ImagePrompt: Discrete Prompt Optimization for Image Generation using Policy Gradient Methods

Reana Naik
Boston University
renaik@bu.edu

Pruthvi Bharadwaj
Boston University
pruthvib@bu.edu

Sandesh Bharadwaj
Boston University
sandeshb@bu.edu

The code for this project can be found in ImagePrompt

Abstract

Prompt engineering is a natural language processing (NLP) problem where the objective is to find optimal inputs for pre-trained Language Models (LM) to obtain desirable or useful results. Although it is possible for humans to manually engineer prompts until the desired output is generated, it is often a time-consuming and labor-intensive process. Automatically finding the optimal prompt for each task, however, is challenging. There are broadly 2 types of prompt engineering approaches that have been explored so far in the Deep Learning literature. One is by using gradient descent on soft prompts (e.g., embeddings) and the other being discrete prompt optimization approaches which involve "enumeration (e.g., paraphrasing)-then-selection" heuristics. Prompt learning methods have been tried on a variety of NLP tasks like Few-Shot Text Classification, Unsupervised Text Style transfer etc. In this project we build a prompt engineering model for text-to-image models (like DALL-E 2, Stable Diffusion, etc) with the objective of finding the right prompt to recreate an image. In particular, we explore a discrete prompt optimization model called ImagePrompt which uses a Policy Network based Reinforcement Learning(RL) algorithm to automate the process of finding the right prompt to efficiently reproduce a target image. This is a proof-of-concept project so we restrict the scope by choosing training examples that consist of images of basic geometrical shapes(e.g. circle, triangle, square, etc.) and basic colors (e.g. red, blue, yellow, etc.). But the same concept can be scaled for the other use cases.

Introduction

Forensic artists use verbal or textual description from a crime witness to portray visual sketches that can help law enforcement identify a criminal. The quality of the drawing by the forensic artist relies heavily on the quality of the descriptions provided by the witness. More importantly, the descriptions should be transferable between multiple artists such that each artist is able to accurately reproduce the similar images every time. It is possible to replace the artist in this process of generating sketches from textual descriptions and automate it using AI models. Generative text-to-image models like DALL-E2 or Stable Diffusion have the ability to generate images from text inputs. The details of the image can be altered by altering the text. An optimal prompt can thus accurately and efficiently recreate a target image. The process of getting the right sketches using a pre-trained model would amount to iteratively improving the prompts.

In this project we built a prompt learning model called ImagePrompt that uses a policy network based reinforcement learning algorithm to optimize prompts. The architecture of the model is explained in the Architecture and Design section and illustrated in Figure 2.

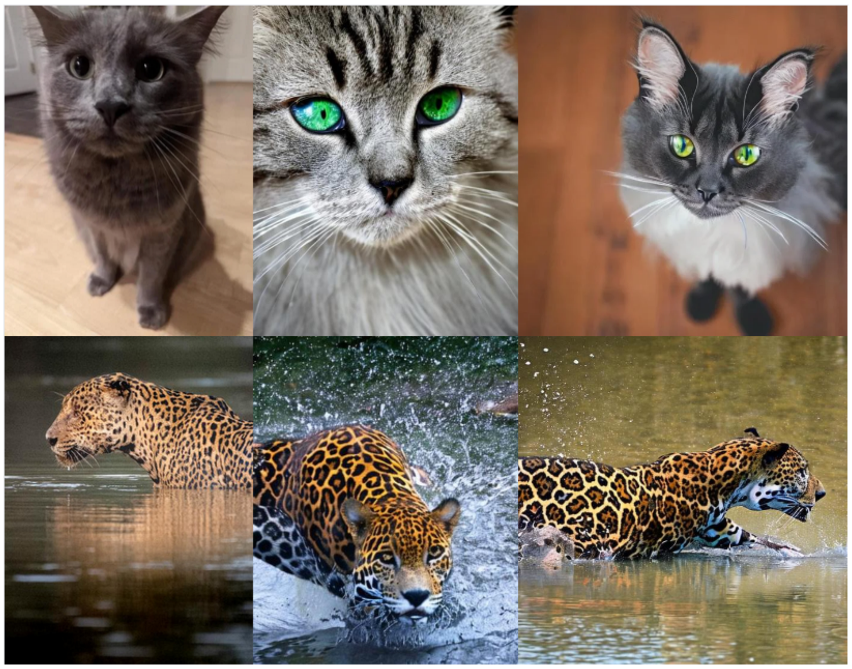


Figure 1: From left to right, in both the rows, the first image is the target image, second image the image generated by DALL-E using model-generated captions (The prompts are: photo of a gray cat standing on hardwood floor looking at the camera with big eyes, photo of a jaguar hunting in water). The third image is generated using human-engineered prompts (The prompts are: gray cat looking camera big green eyes hardwood floor, jaguar stalking in water slow). The first and third images are most similar to each other in terms of features, compared to the second image.

The technical approaches in this paper are closely derived from previous works related to prompt learning, image generation, and image comparison.

- For learning the theory and application of Reinforcement Learning for Prompt Optimization, our main reference was RLPrompt [1], an open-source discrete prompt optimization model that uses policy networks and reinforcement learning (RL) to explore the prompt space.
- We used the stable diffusion model [2] and OpenAI's DALL-E2 as the image generation models
- We used the BLIP[3] image captioning model and the CLIP model [4] to create our prompt space
- We referred to another paper [5] (and its codebase) which attempted to select in-context examples to create the right prompts for GPT3 to solve Semi-Structured Math Word Problems

Problem Formulation

As discussed in the previous section, prompt engineering is a valuable technique that can yield desirable and accurate results from pre-trained language models. The key question is how to find that optimal prompt that will improve a pre-trained model's performance with only a few training examples. An existing and popular solution involves tuning continuous embedding vectors (i.e. *soft* prompts), assuming we had access to the pre-trained model's internal gradients. Unfortunately, these gradients can be expensive to compute or not available for models that are only available as an inference API.

We present ImagePrompt, a new discrete prompt optimization approach based on reinforcement learning (RL). The goal of ImagePrompt is to find the optimal prompt $\mathbf{z}^*$ that maximizes a performance measure R of $y(\mathbf{z}^*, \mathbf{x})$. The variable $y(z, x)$ denotes the output from an image-generation model based on an input target image $\mathbf{x}$ and prompt $\mathbf{z}$. If the length of the prompt is set as a fixed value T, the task of discrete prompt optimization can be expressed as shown below:

$$max_{\mathbf{z} \in \nu^T} R(y(\mathbf{z}, \mathbf{x})) \tag{1}$$

To create a prompt $\mathbf{z}$, the agent will select prompt tokens $[z_1, ..., z_T]$ one by one to maximize the reward $R(y(\mathbf{z}, \mathbf{x}))$. At time step t, the agent receives the previous prompt tokens $\mathbf{z}_{<t}$ and selects the next prompt token z_t according to a policy $\pi_\theta(z_t|\mathbf{z}_{<t})$. In order to create the optimal prompt $\mathbf{z}^*$, we use a policy gradient algorithm to directly learn and optimize the policy π_θ.

Methods

Prompt Optimization Architecture

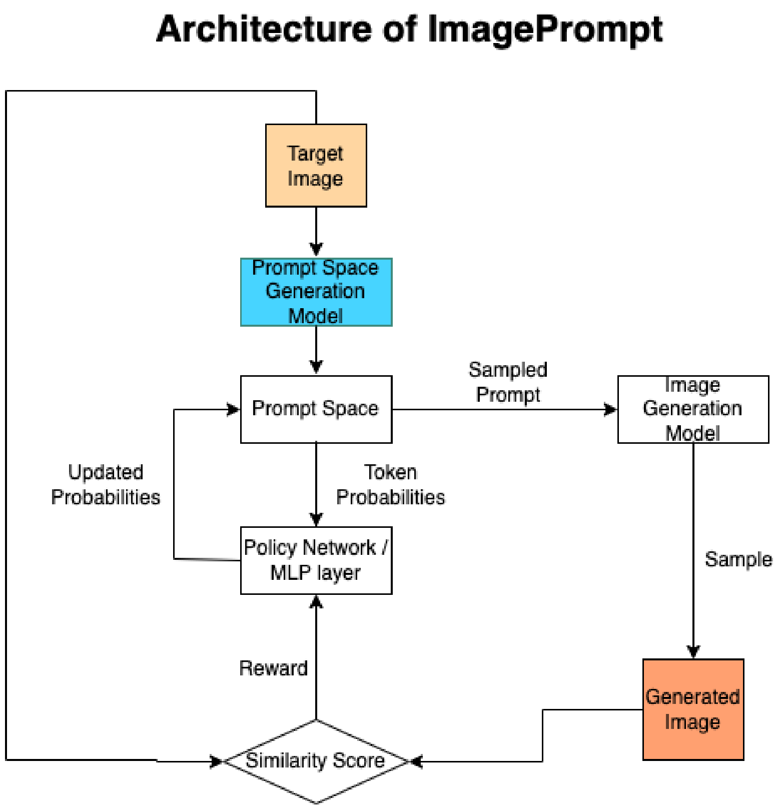


Figure 2: ImagePrompt Architecture

Generating the Prompt Space ν

We start by creating a list $[z_1, ..., z_P]$ that contains unique tokens sampled from a set number C of captions that were generated for an input image using BLIP. This list is called our prompt space ν and it represents a discrete probability distribution of all the tokens we could use to build our optimal prompt. We generate this probability distribution by extracting the image and token features from the input image and prompt tokens respectively using CLIP and then calculating the cosine similarity between them. The cosine similarity is a scalar value between 0 and 1 and represents the probability $P(z_p)$ of a token being associated with an image. After we have calculated the probabilities for each token in μ, we normalize all the token probabilities so that they sum to 1. Lastly, we pick P tokens with the highest probabilities and append them to μ.

Policy Gradient Algorithm

Our next task is to optimize the individual discrete token probabilities such that the similarity score is maximized. In order to accomplish this, we use a policy gradient algorithm called REINFORCE that learns the optimal policy directly, i.e. it is a on-policy algorithm. The parameter θ being updated are the weights of the neural network. REINFORCE uses episode trajectories to update the policy parameter θ and relies on an estimated reward calculated using Monte-Carlo methods. A trajectory consists of all the transitional states, actions and rewards in each episode.

$$\tau = (s_0, a_0, r_1, s_1, a_1, ..., a_k, s_{k+1}, r_{k+1}..) \tag{2}$$

If we formulate our optimization task as an RL problem, the list of tokens in our prompt space will represent the possible actions a that can be taken and our generated prompt, and subsequently, our generated image will act as our state s. We then calculate the return for a trajectory τ as the following:

$$R(\tau) = G_1 + G_2 + ... + G_k \tag{3}$$

where G_k is the future return at time step k for a transition episode k. We optimize θ by maximizing the expected return U(θ) defined as:

$$U(\theta) = \sum_{\tau} P(\tau;\theta) \cdot R(\tau) \tag{4}$$

where $P(\tau;\theta)$ is the probability of each possible trajectory.

Collecting Data from Episodes

During an episode, we use the current policy to choose our first transition token, without replacement, from the prompt space and store it in our token history list. This token is used to create the prompt that represents our current transition state. We then use our text-to-image generation model, DALL-E, to generate an image based on the current prompt state and store that data in our state history list. We then measure the similarity between the target image and the current generated image and this will be our transition reward that is stored reward history list. We continue these steps until we build the entire prompt, which ends the episode. We restart the process again every new episode and append all those transition states, actions and rewards into our running state, action and reward lists. At the end of a trajectory, we will have a history of all the transition and final states, actions and rewards for all the episodes combined into three respective lists.

Model and Training

After we have collected the data from multiple episodes, we feed it into a neural network. Our model is a two-layer network with an input size of 512 neurons, a hidden layer of 128 neurons and an output layer of neurons that equates to the number of tokens in the prompt space. We also implement a ReLU activation in the first layer and a softmax activation in the output layer.

During one iteration of training, we first initialize our neural network and define our optimizer. For training, we will use the Adam optimizer.Next, we collect the state, token, and future returns for a specific number of episodes. We run the forward pass of the neural network with the state history as the input and calculate the probability distribution of each state in our episode history. We then calculate the log of the probabilities of the actions given the policy distribution. The loss is defined as the minimization of the equation for expected return in . The loss is then propagated backwards to update θ in the direction of changing the probabilities such that we maximize R.

Experiments and Results

ImagePrompt can theoretically be used for any desired image so we conducted several experiments to evaluate our proposed approach. Due to limited computing power, we restrict the scope of our experiments to short prompt sizes ($2 < T < 3$) and images with few features. Our target image for this series of experiments will be a 3D rendering of a red sphere. The basic features in this image will help us realistically achieve results since we lack the necessary computing power needed to train complex images with longer prompts.

Experiment 1: Prompt Generation using ImagePrompt

ImagePrompt should ideally be able to build full prompts from the ground up with only a target image as the input. Therefore, this particular experiment aims to generate an optimized RL prompt based on a target image. While training, ImagePrompt will iterate through various combinations and permutations of the tokens such that the similarity score between the generated and target image is maximized.

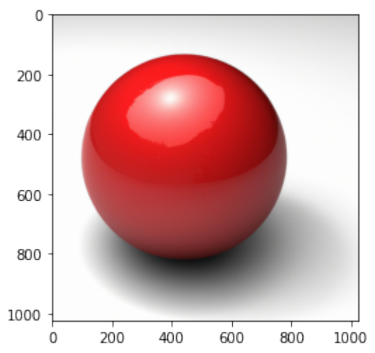
For the experiment, we set the number of captions to be generated to `n_captions = 5` and the number of tokens we want to sample for our prompt space to `n_tokens = 4`. With these hyper parameters, we generate a prompt space and an associated discrete token probability distribution:

```
Prompt Space: ['sphere', 'ball', 'egg', 'red']
Prompt Space Probabilities.: [0.2380530 0.2610056 0.2390593 0.2618819]
```

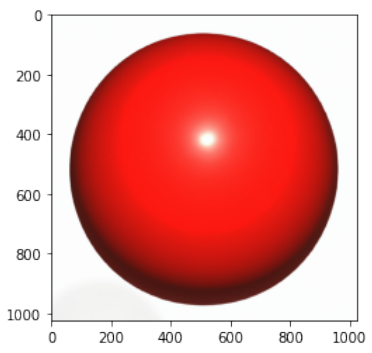
We then train our model for 20 iterations, each with a trajectory containing data from 20 episodes. During inference, we greedily choose the tokens with the highest probabilities to generate our prompt. Based on these training parameters, we were able to train a model that produced the prompt "red sphere".

```
Generated Prompt: red sphere
```

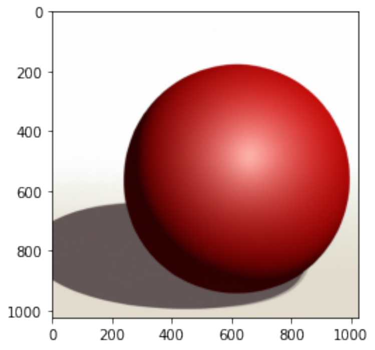
When we feed our generated prompt into DALL-E, we generate the image shown on the right in figure 4. The average similarity score for all images is 0.902. Given the high similarity score, ImagePrompt was accurately able to iterate through the prompt space and choose the tokens that generate a maximized score.



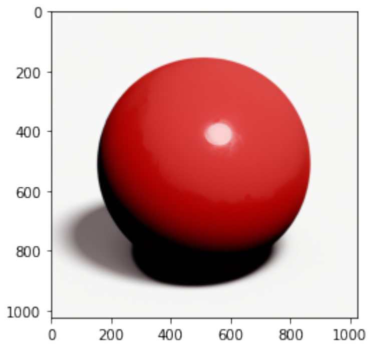
(a)



(b)



(c)



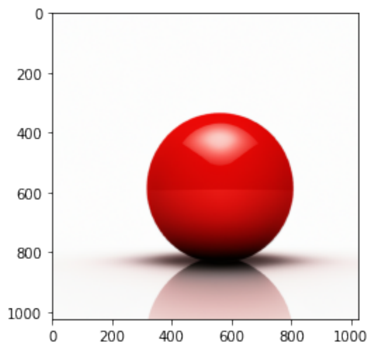
(d)

Figure 3: The figures (a)-(d) shown above are the images generated by DALL-E using the prompt created by ImagePrompt. The prompt "red sphere" seems to capture all the relevant features of the target image given that the average similarity score for all four images is 0.902.

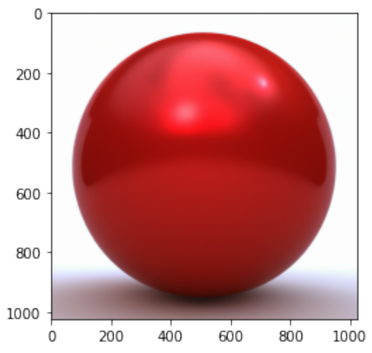
Experiment 2: Prompt Generation using BLIP and CLIP Only

In order to evaluate the performance of ImagePrompt, we experiment with generating a prompt solely based on the initial probabilities computed by CLIP. Once again, we pick the tokens in descending order of their probabilities and create the following prompt:

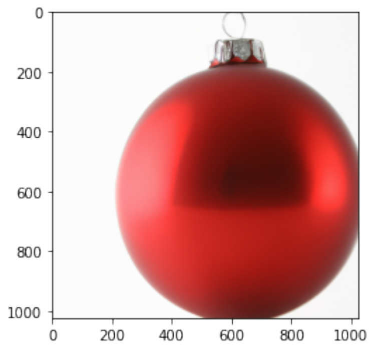
```
Generated Prompt: red ball
```



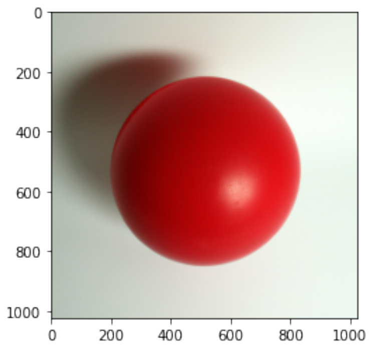
(a)



(b)



(c)



(d)

Figure 4: The figures (a)-(d) shown above are the images generated by DALL-E using the prompt created by BLIP and CLIP. The prompt "red ball" seems to almost capture all the relevant features of the target image given that the average similarity score for all four images is 0.863, slightly lower than the prompt generated by ImagePrompt.

The generated images are very similar to the target image, with an average similarity score of 0.863. It seems like the prompt "red ball" can be interpreted in ways other than simply a red sphere. The image generated in Figure 4c generated an image of a red Christmas ornament, which is very similar to the target image but the additional features is what caused the score to go down.

Conclusion

In conclusion, we were able to build and train a model that can theoretically build the optimal prompt using policy gradient methods. However, we were unable to train the model for a longer period of time due to issue with DALL-E's servers. We managed to train for at most 400 episodes and generate results for a very simple example. Our model was able to create a prompt that generated an image that had approximately a 90% match with the target image. As an evaluation metric, we also compare ImagePrompt with CLIP and BLIP, which are existing image captioning models. Those models were able to generate prompts that achieved a high similarity score, but not as high as ImagePrompt. ImagePrompt is therefore a good starting point for achieving the goal of generating the best prompt. We hope to continue work in the future to improve on ImagePrompt by training it for more episodes and iterations and experiment with longer prompts and complex images.

Git Repositories

- PromptPG
- Deep-RL Exercise
- Policy Network for CartPole
- OpenAI DALL-E2 API reference
- Stable Diffusion API reference
- CLIP
- BLIP

References

[1] M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. P. Xing, and Z. Hu, "Rlprompt: Optimizing discrete text prompts with reinforcement learning," *arXiv preprint arXiv:2205.12548*, 2022.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[3] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," 2022. [Online]. Available: https://arxiv.org/abs/2201.12086

[4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[5] P. Lu, L. Qiu, K.-W. Chang, Y. N. Wu, S.-C. Zhu, T. Rajpurohit, P. Clark, and A. Kalyan, "Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning," *arXiv preprint arXiv:2209.14610*, 2022.