DEEPDIXIT: AN IMAGE GUESSING GAME

Nishkrit Desai
nishkrit.desai@mail.utoronto.ca

Ethan Baron
eth.baron@mail.utoronto.ca

George Saad
g.saad@mail.utoronto.ca

ABSTRACT

We present a game that shows players AI-generated images and allows them to guess the prompts used to generate these images. The primary machine learning task in creating this game is developing image generation models that produce realistic yet interesting images from text prompts. For our image generation, we experimented with using CLIP as a scoring function combined with either SIREN or BigGAN as a generator network to produce images given a text input prompt. We then set up a database and website that allows users to play our game in their browser. The major limitations of the current version of our game include low quality of distractor options and some issues with modal collapse in image generation. We describe possible approaches to mitigate these issues with future investigations. Lastly, we discuss the ethical implications associated with our project. Our code is available at nshdesai/deepdixit.

1 Introduction

Our goal for this project to create an online browser-based game that shows the player an AI-generated image and lets the player try to guess the prompt that was used to generate that image. The key question we hoped to answer was "Can we leverage AI image generation to create fun and interesting game?" Our idea is inspired by pictionary and the board game "Dixit", which is why we called our project "DeepDixit".

The primary machine learning task for our envisioned game is an image generation module. We first present prior work related to this task. We then describe our data collection process and the methods we explored for image generation, including CLIP, vision transformers, and sampling from BigGAN. Finally, we present our final product and discuss our results.

2 Related Work

There have been some recent advances ([1]–[3]) in the field of image generation using large pre-trained vision transformers and diffusion networks to generate a diverse and realistic set of images from a text prompt. A major drawback of these methods, however, is that they require a *lot* of computing power and data to perform well. Since we don't have access to such resources, we are

forced to explore cheaper ways of performing the same task. Nevertheless, these models produce really high-quality results on complex prompts. Some of these results are shown in Figure 1.

(a) Baby daikon radish in a tutu walking a dog — DALL-E

(b) A hedgehog using a calculator — GLIDE

(c) An Astronaut riding a horse — DALL-E 2

Figure 1: Results from pre-existing image generation models

CLIP (Contrastive Language–Image Pre-training) is a Vision Transformer (ViT)-based deep neural network by OpenAI that has been pre-trained contrastively on a large set of (image, text) pairs from across the internet. Via pre-training, the model has learned visual concepts and can then be trained to perform a broad variety of vision tasks, including zero-shot image captioning. Specifically, the architecture of CLIP includes a modified ResNet50 for the image encoder, and a masked self-attention transformer for the text encoder. OpenAI has trained the encoders to maximize the dot-product between text embeddings and image embeddings for (image, text) pairs via a contrastive loss. These pre-trained encoders can then be used to generate zero-shot predictions of text given a previously unseen image [4]. Figure 2 provides a summary of the main ideas behind CLIP. The model card for CLIP can be found here.

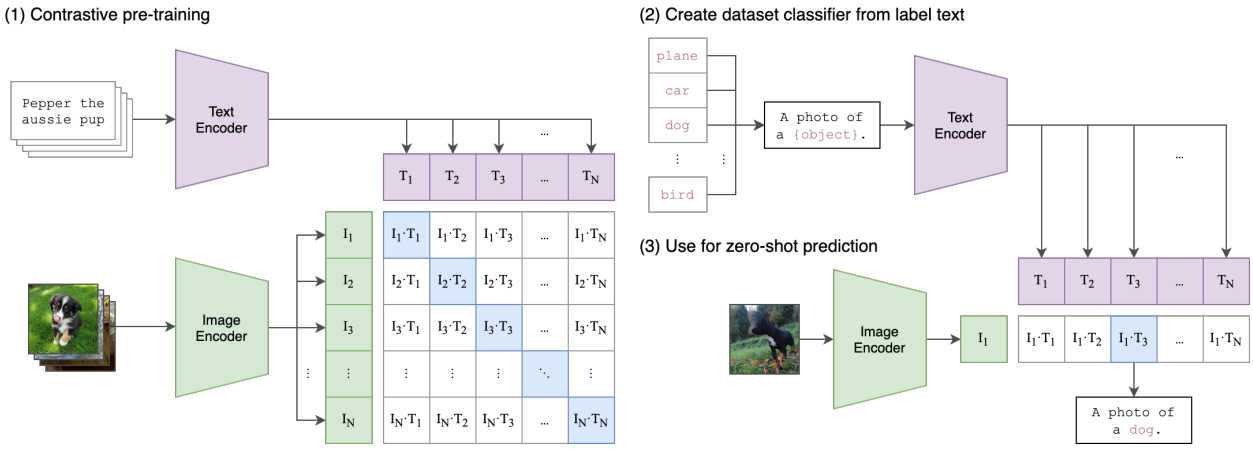


Figure 2: CLIP Summary Graphic, taken from here.

The idea to combine an image generator with CLIP to perform language-guided zero-shot image generation was used by Ryan Murdock, who combined CLIP with VQVAE and SIREN to create Deep Daze [5], and with BigGAN to create Big Sleep [6].

VAEs (Variational Autoencoders) are networks that can create interesting reconstructions of signals (such as images) while also generating latent embeddings for these signals. They are typically

used in cases where the latent variables of a signal can be leveraged to generate meaningful reconstructions of a signal. For example, an animation transitioning between two images can be created by performing a latent interpolation over their latent embeddings. Unfortunately, VAEs are known to suffer from issues such as posterior collapse while also coming with constraints such as requiring a manually-defined prior.

VQVAEs (Vector Quantized VAEs) use a VAE to create discrete latent embeddings using vector quantization [7]. They address both these issues of VAEs by providing mechanisms to train the network using learned priors while also employing vector quantization to address the issue of posterior collapse. This makes VQVAEs a fitting architecture to generate realistic image reconstruction from either ground truths or latent embeddings.

SIREN (sinusoidal representation network) is a multilayer perceptron network that uses sinusoidal activations. Since the second derivative of the ReLU activation function is 0 almost everywhere, it is challenging if not impossible to use ReLU MLPs to learn higher order derivatives of a signal. [8] show that SIREN can help represent differentiable real-world signals (such as images) much more realistically. This makes sense since sinusoidal functions form a Fourier basis that can be used to approximate any continuous signal. Figure 3 shows an example of how the quality of image generation results improve with a SIREN generator.

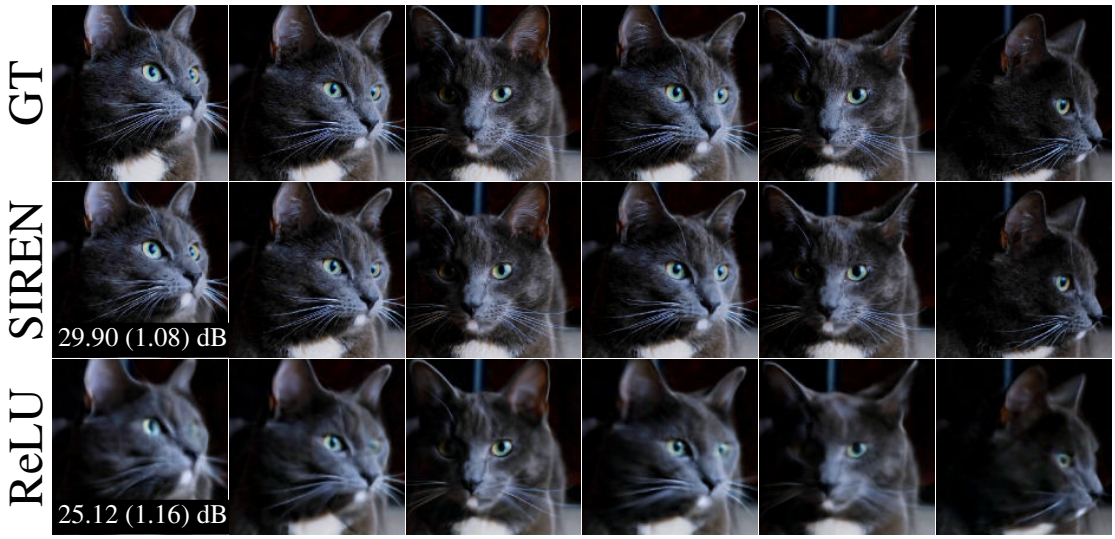


Figure 3: Reconstruction of video frames. It is clear that SIREN can reconstruct the detail in an image better than ReLU networks [8].

Although SIREN networks are able to reconstruct natural signals, they don't have any mechanisms for creative reconstruction. That is, they only provide the mechanisms to reconstruct an image, but not produce imaginative-looking visualizations. This comes from the fact that SIREN has modified activations that have the right inductive biases to produce realistic image signals, however, they haven't been trained on any real image data in the context in which we use them. BigGAN (described below) addresses this shortcoming.

BigGAN is a large GAN used to perform realistic a diverse generative image modelling. [9] use a variety of regularization methods to make training BigGAN more stable after training on a lot of data. However, one of the biggest problems with training GANs is that they suffer from modal collapse [10]. This becomes especially problematic when performing zero-shot prediction.

One way to deal with the instability of GANs is to use better sampling techniques. The predictions of generative models can be interpreted as point estimates, but they can also be interpreted as performing parameter estimation of posterior probability distributions which are sampled from to generate predictions. Varying the sampling method can result in more stable outputs. Advanced sampling methods such as MCMC sampling [11] and Gibbs sampling [12], or incorporating direct categorical sampling in the forward-pass using the Gumbel-softmax trick [13], can help combat this issue, but are more useful when combined with generative networks that further leverage their differentiable properties (such as Hopfield networks [14], RBMs [15], or Deep Belief Networks [16]).

3 Data Collection

The primary data gathering task for our project is collecting possible prompts that can be used for image generation. We have started performing this data gathering step using three approaches: manually writing prompts, using web scraping software to collect prompts from the internet, and using GPT-3 to obtain AI-generated prompts. A summary of the prompts collected so far is provided in Table 1, including information on the number of prompts collected in different categories, and a few examples. We describe the various methods used to collect prompts in the sections below.

3.1 Manually Writing Prompts

Since the image generation process did not go as smoothly at first as we had hoped, we decided to manually compose a few prompts that would help us quickly diagnose the performance of the image generation models. Specifically, we focused on simple prompts for which we would expect good images to be reliably generated. We composed prompts from several categories, in order to identify whether certain models were able to generate images reliably from certain categories more than others. In total, we composed 34 prompts from 5 categories.

3.2 Scraping Prompts

To gather prompts from specific themes, we scraped data from several websites. This process required identifying websites with the required information, parse the pages' contents to extract the relevant data, and post-process the data such as by cleaning format inconsistencies. We developed our web scraping software using Python and Beautiful Soup.

So far, we have collected prompts from three diverse categories using web scraping: Star Trek Episodes [17], Beatles Songs [18], and titles of the AAAI's Classic Paper Awards winners [19].

3.3 Data Collection

We also gathered data from pre-processed data files. Specifically, we obtained a list of the top 100 cited articles of all time, per Google Scholar, as of 2014[1].

3.4 AI-Generated Prompts

Another source of prompts was generating them with GPT-3. This was done by giving GPT-3 various different starting phases and examples and by varying the hyperparameters. For example, one of the

[1] https://www.nature.com/news/the-top-100-papers-1.16224

Category	Source	Prompts	Examples
Clothing	Manual	8	“purple shirt”, “baseball cap on the floor”, “graduation gown”
Food	Manual	6	“slice of pepperoni pizza”, “strawberry ice cream”, “eggs and bacon breakfast”
Home	Manual	5	“square orange carpet”, “cozy fireplace”, “empty backpack”
Nature	Manual	6	“sunset in the seas”, “sandy beach”, “snowy mountain range”
Sport	Manual	9	“ski lift in the summer”, “small soccer ball”, “tennis racket on a wall”
Star Trek Episodes	Wikipedia	809	“The Devil in the Dark”, “Wink of an Eye”, “Loud as a Whisper”
Beatles Songs	BeatlesBible.com	341	“Sea Of Monsters”, “Happiness Is A Warm Gun”, “Child Of Nature”
Academic Papers	Nature.com	100	“Measurement of protein using bicinchoninic acid.”, “Multiple range and multiple F tests.”
AI papers	AAAI.org	39	“Boosting Combinatorial Search through Randomization”, “Reactive Reasoning and Planning”
General	GPT-3	100	“A couple takes a selfie“, “A group of friends hangs out together“
Music	GPT-3	30	“Music can move mountains“, “No one can stop the music“
Nature	GPT-3	30	“A deer grazing in a meadow“, “A waterfall tumbling over a cliff“
Food	GPT-3	40	“A crispy, golden onion ring“, “Freshly cracked eggs“, “A slice of cheesy pizza“

Table 1: Summary of prompts gathered so far

prompts provided to GPT-3 was "Generate 10 short image captions about music:" while varying the temperature and top P to control the variance in the captions generated. Furthermore, some of the generated prompts and manually written prompts were provided as a prompt to GPT-3 along with asking to generate similar captions. The frequency penal and presence penalty hyperparameters were modified as well to control the variance in the captions, and the maximum length hyperparameter was modified to control the amount of prompts generated.

4 Method

In this section we describe our CLIP-based model for performing image generation. We also discuss some trade-offs involved with using different generators. Lastly, we describe the architecture we used to set up our final product.

4.1 Image Generation

Our project requires generating images given a text prompt. Since these images must be completely new, this is case of *zero-shot learning*. To perform zero-shot learning, we use pre-trained language-

image networks. Specifically, we explore CLIP in combination with SIREN or BigGAN. We include examples of the results generated from each approach below, with the prompt used to generate each image listed in the caption.

In experimenting with image generation, we have created a dataset that consists of hundreds of images. Although most of these images provide some insight into the image generation, in the interest of brevity, we have not included all of them in this report. As a consequence, the report tends to present a cherry-picked set of images that best demonstrate the ideas mentioned. To provide a more complete picture of our results, this link shows the raw images generated on our manually gather prompts using the generator described in section 4.1.3 (along with some miscellaneous images from previous iterations and generators).

4.1.1 CLIP as a Scoring Function

We take advantage of CLIP by using it as a scoring function that allows us to train an image generator. Specifically, the idea is to pass text as an input for a deep generative network to generate an image, and then use this CLIP to predict a prompt describing this image. Then, we can define a loss between the CLIP-generated description and the original prompt. This loss can be back-propagated through the generator network to drive the network to produce a realistic image that effectively captures the prompt text. Figure 4 shows an overview of this architecture. The cost function used is just a cosine similarity:

$$\mathcal{L}(\hat{x}, x; \theta) = \frac{\hat{x} \cdot x}{||\hat{x}||_2 \cdot ||x||_2} \tag{1}$$

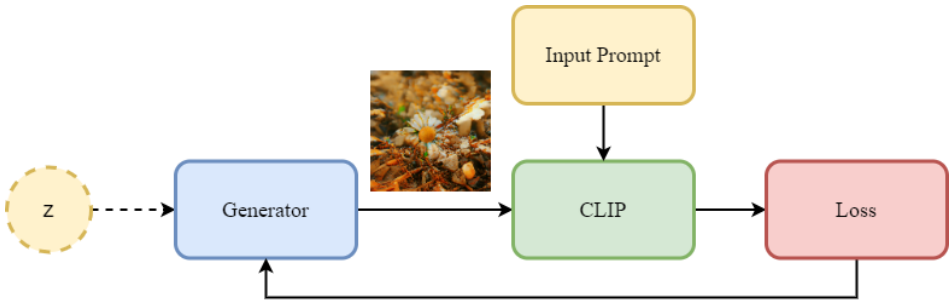


Figure 4: Architecture for using CLIP as a scoring function

We have explored two options for the image generator network: (1) SIREN, and (2) BigGAN.

4.1.2 SIREN Generator

To make an image generation network, we can simple use a deep SIREN network that optionally takes in a latent code as input. As an example of the results generated using SIREN, see Figures 5a, 5b, and 5c. We see that the images do somewhat correspond to the prompts, but are not very realistic representations.

4.1.3 BigGAN Generator

SIREN networks, although able to reconstruct natural signals, don't have any mechanisms for creative reconstruction. That is, they only provide the mechanisms to reconstruct an image, but not produce imaginative-looking visualizations.

Using BigGAN as our image generator network also allows us to perform image generation with class-conditional latent embeddings or additional context desired during image generation. For

(a) Purple Shirt

(b) Thin Hulk driving

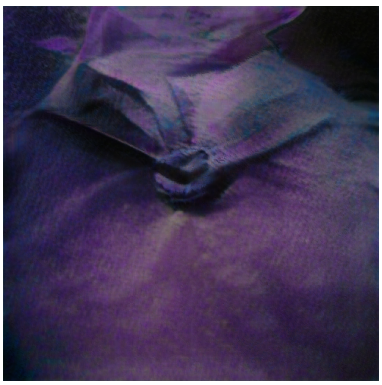
(c) Monkeys in a school bus

Figure 5: An assortment of results for our image generation model

instance, we are able to explicitly specify prompts that should be avoided in generating the images. Additionally, this comes with many benefits of being able to generate a more diverse collections of images.

A few results generated using BigGAN are shown in Figures 6a, 6b and 6c. Note that the results from BigGAN are a lot more realistic-looking than SIREN as this generator network has been conditioned on a really large number of real images (which is not the case our SIREN network).

(a) Golf course



(b) Purple shirt

(c) Cozy fireplace

Figure 6: An assortment of results for our BigGAN image generator

We found that using BigGAN to generate images resulted in many images that resembled close-up photos of birds, butterflies, or insects. As an example, see Figure 6a. Our hypothesis is that since the generation was preconditioned to penalize blurry photos, the photos selected by the network to match were primarily high-quality zoomed-in photos of wildlife. This may also be an effect of the modal collapse issues for GANs discussed above.

To curb this effect, we experimented with several forms of sampling from BigGAN, including:

- *Top-k sampling* samples from the k points with the highest probability masses from a discrete probability distribution. We use this sampling method as the baseline, and noticed that while it works okay sometimes, modal collapse arises often.

- *Nucleus/Top-p sampling* samples from the k points with the highest probability masses such that their cumulative probability is at least p. This method provides good regularization in general but was not effective in our case.
- *Gibbs sampling* (limited experimentation only). Using Gibbs sampling allows us to directly sample from large, complex distributions. However, they require a Bayesian formulation of neural networks. We tried a series of hacks to perform Gibbs sampling from the generator's distribution. Regardless, Figure 7d shows some preliminary results from this method.

After experimenting with these techniques, we found that using a temperature-based top-p sampling method provided the best balance between the diversity of samples, quality, and computation time. Figure 7 shows the best examples generated while applying different sampling methods while trying to generate images of cats.

(a) top-k sampling (b) top-p sampling (c) top-p with temperature (d) differential sampling

Figure 7: Images generated for prompt "sleeping cat" using different sampling methods

4.2 Project Architecture

To meet our goal of creating an online interactive game based on AI-generated images, we had to set up several components, including creating a web server to host an online version of the game, building the website's front-end, and instantiating a database in which to store prompts, images, and other important data points. A summary of our project architecture is shown in Figure 8.

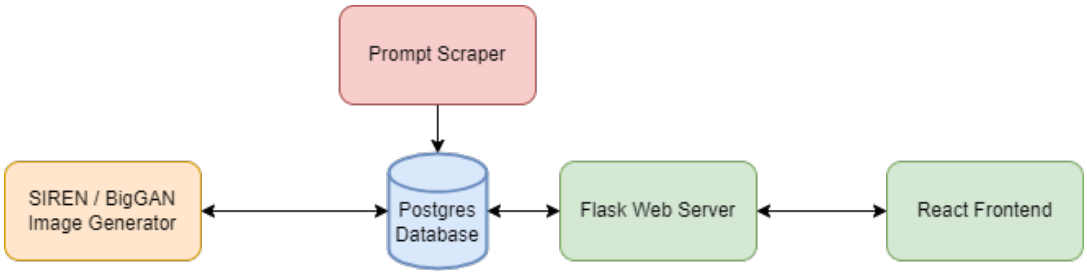


Figure 8: Overall Project Architecture Diagram

4.2.1 Web Server

In order to implement all the features of the project, including prompting the user with an image, the multiple choice prompt selection, and image scoring, we implemented an API using the Flask library in Python. We have implemented an endpoint to retrieve a random image, along with its associated real and fake prompts. The fake prompts are sampled randomly from the real prompts in

the same category as the random image. In order to ensure that the same images don't appear close to each other, 10 unique images are cached at a time. This ensures that there at least 10 images appear before the same image appears again.

4.2.2 Database

Our website architecture requires a database in which to store relevant data, such as the prompts, categories, and images. Before setting this database up, we created a draft database schema (see Figure 9 for a screenshot). The draft schema can be found here. Some considerations that went into this schema included how to connect prompts to categories and images, and how to store players' inputs for both the free-form and multiple-choice modes.

Using the SQLAlechmy ORM library in Python, we created classes for all the database schemas required based on our model and added the code required to connect to the database (which is hosted on AWS via Heroku) in Python. Using SQLAlchemy for the database connection setup allows us to use Python scripts to save data points directly to the database.

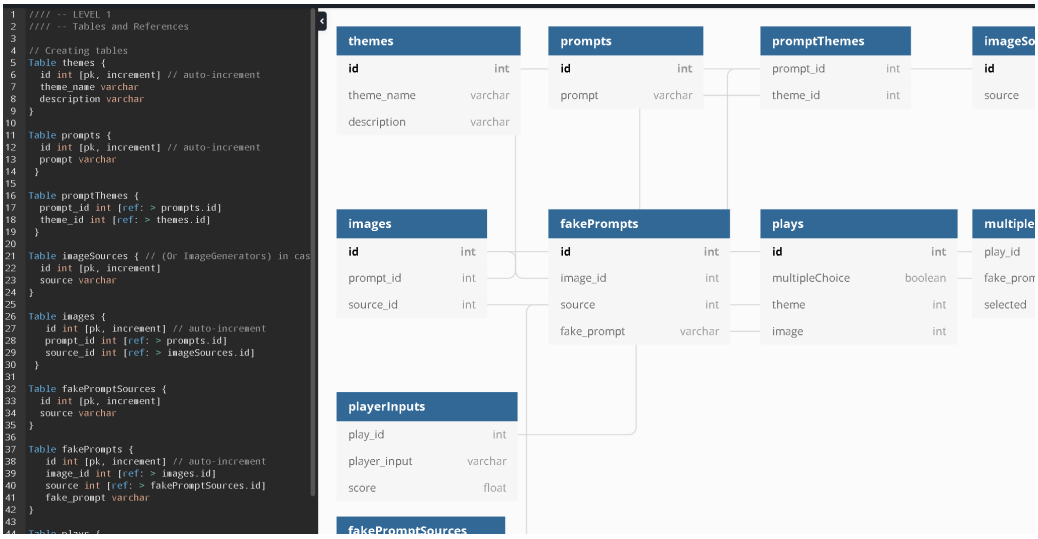


Figure 9: Screenshot of database scheming software.

4.2.3 Frontend

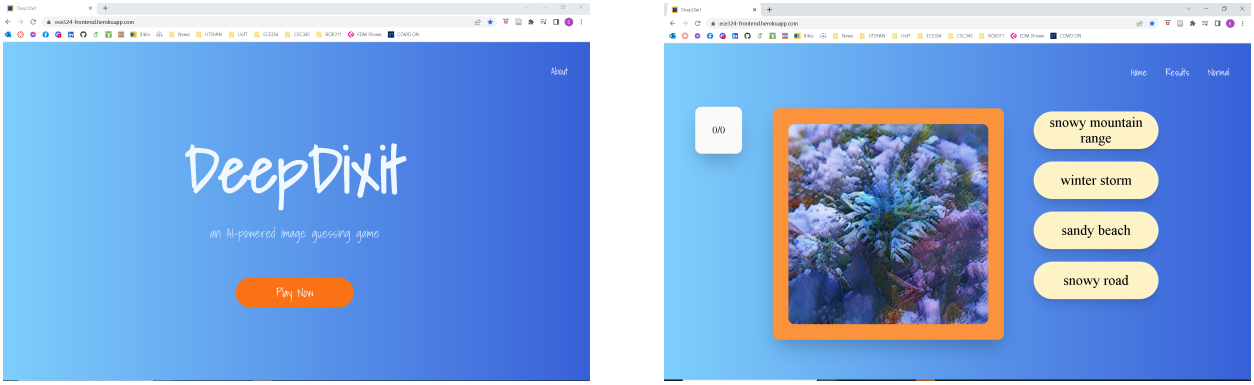
We built the front-end using React and it makes a POST request to the 'random-image' endpoint through the web server to retrieve the images and prompts and keep track of the cache. The main components of our frontend application are as follows:

- generated images (randomized, retrieved from the database through the web server),
- multiple choice prompt selection (random prompts from within the same category as the random image), and
- user's game score (correct guesses out of of total guesses)

We created two views for the website: a "presentation" mode for use on large screens and a "normal" mode to use for individual gameplay.

5 Results

The final product for our project is an online interactive browser-based game that is available to play at https://ece324-frontend.herokuapp.com/. Upon launching the website, the player sees a screen that looks like Figure 10a and is prompted to press a button labeled "Play now". Upon clicking this button, the user is taken to the game play page, which looks like Figure 10b. On this page, they are shown an image that we have generated using neural networks and loaded into our database. They are also shown four options of captions, one of which is the actual prompt used to generate that image. Upon guessing an option by clicking on it, users are shown the correct choice, and their score is incremented in the top-left corner of the screen. Then, a button labelled "Next Round" appears that loads a new image and prompt options when clicked.



(a) Home page (b) Gameplay page

Figure 10: Screenshots from final product

Our website received generally favourable feedback when shown to several users. Players seemed to enjoy the game idea and were satisfied when they were able to guess correctly. Generally, players were fairly accurate with their guesses, often scoring upwards of 70%. There were two aspects that received some negative feedback. First, many of the images did not closely resemble any of the options shown, including the correct prompt. One major contributing factor to this was the issues with modal collapse described above. Secondly, since distractor options were selected randomly, in some cases the correct answer was fairly obvious, making the game less enjoyable. More effective fake prompt generation would allow us to improve this aspect of the user experience.

6 Discussion

The two main limitations of our current approach are: (1) some of the generated images do not align well with the input prompt, and (2) the distractor prompts shown to the user make the correct answer rather obvious.

The first issue, regarding poor generated images, normally stems from the problem of modal collapse when working with a BigGAN generator. As discussed in section 4.1.3, this resulted in many of the generated images resembling plants, birds, or insects instead of the actual input prompt. Such images detract from the user experience of the game as they do not correspond well with the correct prompt. While we have begun trying to address this problem by exploring different sampling methods, this remains an area for future investigation. Potentially, considering other image generation networks, such as GLIDE or DALL-E 2, could improve the quality of generated images.

The second issue, regarding ineffective distractor prompts, can be addressed by incorporating a fake prompt generation component into the game. Specifically, this fake prompt generator should produce fake prompts that could conceivable have led to the image shown, either by composing these prompts directly or by identifying such prompts from within the actual prompts in the database. We have considered two potential approaches for this fake prompt generation, but have identified significant shortcomings for both ideas.

The first idea was using an adversarial setup. Using a GAN setup, we can have a generator network generate fake prompts, while the discriminator tries to tell apart real and fake ones. Unfortunately, this approach does not work effectively because working with GANs on text-based data requires the generator and discriminator network to both be either recurrent or be a Transformer network. GANs and recurrent networks are known to suffer from several issues during training. Having an architecture that requires both of these paradigms to cooperate only makes training a *lot* more unstable. Training text-based GANs on small text datasets have not been shown yield good results yet. In the context of our game, even a perfect GAN might not generate the fake prompts that we are looking for. Such a GAN generator is going to try and generate fake prompts that try and mimic the *meaning* of the real prompt, when what we really want is a diverse set of prompt that could all plausibly represent what is in the image.

Another idea we considered for generating prompts was using a classifier architecture with a closed set of possible prompts. This would involve using CLIP to encode a set of prompts into their text embeddings, find the similarity between all those embeddings and the image's CLIP embedding, and select the top 3 closest prompts to the image as distractor prompts. We experimented with this approach briefly, and found that the main shortcoming of this was that the results were not significantly better than randomly sampling from the prompts. A contributing factor to this is that we used a relatively sparse set of possible prompts, so that all incorrect prompts were poor fits with any generated image. Using a larger set of classes for the classifier could yield better results.

One other aspiration for our project is to allow users to play the game in a free-form mode, where they are able to input a guess and a semantic scoring function is used to compute the player's accuracy. To implement this, one approach is to collect data from human game-play in multiple-choice mode to train a semantic scoring function.

Finally, there are a few further tasks that can be undertaken to improve the user experience of the game. Generating images for our entire data set of prompts, and updating the website backend, would allow us to include a wider variety of images and prompts, and also introduce themed category modes in the game.

7 Ethical Implications Statement

One concern associated with our final product is that the prompt options and generated images could potentially have ethical issues. For instance, the prompts might contain offensive or profane language if not controlled. Likewise, the images could include offensive symbols or graphic content. Additionally, the networks used to generate images could produce images that take certain objects, people, objects out of context to create content that is potentially inconsiderate, insensitive or offensive. For the current version of our website, we have carefully selected prompts and images so as to ensure these issues do not arise, but it is possible for such issues to occur with a larger dataset of prompts and images that is not manually curated and/or automatically censored.

Furthermore, the images generated by the networks can propagate certain harmful stereotypes. For example, when tasked with generating images referring to "a man", our networks usually

produced images of white people. As another hypothetical example, if the networks were tasked with generating images referring to "a doctor", they might produce images predominantly of male doctors. This possible ethical implication would also be especially important if semantic scoring was introduced - the semantic distance between "a man" and "a doctor" might be smaller than between "a woman" and "a doctor" according to the semantic scoring algorithm used, even if the doctor generated in the image was a woman. There's an interesting connection here between the diversity of samples and realism (or matching the distribution of the dataset), and the trade off between calibration and parity in counterfactual fairness. Just as parity issues arise from an underlying imbalance in the dataset, and calibration can be satisfied by varying thresholds, realism within image generation comes trying to resemble the dataset closely, whereas generating more creative and diverse samples requires us to deviate from the nature of the samples in the dataset.

Another concern is that as the image generation techniques get better, the line between real images and generated ones is blurred. Specifically, an image generated by our a neural network might closely resemble the style of a particular artist or photographer. However, this artist or photographer would not be credited alongside the image in our game. Additionally, the networks involved in our image generation process could reproduce trademarked logos and copyrighted graphics without appropriate licensing or attribution. The ML industry has not really figured out a way to avoid this problem [20], so it is one for further consideration.

One additional concern discussed in the DALL-E model card [20] is that our game could reinforce distrust in the general public about real images. Specifically, seeing how realistic AI-generated images look could result in players mistrusting real images.

Lastly, we should mention that generating images is a computationally intensive process, and game-play of users is associated with electricity usage. Both these aspects can contribute to climate change via energy consumption, and therefore have a minor negative impact on environmental sustainability.

8 Conclusion

The goal of our project was to determine if we can leverage AI image generation to create a fun and interesting game. We explored using OpenAI's CLIP as a scoring function in combination with either a SIREN network or BigGAN as the image generator. Specifically, an image was generated by the generator and then CLIP was used to compute a loss based on the similarity of that image's embedding to the input prompt's embedding. The loss was back-propagated through the network to optimize the generator. The image generation results we achieved allowed us to create a game that was at a reasonable difficulty and enjoyable to play, based on feedback from users who played our game. Therefore, our goal of creating a fun and interesting game from AI generated images was achieved.

References

[1] A. Ramesh, M. Pavlov, G. Goh, *et al.*, "Zero-shot text-to-image generation," *arXiv:2102.12092 [cs]*, Feb. 2021, arXiv: 2102.12092. [Online]. Available: http://arxiv.org/abs/2102.12092.

[2] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents," p. 26,

[3] A. Nichol, P. Dhariwal, A. Ramesh, *et al.*, "GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv:2112.10741 [cs]*, Mar. 2022, arXiv: 2112.10741. [Online]. Available: http://arxiv.org/abs/2112.10741.

[4] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. arXiv: 2103.00020. [Online]. Available: https://arxiv.org/abs/2103.00020.

[5] P. Wang, *Lucidrains/deep-daze*, Jan. 2022. [Online]. Available: https://github.com/lucidrains/deep-daze.

[6] ——, *Lucidrains/big-sleep*, Jan. 2022. [Online]. Available: https://github.com/lucidrains/big-sleep.

[7] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *CoRR*, vol. abs/1711.00937, 2017. arXiv: 1711.00937. [Online]. Available: http://arxiv.org/abs/1711.00937.

[8] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *CoRR*, vol. abs/2006.09661, 2020. arXiv: 2006.09661. [Online]. Available: https://arxiv.org/abs/2006.09661.

[9] A. Brock, J. Donahue, and K. Simonyan, *Large scale GAN training for high fidelity natural image synthesis*, 2019. arXiv: 1809.11096 [cs.LG].

[10] G. Developers, *Common problems*.

[11] T. Kim and Y. Bengio, "Deep directed generative models with energy-based probability estimation," *arXiv preprint arXiv:1606.03439*, 2016.

[12] A. E. Gelfand, "Gibbs sampling," *Journal of the American Statistical Association*, vol. 95, no. 452, pp. 1300–1304, 2000.

[13] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[14] *Hopfield network*, en, Mar. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hopfield\%5Fnetwork\&oldid=1075856675 (visited on 03/11/2022).

[15] *Restricted Boltzmann machine*, en, Nov. 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Restricted\%5FBoltzmann\%5Fmachine\&oldid=1057456034 (visited on 03/11/2022).

[16] *Deep belief network*, en, Mar. 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Deep\%5Fbelief\%5Fnetwork\&oldid=1012688873 (visited on 03/11/2022).

[17] *Star Trek episodes*, May 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Lists\%5Fof\%5FStar\%5FTrek\%5Fepisodes\&oldid=1024306670.

[18] The Beatles Bible, Oct. 2021. [Online]. Available: https://www.beatlesbible.com/songs/.

[19] AAAI, *AAAI Classic Paper Award*. [Online]. Available: https://aaai.org/Awards/classic.php.

[20] P. Mishkin, L. Ahmad, M. Brundage, G. Krueger, and G. Sastry, "DALL-E 2 preview: Risks and limitations," 2022. [Online]. Available: https://github.com/openai/dalle-2-preview/blob/main/system-card.md.