THOMPSON RIVERS UNIVERSITY

Final Documentation

April 20, 2024

Group Members:

Hamza Ahmad - T00670206

Martin Atanacio - T00684924

Carlos Avila - T00651182

Problem Description

As mentioned in the project proposal, pixel artists usually struggle to pick appropriate color palettes when creating an environment for their projects. We are aiming to simplify this tedious process by using AI technology and methods to automate this process with accuracy and speediness. Our main goal is to train an adversarial network to recognize a user's prompt and suggest a color palette.

The way in which we will implement this is by utilizing a Conditional Generative Adversarial Network (CGAN). This is a type of Deep Learning architecture in which we train two neural networks to compete one against the other to generate data. It's similar to what we saw in our AI class over the con artist and the detective. One of the neural networks will try and generate new data, which will be our con artist, and the other neural network will try to "predict" or "check" whether this data belongs to the original dataset or not. [1]

Having trained our cGAN to output 5 color palettes with 5 colors each, we have discovered that after numerous tries, we are able to output colors similar to the original ones.

Method

The way we developed the cGan was by first, we loaded the dataset we found on the internet called Text2Colors which consists of 9165 prompts and color palettes. After loading and making sure all the dataset was correct and usable we started developing the conditional generator and conditional discriminator. We followed various tutorials as we had to learn how to use pytorch, pickle, numpy and matplot libraries. We created a class for the Conditional Generator and Conditional Discriminator which contains a nn.Module as a parameter and we defined two functions inside these, one for the forward passing in the NN and: For the Generator, to generate the fake palettes using the specified data set. For the Discriminator, to “discriminate” a specified color palette passed to it and returning an output.

After defining both the CGenerator and CDiscriminator, we proceeded to code the train function that takes in the following parameters: Generator, Discriminator, the Dataset of the prompts and images, the number of epochs(generations) it will take to learn, the dimension of the latent noise vector and the dimension of the prompts, in our case 11. How this function works is, we loop through each one of the epochs and in each epoch we generate fake palettes with a random noise vector. Then we used a real palette and a fake palette, and got both of the outputs. We use these outputs to generate the discriminator loss, which is used to determine how good or bad the discriminator did to identify a real from a fake palette. Next, we reset the parameters for the discriminator and we backpropagate the loss in the discriminator. Finally, we train the generator, we aim to “fool” the discriminator into classifying fake palettes as real ones. We reset the generator’s parameters, backpropagate the loss and we generate the loss for the discriminator and generator and we move to the next epoch.

Moreover, we define a function to pre-process the color palettes and the prompts. For the color palette we simply convert the palette to a tensor (which is what pytorch uses in order to process data). For the prompts, we have to create a sort of dictionary for all the words, then we convert the words to numbers (we need to do this because tensors are arrays of numbers only), then we just convert the array of numbers into a tensor. The last steps are just loading the Tensor Dataset, load it into a variable. Use this to create the Conditional Generator and the Conditional Discriminator. Finally, we just trained the cGAN, added prompts as numbers in an array and output the results with the fake color palette and the original.

Materials

As previously stated, we utilized the Text2Color dataset from the following website: https://github.com/awesome-davian/Text2Colors. Additionally, we used multiple libraries to help generate/create the cGAN and the outputs, the main library used was PyTorch from the following website: https://pytorch.org/. Finally, we used multiple resources to help create and guide us through this process. These are the main tutorials:

- Build a Generative Adversarial Neural Network with Tensorflow and Python | Deep L...
- 247 - Conditional GANs and their applications
- What are GANs (Generative Adversarial Networks)?

Evaluation

Evaluation of the output generated by the method primarily focused on assessing the similarity between the generated color palettes and the original ones. This evaluation was conducted visually, relying on human judgment to determine the degree of resemblance. Additionally, the learning progress of the model was monitored by observing changes in the generated palettes across different epochs.

The evaluation process involved comparing each generated color palette to its corresponding original palette. Human evaluators examined both palettes side by side, considering factors such as color composition, distribution, and overall visual appeal. The similarity between the generated and original palettes was subjectively assessed, taking into account how closely the generated colors matched those in the original palette.During evaluation, it was observed that the method showed varying degrees of success in replicating the original color palettes. Some generated palettes closely resembled their originals, exhibiting a high level of similarity in color distribution and tone. In contrast, other generated palettes deviated significantly from the originals, displaying noticeable differences in color composition or hue.

The evaluation process also involved tracking the learning progress of the model over multiple epochs. Observing changes in the quality of generated palettes throughout training provided insights into the model's capacity to learn and improve over time. Progress was monitored by assessing whether the generated palettes became more aligned with the original ones as training epochs advanced.

In summary, the evaluation of the method primarily relied on subjective visual assessment by human evaluators. By comparing generated color palettes to their original counterparts and monitoring the model's learning progress, insights were gained into the method's effectiveness in replicating original palettes and its capacity for improvement over time.

Limitations

In the development process, one significant setback encountered was the limitation of the dataset. Working with the dataset presented challenges in terms of its size, quality, and diversity. The dataset's size was insufficient to capture the full spectrum of color palettes and textual prompts, leading to biases or limited generalization of the model. Additionally, the quality and diversity of the dataset had influenced the model's ability to learn different representations of color palettes and associated textual descriptions.

To overcome these dataset limitations, several strategies we thought of a few solutions. Data preprocessing techniques were applied to enhance the dataset's quality, such as data cleaning, augmentation, and normalization. Despite these efforts, these limitations continued to hinder the performance of the system. The largest limitation is the reliance on the dataset's equality and diversity, which directly impacts the model's performance and generalization capabilities. Improvements in dataset collection, curation, and augmentation techniques could address this limitation, potentially leading to better model outcomes. Moreover, scalability could be a concern, especially if the dataset size remains limited. Scaling the model to handle larger datasets efficiently while maintaining performance is crucial for real-world applications.

In conclusion, while significant efforts were made to mitigate dataset limitations during development, further improvements in dataset quality, diversity, and scalability are essential to enhance the system's performance and applicability.

Next Steps

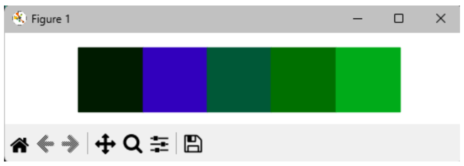
Building a cGAN is no easy task, after hours of coding/studying/researching we barely achieved a minimum viable product which outputs at least 1 similar color to the original palette. I believe that with more refined algorithms and a different design in our training method, we would be able to achieve a more suitable product that outputs palettes more accurately. We would want to, if possible, try to find a different dataset that only utilizes one prompt and that prompt is not repeated to improve the quality of the dataset.

Gallery

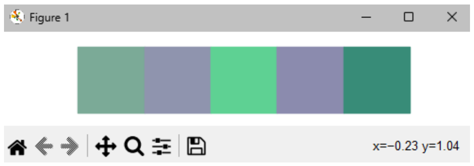
Number of Epochs: 10 - Batch Size: 9165

Prompt: "Green"

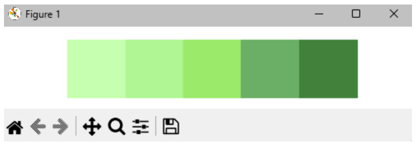
Fake Green Palette



Fake Green Palette V2

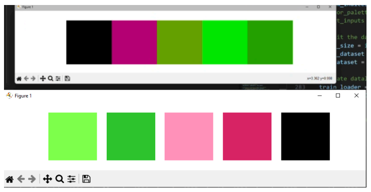


Original GreenPalette



Prompt: "Watermelon

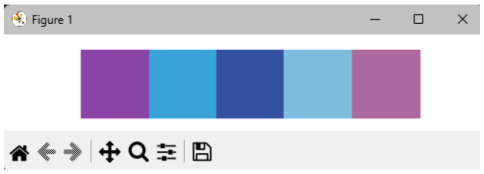
Original Watermelon Palette



Fake Watermelon Palette

Prompt: "Screaming into the void"

Fake Screaming Palette



Original Screaming Palette