ART AND MACHINE LEARNING
CMU 2022 SPRING
PROJECT 2

COVER ARTIST

Generating album cover art using StyleCLIPDraw

"Lose you to love me" - *Selena Gomez*

Group 8
Aarthi Ramsundar: Senior in AI, SCS, some ML experience
Nikitha Murikinati: Senior in CS, SCS, some ML experience
Qi Xuan Teo: Masters in ML, SCS, some ML experience
Audrey Zhang: Masters in Data Analytics, Heinz, some ML experience

DESCRIPTION

Concept

Our team was brought together by our shared interest in music. We were interested in the relationships between the themes in popular songs and the cover art that is used to represent the song to the public. For this project, we wanted to find a way to automatically create visual art based on themes in songs, extrapolated from the song lyrics. To further tie a particular song's themes to visual design, we would also use several existing covers for thematically similar songs as the style images for our target song's cover. Our project thus involves art generation with multi-modal inputs (text and several images). We planned to use StyleCLIPDraw to generate images using these inputs.

Technique

Album Data Collection

To collect details about various songs, we leveraged the Spotify library, using the Spotify API in order to get information about song titles, artists, Spotfify URIs, and album cover art. We then web scraped song lyrics off of Genius using the artist and song names. Songs were filtered out if they did not have lyrics or a cover image available. We were able to gather a collection of about 10k songs through this process. The choice of songs was taken from a Kaggle dataset that contained Spotify URIs we used to query the Spotify API.

Lyric Summary

The lyrics for a song are typically quite long in length. There are also many repeated lyrics in each song. To condense the information of the text input and shorten the length of the input string to the StyleCLIPDraw algorithm, we explored several text summarization techniques, limiting the output summary to 10-20 words. First, we cleaned the lyrics to remove tags such as [chorus] and [bridge], then corrected some formatting issues from the web scrape. We then used two different algorithms (extractive and abstractive) to summarize the song lyrics, and pre-processed this step so the summaries can be directly used as input to the StyleCLIPDraw pipeline. The extractive algorithm tokenized the lyrics and then used NLTK's probabilistic frequency distribution in order to determine two lines in the lyrics to represent the summary. For the abstractive summary experiment, we used the T5 for Conditional Generation (base) pre-trained model to generate lyric summaries.

Finding Similar Covers

The inputs for the style-transfer aspect of our project were existing album covers for songs that have similar themes to the target song. To get the similarity between songs, we used the

FastText algorithm to generate word embeddings for the song lyrics, after tokenizing the words and removing stopwords. We picked FastText since the algorithm takes into account sub-words (character n-grams) when creating embeddings. This allows the algorithm to better handle out-of-vocabulary words than other text embedding algorithms. This was particularly relevant for our project as song lyrics are often contracted, misspelled, or involve onomatopoeia to capture vocalizations.

After calculating the embeddings for the lyric summaries, the average embeddings for each summary were used to get the pairwise cosine similarity between songs. A higher cosine similarity score indicates more similarity between two songs, and the top-5 similar songs for each row in the dataset (i.e. each song) were added to the dataset for later use.

Generating Album Art

The algorithm used to generate album art using lyric summaries and album covers of similar songs is StyleCLIPDraw. We adapted the algorithm to be able to process multiple style images as inputs, using different ways of combining the style embeddings of different images. We experimented with averaging styles, suming losses across multiple styles, alternating styles by interweaving style tensors (by rows or by columns), and clumping styles by concatenating fractional style tensors (by axis 1 or axis 2). The experiments and select results are further described below.

PROCESS

Dataset Selection & Data Collection

Before deciding on music album art generation, we also had a similar idea of generating book covers using the text and the existing cover art of other similar books. We found a Kaggle dataset with book cover information (n=32,600), then used the GoogleBooks API to collect a brief description of each book. Unfortunately, not all books have a description available, so our final dataset with non-missing book descriptions was 16,455 rows. As a group, we felt that it would be challenging to work with a dataset where many of the descriptions were missing and decided to explore song album cover generation instead.

To get song album cover and lyric information, we used the Spotify API and web scraped off genius.com to get the relevant data needed for this project. The initial set of songs was pulled from this Kaggle dataset and cross-referenced with Spotify using the Spotify URIs. Using this method, we were able to get more complete data for our observations. We ran the pipeline for around 10k songs to get the appropriate information. Since this method proved to create fewer missing observations, we decided to use this dataset to generate song album covers.

Lyric Summarization

There are two main approaches in current text summary algorithms: extractive summary, and abstractive summary. Extractive summary pulls one or several sentences directly from the input text to summarize the text, often relying on between-sentence similarity scores to identify the sentence(s) that share the most similarity with all other sentences in the text. Abstractive summary, on the other hand, generates one or several new sentences upon processing the input text. We experimented with both methods for lyric summarization. The summary models were run independently of our final album art generation pipeline to pre-process the input data, and avoid adding computational time for each iteration of StyleCLIPDraw.

Extractive summary:

For the extractive summary experiments, we first tokenized the lyrics and created a NLTK frequency distribution in order to weight tokens based on their frequency. Then each sentence's score was calculated using these weights. The sentences with the 2 highest scores were selected as part of the summary.

Abstractive summary:

For the abstractive summary experiment, we used the T5 for Conditional Generation (base) pre-trained model to generate lyric summaries. The T5 model is a transformer model that uses a maximum likelihood objective function during the generative phase. This can help increase coherency in the output summary.

The generated results were mixed in quality. This is likely because music lyrics have an inherently different structure than normal text. Thus, algorithms trained on text such as Wikipedia articles may not work too well with lyrics. Our team decided that it might make more sense to use the extractive summary results as input text for the album art generation.

StyleCLIPDraw

Our team modified the StyleCLIPDraw algorithm to be able to combine styles from multiple image inputs. We experimented with different methods of combining style inputs, including:

1. Averaging out styles. While the original StyleCLIPDraw uses the VGG19 feature extractor to get feature styles in a 3-dimensional tensor, we extend this to multiple style images by applying the feature extractor on each individual style image, before taking the mean of the styles.
2. Summing losses. We calculate style features for each style image. During each iteration, we calculate losses for each style, and sum them up to get a total loss.

3. Alternating styles by axis 1. One possible weakness of our previous methods is that averaging out scores may lead to different features ‘canceling’ each other out during the mean operation. We would ideally want different aspects of each style to be combined, rather than to have a fraction of each aspect. We were inspired by a blog post to attempt to combine styles over a certain axis. Since our style tensor is effectively two-dimensional (it is a three-dimensional tensor with the size of the first dimension = 1), we can reconstitute our style tensor row-by-row or column-by-column, selecting every k-th row/column from one style image, and so on.
4. Alternating styles by axis 2. This is similar to method 3, but done over a different axis.
5. Clumping styles by axis 1. We combine styles in clumps rather than alternating axes. The axis in question was length 1000, and we had two style images, we would select the first 500 entries of the style tensor from the first image, and the next 500 from the other.
6. Clumping styles by axis 2. This is similar to method 5, but done over another axis.

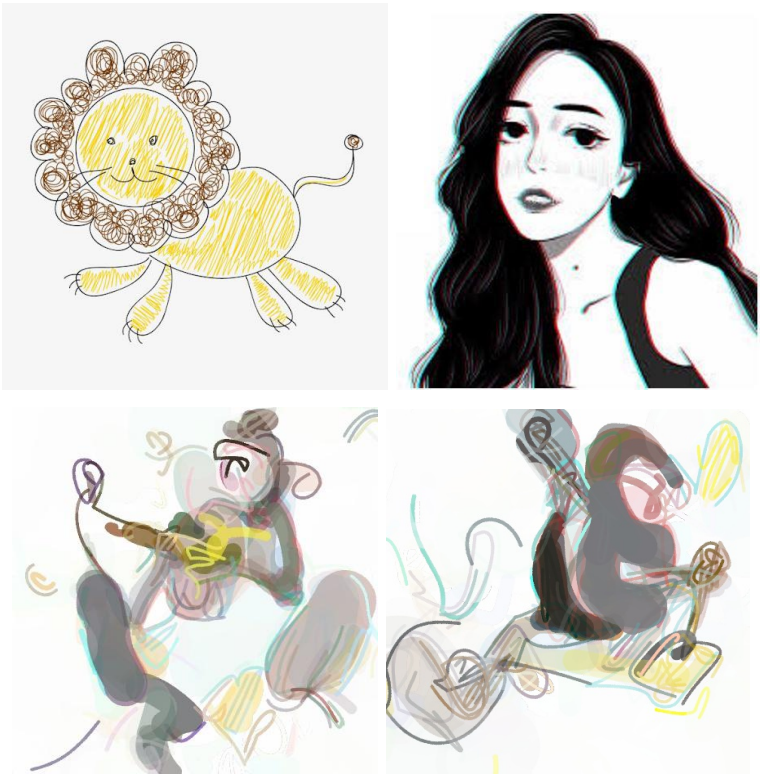
We generated an image based on the prompt “A monkey playing guitar”, using the lion and TikTok style images.

First row (left to right): ‘lion’ Style Image, Experiment 1, 2, 3
Second row (left to right): ‘TikTok’ Style Image, Experiment 4, 5, 6

We noticed that our experiments done on axis 1 led to weird colors emerging, such as green, blue or red, which indicated that such methods may not be combining style images correctly. Among our other experiments, experiments 4 and 6 seem to better combine multiple styles, since the images appear more striking, have more vivid colors, and generally seem to combine elements of both styles rather than merely being a weaker implementation of a single-style

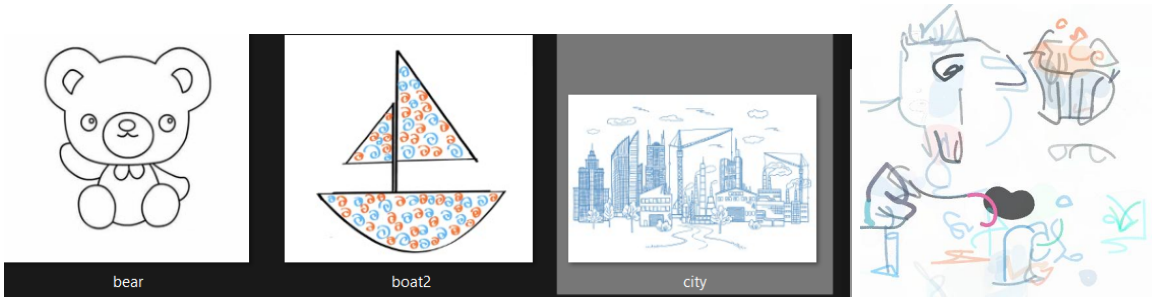
drawing. After analyzing several test image outputs from these experiments, we ultimately decided on experiment 6 as the method of combining different style images.

We experimented with different multi-input style transfer techniques using both StyleCLIPDraw and StyleCLIPDraw Slow pipelines. The creators of StyleCLIPDraw suggested that while StyleCLIPDraw Slow takes longer to run, it is able to produce a stylistically better result. Therefore, we attempted to adapt StyleCLIPDraw Slow to take in multiple style images as input using the six style combination methods described earlier. We ran the slow pipeline for a few examples using each technique, and varied the number of iterations per run. When comparing the results of the slow pipeline with the results of the original pipeline with the same inputs, we did not notice any significant improvement in performance and observed that it still took significantly longer for each result to be generated. As a result, we decided to use the standard StyleCLIPDraw version for our final pipeline rather than the slow version.

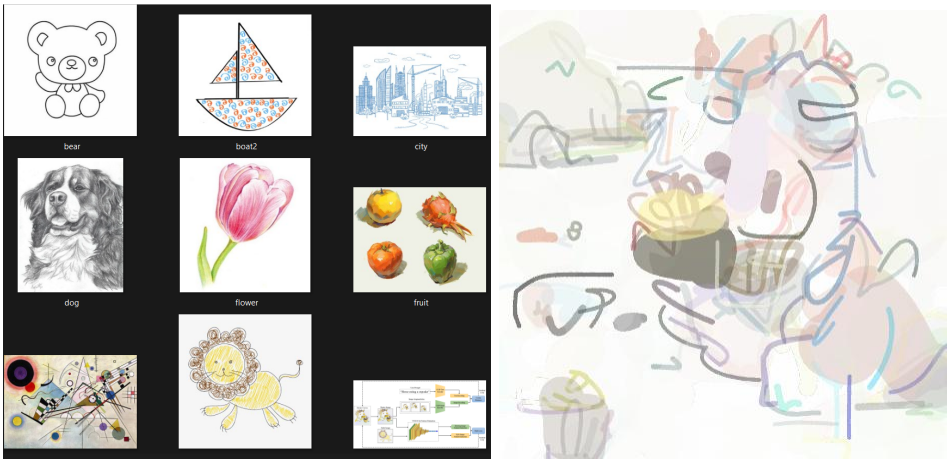


Top: 2 style image inputs
Bottom Left: Result of 2 images (Top) with the prompt "A monkey playing guitar" inputted to Modified StyleCLIPDraw (version 1)
Bottom Right: Result of 2 images (Top) with the prompt "A monkey playing guitar" inputted to Modified StyleCLIPDraw Slow (version 1)

We also experimented with using varying numbers of style input images to test output quality. We conducted experiments that inputted between 1 and 9 style images to observe how the algorithm behaved. We noticed that for up to three style images, performance was relatively good and the result was able to reflect the text input as well as the style inputs. However, beyond three style images, we observed that the style of the image did not appear to combine the style inputs well and was often visually displeasing. We also observed that when a higher number of style images were inputted, the result was less reflective of the text prompt and harder to understand. As a result, we limited the number of style inputs to a maximum of three when working with song lyric inputs.



From left to right: Style images of bear, boat and city, as well as result of 3 images with the prompt "A horse is eating a cupcake" inputted to Modified StyleCLIPDraw (version 6)



Left: 9 style images inputs
Right: Result of 9 images (Left) with the prompt "A horse is eating a cupcake" inputted to Modified StyleCLIPDraw (version 6)

Reflection

While we achieved certain results that we felt were relatively presentable, we still feel that our results would not be feasibly considered a like-for-like replacement for album cover art generation. While our experiments with the original sample prompts (e.g. “A man is watching TV”) highlighted a few issues, the actual results using our summary methods truly exemplified the issues. We will touch on a few of these below.

1. Differences between album covers and drawings
   StyleCLIPDraw, as its name suggests, is based upon CLIPDraw, which uses brushstrokes to recreate drawings. However, most album images tend to be based in the medium of photos. As such, it is impossible to get a like-for-like recreation of album covers. Album covers often also focus heavily on human faces and contain text, which are difficult for the model to handle. We sometimes noticed random English characters in the output of our model, which contain no semantic meaning. It might be better for our model to either obtain a textless version of the album art, or to somehow train a language model to identify the text on album covers, and use that to generate our own album cover text.
2. StyleCLIPDraw is overly literal
   StyleCLIPDraw excels in drawing output based upon a prompt. However, song titles and/or lyrics are often unable to entirely encapsulate the meaning of a piece. In addition, StyleCLIPDraw is unable to capture metaphors in the lyrics (Ex: "bad blood"). This means that the prompts we give StyleCLIPDraw would either be overly descriptive/literal, or contain too much semantic information for StyleCLIPDraw to digest.

Ultimately, this resulted in a much higher number of trials to achieve presentable image results, due to either hyperparameter tuning, randomness, or inherent weaknesses with the prompt (some songs are easier to draw about than others). Although we eventually found multiple results that we were happy with, there was definitely a higher rate of failure for our task as compared to the original setting. We feel that we have made tangible improvements to the project in the form of expanding StyleCLIPDraw to have multiple inputs, and we have also demonstrated the possibility of using the method to generate different forms of art. While StyleCLIPDraw may not be suited to the task of album art generation, we might observe better results in other contexts (e.g. converting short stories into children’s picture books using styles of certain authors).

RESULT

Since the dataset consisted of several thousand songs, many of which were less well-known, we selected a few songs that were more popular and ran the main StyleCLIPDraw function on generated lyric summaries and suggested style images. As decided based on our previous tests, we clumped style images along the 2nd axis to generate the style tensor for the algorithm (experiment 6). We experimented with using the extractive summary, abstractive summary, and song title as text inputs, and tried various combinations with different numbers of input style images. We also ran a quick hyperparameter search on a subset of these songs, but found no noticeable improvements.

We noticed that there is a large degree of variance in the quality of the output. This is inherently due to randomness - the best album covers have similar style images recommended, with relatively little words on the cover. However, since our method of suggesting style images does not take into consideration the similarity of the images, we occasionally get contrasting images that result in less satisfying output. Interestingly, we noticed that our results for Selena Gomez's *Lose You to Love Me* have great similarities to the actual album cover, with matching color scheme and a motif of a female figure. We believe that this shows the promise of the method, and that even better results may be obtained through fine tuning and more task-specific training. A selection of our results are presented below.

"Lose You to Love Me" - Selena Gomez

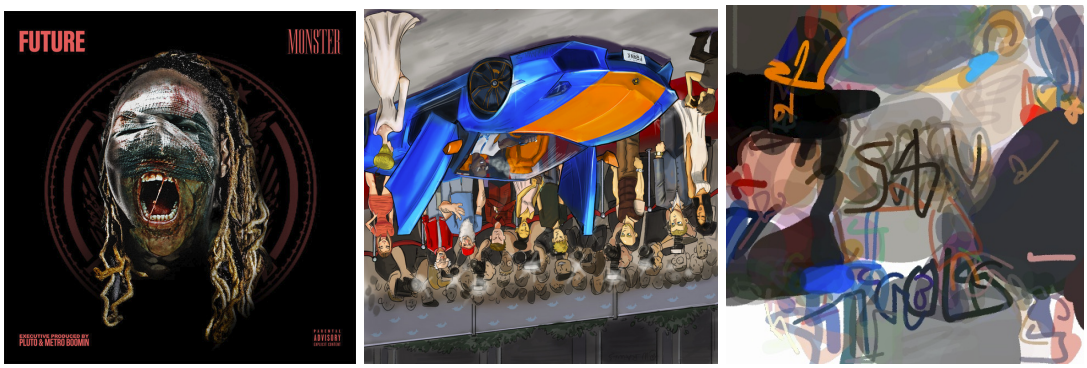
Inputs: song title + 2 style images

From left to right: style images 1 and 2, and output image

“I Write Sins Not Tragedies” - Panic at the Disco

Inputs: extractive summary + 2 style images

“Well, this calls for a toast, so pour the champagne. Pour the champagne”



From left to right: style images 1 and 2, and output image

“Smash Mouth” - All star

Inputs: abstractive summary + 3 style images

“bob greene says he's never been bored with the backstreets”

From left to right: style images 1, 2, and 3, and output image

The rest of our results can be located at this link:
https://docs.google.com/presentation/d/1-PY08hozY0WV7Ni_sLPW8NVW1c1U8sxN7woVwJKuNnE/edit?usp=sharing

CODE

Github repository: https://github.com/aud-z/cover_art_generation

REFERENCES

Agarwal, Vardan. “Combining Numerous Artistic Styles in Tensorflow.” Medium, Towards Data Science, Jun 1 2020, https://towardsdatascience.com/combining-numerous-artistic-styles-in-tensorflow-6e12a99b103f

Badawi, Anas. Song Lyric Compilation Based on Machine Learning. https://github.com/anasbadawy/Song-Lyrics-Compilation-based-on-Machine-Learning

Facebook Inc., FastText, 2022 https://fasttext.cc/docs/en/english-vectors.html

Goutham, Ramsri. “Simple abstractive text summarization with pretrained T5 - Text-To-Text Transfer Transformer”, Medium, Towards Data Science, Apr 16, 2020, https://towardsdatascience.com/simple-abstractive-text-summarization-with-pretrained-t5-text-to-text-transfer-transformer-10f6d602c426

Khan, Maaz. “How to Leverage Spotify API + Genius Lyrics for Data Science Tasks in Python.” Medium, The Startup, 21 Nov. 2021, https://medium.com/swlh/how-to-leverage-spotify-api-genius-lyrics-for-data-science-tasks-in-python-c36cdfb55cf3.

Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." arXiv preprint arXiv:1910.10683 (2019).

Shaldenbrand, Peter, et al. StyleCLIPDraw, Sep 2011, https://github.com/pschaldenbrand/StyleCLIPDraw