made to this architecture, involving removing the softmax output to make the model run a regression on the output and also simplifying the submodel behind the "Attention" mechanism.
The attention model also comes equipped with a custom loss function which extrapolates differences in RGB values through time, making sure that the model doesn't create a jittery or too flashy of a lightshow. It does so by taking a moving average and invoking that into the loss function.

$$\textbf{Loss} = \textbf{Cosine Similarity} + \textbf{Moving Average Metric}$$

5 Results

The baseline model finished with a cosine similarity of 0.7624 and the attention model finished with a cosine similarity of 0.5541. Training curves for the baseline and attention model are shown below with epoch number on the x-axis and loss on the y-axis:

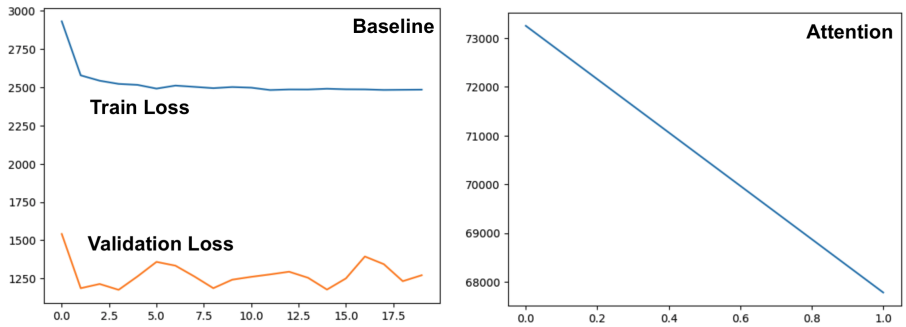

Figure 5: Training curves for the baseline and attention model

Due to the limited computational resources, running a large number of epochs for the attention model was not feasible. Nonetheless, the drastic drop in training loss presented in the attention model signifies

an eventual convergence to a model which performs drastically better than its baseline counterpart. Additionally, it is immediately obvious that that baseline model is performing sub optimally due to its plateauing loss. To combat this, additional convolutional and dense layers were added to the model. With minimal effect, an increase the learning rate was then attempted to combat the vanishing gradient. Yet, when this was again met with minimal effect the conclusion was made that the dataset's distribution is too small, and data augmentation must be explored.
The song "California Dreaming" by the Mamas and the Papas was inputted into both models to test their capabilities. Due to the limited format of a paper, in lieu of the visual representation created by the model the following graph of the "intensity" of the RGB through time illustrates the output.

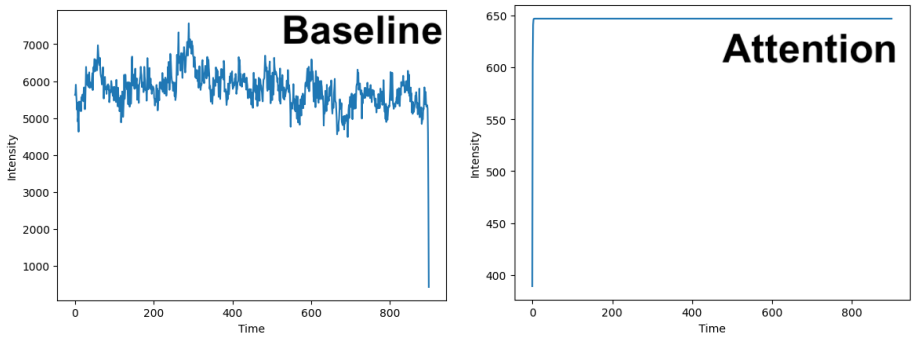

Figure 6: Intesity of the RGB values over time for a predicted song

While the baseline creates a semi-decent visualization of the song, it does so with near equal RGB values. This shows that the model is not learning to make significant commitments to RGB values, which may reflect the dataset since the average pixel value of a frame tends to oscillate between the RGB representation of gray. On the other hand, the attention model has major issues committing to time distinct features through time. This may be due to the fact that the attention model's loss includes the moving average metric, meaning it is learning to localize itself onto one average RGB value (perhaps the exact RGB gray value described above). We can see major advantages with the baseline; not only does it run drastically quicker than the attention model but it creates a semi realistic result. In regards to the attention model, theoretically it should create a more sophisticated model due to its attention to previously predicted values. However, due to the lack of computational resources this result has yet to be seen. Needless to say, once the dataset is better represented and computational assets have been devoted, the attention model will create more visually pleasing depictions.

6 Conclusion

In conclusion, though the baseline model performs reasonably better than its more advanced attention model counterpart, due to the high complexity and high input parameter dimension, I have reason to believe that the attention model has not been provided the sufficient computational resources it requires to completely exhaust the dataset of information. Beatomizer is currently an open project and future work is required to increase its consistancy. Primarily, this will involve a significant devotion of computation resources to the attention model, but promising avenues to look towards involve: increasing the size of the dataset through broader sampling, augmenting the dataset to parse the implied color of the frame instead of the average (which will aid in de-muffling the dataset), and running a top to bottom hyperparameter search on the successful models.

7 References

[1] https://cdn.openai.com/papers/dall-e-2.pdf
[2] https://arxiv.org/abs/1409.0473
[3] https://arxiv.org/abs/1409.3215
[4] https://arxiv.org/abs/1406.1078
[5] https://arxiv.org/abs/1810.04805