Generating Pokémon Using Stable Diffusion
CS680 Project Proposal

Ali ElSaid
Department of Statistics and Actuarial Science
University of Waterloo
ali.elsaid@uwaterloo.ca

Abstract

I attempt to use generative models, namely Stable Diffusion (SD), to generate new fake Pokémon. GANs was first explored as an option but the results from previous similar works were unsatisfactory, therefore SD was decided to be the main tool used since it seems to have high potential. Initial attempts at generation using the online demo of Stable Diffusion on *huggingface.com* yielded bad results. SD v1.5 was than fine-tuned using a dataset of captioned images of 800+ Pokémon [1] which produced good results that resembled Pokémon and were somewhat generalizable. But the quality, detail and originality wasn't great. Then I went back to trying to create better prompts using the default SD v1.5 to generate higher quality images. This produced surprisingly good results albeit generating somewhat generic creatures which sometimes can pass as Pokémon and sometimes not really; the quality was also inconsistent, sometimes it was great sometimes not. Results were also very hit or miss depending on seed. I then learned about a new technique called Textual Inversion (TI) [2] and found four pretrained embeddings for Pokémon using TI on huggingface [3][4][5][6]. I retrained one that had modern concept art for all the recent 900+ Pokémon [3] to get to know the process of Textual Inversion and train my own embedding. I then attempted to use the four embeddings over SD v1.5 as styles to generate creatures in the style of Pokémon; the results were mediocre at best. I then used Textual Inversion to train my own embedding over SD v1.5 for just one Pokémon "Lucario" in an attempt to get good results by focusing the training on only one Pokémon's features; this presented mediorcly good results. After using some of the same previous good prompts on it, the results were also great. At the end of it all, I jumped back into the prompt engineering lab and attempted to generate the best results I can with SD v1.5 and another version of SD v1.5 called Anything-V3 which is fine-tuned on 19M anime style pictures. The final results were beautiful.

1 Introduction

Pokémon is the single highest grossing media franchise in history worldwide, accumulating over $118.5 billion in revenue since its year of inception, 1996, across video games, anime, movies, merchandise and more. As of today, there are 905 unique Pokémon (short for Pocket Monsters) and tremendous effort is spent into designing each and every one of them. Wouldn't it be great if we were able to generate completely new Pokémon with just the click of a button or a simple short description? This is where the currently widely trending generative deep learning models come in.
The goal of this project is to use the Stable Diffusion model to generate new Fake-mons (Fake Pokémon) from short text prompts that are as high quality as possible and that can be mistaken for deliberately designed real Pokémon. I am a huge Pokémon fan and I grew up playing and watching it, and I still do until now. So, the thought of creating my own new Pokémon in a matter of minutes excites me. This will also be my first experience with deep learning, so I am using this project as a

gateway into deeply learning the world of deep learning.
Aside from my personal goals, this project could be eventually made in the future into a website that provides an interface for Pokémon fans (especially children) that enables them to create their very own Pokémon. These newly generated Pokémon could also serve as a seed for new design ideas for the creators of Pokémon themselves, easing the design process and enabling the generation of novel designs and combinations they would have never thought of. Also, the model could be used to generate beautiful AI generated artwork for people to enjoy and use as their backgrounds or other similar usages.

2 Literature Review

As far as academic papers go, there are not many that have explored the topic of Pokémon generation. I have seen two papers that are related to Pokémon and deep learning. The first [7] is a paper that attempts to generate new fake Pokémon trading cards (for the Pokémon trading card game (PTCG)) using deep learning. The authors used Generative Adversarial Networks (GANs) and Conditional Adversarial Networks to generate the images, and they used Char-Recurrent Neural Networks (RNNs) to generate the text for the cards; their results were mediocre. The second paper [8] applies style transfer techniques using Convolutional Neural Networks (CNNs) and GANs (CartoonGAN, CycleGAN and StyleGAN) to transform the style of photos of pets to be in the style of Pokémon; the results of this paper weren't great. The main contributions I found for Pokémon generation using deep learning were from articles on the internet. Most of them trained variations of GANs from scratch to generate new Pokémon images, however most of their performance was not satisfactory; the Pokémon images were too blurry and had no distinct features (References [3] to [9]) see Figure 1.

Based on Professor Yu's recommendation I began looking into Diffusion models (specifically Stable Diffusion), I found an article [9] that fine tunes the Stable Diffusion model on Pokémon images to have it generate Pokémon-style images from prompts (for example, if the prompt is "Donald Trump", the model will generate an image of Donald Trump in the form of a Pokémon, see Figure 2).

This was the closest attempt I found to what I aim to do, and the most successful one too. I also found a bunch of articles that are unrelated to Pokémon, but describe how to fine tune the Stable Diffusion model using multiple methods, one of which is called Textual Inversion [10] which I will be using as references to create my own version of fine tuning Stable Diffusion.

Figure 1: Results of using GANs to generate Pokémon.

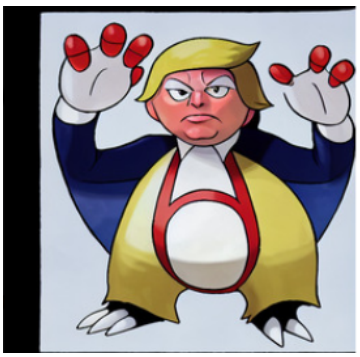

Figure 2: Donald Trump as a Pokémon

3 Experiments & Results

I have conducted a few experiments of generating Pokémon using Stable Diffusion (SD). The next section will describe the experiments conducted. I will be using the Pokémon Greninja 4 and Bul-

basaur 3 as my baseline; I will be generating both of those Pokémon in every experiment using the same settings (except for the part that needs to be changed to conduct the experiment) to be able to most accurately determine the actual effect of the experiment. The aforementioned baseline settings for generating images using the SD model are as follows {Image Size: 512x512, Seed: 3118917620, CFG Scale: 7, De-noising Steps: 20, Sampler: Euler a}. The following is the explanation of these settings [11]. Image Size is how many pixels by pixels the output image will be. Seed is the number used to seed the randomization that will generate the image of noise that will then be de-noised or de"diffused" to generate the output image. CFG Scale is how intensely the model will condition on your text prompt to generate the output image, i.e. how much the model will try to make the output image follow your prompt. De-noising Steps is basically how many discretizations of the differential equation will be used and solved to de-noise the image, i.e. how many layers of de-noising the noise image will go through to generate the output image. The Sampler is basically the method of solving the differential equations along the way, and different methods result in slightly different images [12]. Before doing any of the experiments, I researched both GANs and Stable Diffusion and the previous similar works using these two models. After that I started with the Stable Diffusion experiments since it was the more powerful one according to the survey of previous works.

3.1 Experiment 1

The first experiment consisted of generation using basic prompts on Stable Diffusion's online demo version. Figures 5 and 6 represent the two initial generated images of (supposedly) Bulbasaur and Greninja respectively. I will be using these as a baseline to judge the progress of generation in the next experiments. The prompts used were "Bulbasaur" and "Greninja".

Figure 3: Bulbasaur

Figure 4: Greninja

Figure 5: Prompt: "Bulbasaur"

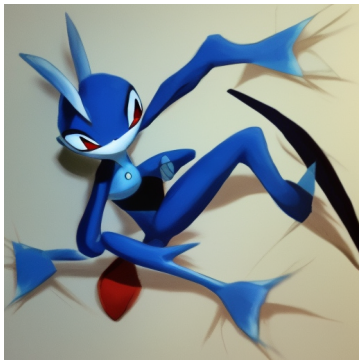

Figure 6: Prompt: "Greninja"

3.2 Experiment 2

In the second experiment I used Justin Pinkney's article [9] as a reference to fine-tune Stable Diffusion on captioned images of 800+ Pokémon (similar to the above images of Bulbasaur and Greninja).

This was done on an instance of the 2xA6000 machine on the Lambda Cloud computing platform. The goal of this was to have Stable Diffusion generate images of new Pokémon from prompts of potentially unrelated things. The following are some of the results of generating images after the fine-tuning. Both Bulbasaur 7 and Greninja 8 as the baselines. The results do have some Bulbasaur and Greninja features but don't really resemble them much. That is most likely because the captioning of the images was done automatically using BLIP and the captions don't include the names of the Pokémon. Therefore the Bulbasaur features for exampmle are most likely the features learned by the original model before fine-tuning, and the fine-tuning directed it to be in the style of Pokémon. Prompt "Spider" was used to generate a Spider Pokémon 10, which I would say looks mostly good except for the fact that it doesn't have much detail and alot of its features, especially the face and round things on its back, are taken directly from the Pokémon Paras which resembles a bug/crab fusion. And finally Kratos from God of War as a Pokémon with prompt "Kratos holding the Blades of Chaos" 9. This is the result I like the most and I'm surprised that the model actually generated his beard. Overall, the results were good in terms of 1. generating images in the Pokémon style and 2. generating images resembling Pokémon and having their features and 3. merging unrelated concepts with the concept of Pokémon to create things like Kratos and Donald Trump as Pokémon. However they were lacking in terms of detail, quality and originality. As the outputs were somewhat simplistic, had many artifacts and one who is familiar with Pokémon can easily point out specific features taken from specific Pokémon in an overt manner.

Figure 7: Prompt: "Bulbasaur"

Figure 8: Prompt: "Greninja"

Figure 9: Prompt: "Kratos holding the Blades of Chaos"

Figure 10: Prompt: "Spider"

3.3 Experiment 3

After not being able to get higher quality results form the fine-tuned version of SD, I decided to go back to the base version and try more prompt engineering to try to get better results. However this was not feasible on the online demo since its takes very long to generate images and is neither convenient not customizable enough. So I downloaded SD version 1.5 and both the invokeai and Automatic1111 web UIs on my machine to be able to run the model locally. I then began exploring

the internet, Reddit and Stable Diffusion Discord servers for tips on how to create better prompts for stable diffusion. I began playing around with prompts and parameters to get a feel for how things work. The first thing I learned was about including keywords that portray the effect you want in the image. For example, if you want the image to be a 3D render you could add the word "3D" to your prompt. If you would like the image to be high quality or highly detailed you could add those exact phrases as keywords in the prompt. These words are separated from the main prompt by commas so that the model understands that they are different concepts from the main prompt. And they generally come after the main prompt so that the main prompt has more importance. The following are the prompts used for Bulbasaur and Greninja in that form: "a <name> pokemon, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d" where <name> is either Bulbasaur or Greninja; see figure 11 and 12. I used the Pokémon keyword to reinforce the Pokémon concept and other keywords for quality and for getting a 3D render. The artstation keyword is used to bias the model towards art that it learned from the website artstion which generally has good and detailed art. The Bulbasaur result is good but the Greninja one doesn't exactly resemble it albeit having some of its features.

Figure 11

Figure 12

Another learning was how to differently weigh keywords in a prompt. Adding round parentheses around a keyword multiplies its importance by 1.1, and you can nest parentheses inside each other to increase the importance even further. So to add importance to Bulbasaur and Greninja and accentuate their details, I added 3 parentheses around their names. The prompts used are: "a (((greninja))) pokemon, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d"; see figure 13 and 14. The results are very similar to the previous ones without the weighting, but they have more details, look slightly better and resemble the original Pokémon better.

Figure 13

Figure 14

Since one of the original goals was generating completely new Pokémon, I began experimenting with prompts that attempted to generate new designs that aren't necessarily aimed towards replicating a specific Pokémon. I started with the following prompt: "A white quadrupedal Pokémon, glowing pink crystal ball on its head, sharp claws, spring for legs, ornate, dynamic, particulate, intricate,

elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d"; figure 15. Looks great, infact it is the best result so far, however it totally ignored the "spring for legs" part. So I removed it; figure 16. However for some reason (probably related to the removal of the word "legs") that resulted in it not being a quadrupedal anymore. So I added weight to the word quadrupedal by changing "quadrupedal" to "(quadrupedal)"; figure 17. That did not work, so I added one more pair of parantheses; figure 18. Again, did not work, so added more weight; figure 19. Finally that worked, but the balls are gone. Therefore, I added weight to the ball word; figure 20. This seems like it added to much importance and replace the whole face with a silver ball. So I removed the parentheses from "ball" and put them around "glowing pink crystal ball on its head"; figure 21. This creature in the final result is great and extremely close to the desired design! (I'll call him Whitey for the rest of the report). This demonstrates that the base SD can generate creatures that can pass as Pokémon using a moderate amount of prompt engineering, albeit not having the iconic style or features of Pokémon. Another realization from this was that sometimes the output image responds very unexpecetedly/unnaturally in response to changes in the prompt. And also the images change wildly in quality from one seed to another.

Figure 15

Figure 16

Figure 17

Figure 18

3.4 Experiment 4

The next experiment is using a technique called Textual Inversion to fine-tune the model to help it better recreate a set of image's distinct features; it is a technique for capturing novel concepts from a small number of example training images [2]. To use Textual Inversion we first specify an arbitrary unique token, for example "lucario680". Then it learns a new embedding for that token by taking one of the example images and noisy versions of it, then the SD model attempts to predict the denoised version of that image in conjunction with a prompt that includes that arbitrary token. The embedding is then adjusted based on how well the model does in producing a denoised image close to the original one. If the model does poorly, then the embedding for that arbitrary does not contain enough useful information yet to represent the concept in the training example images. On the other side, an embedding that captures the concept of the training images well will enable the model to

Figure 19

Figure 20

Figure 21

produce an image close to the real image [13].

I found four pre-trained embeddings on huggingface for Pokémon. I first tried to re-train one of them and then generate results using all four. The first embedding was one trained on battle images of Pokémon [3] (Galar Embedding) figure 22, second on Pokémon concept art [4] (Modern Embedding) figure 23, third an classic Pokémon concept art [5] (Classic Embedding) figure 24, fourth on old pixel art sprites of Pokémon [6] (Sprite Embedding) figure 25. See figures for comparison of images of Bulbasaur in the four embeddings' training data.

Figure 22: Galar Embedding

Figure 23: Modern Embedding

Figure 24: Classic Embedding

Figure 25: Sprite Embedding

I then generated images of Bulbasaur and Greninja for each of the four embeddings with prompts in the form of "A <name> Pokémon in the style of <embedding-name>" where <name> is either Bulbasaur or Greninja and <embedding-name> specifies one of the four embeddings used to style the image. Below are the results. The results are not good, however it is really clear that the embeddings styles are learned and present in the images.

I then attempted to create the previous great result from Experiment 3 but in the different embeddings styles with a prompt of the form "A white (quadrupedal)+ Pokemon, glowing pink ball on top of its head, sharp claws in the style of <embedding-name>, ornate, dynamic, particulate, intricate,

Figure 26: Bulbasaur in the style of the Galar Embedding

Figure 27: Bulbasaur in the style of the Modern Embedding

Figure 28: Bulbasaur in the style of the Classic Embedding

Figure 29: Bulbasaur in the style of the Sprite Embedding

Figure 30: Greninja in the style of the Galar Embedding

Figure 31: Greninja in the style of the Modern Embedding

Figure 32: Greninja in the style of the Classic Embedding

Figure 33: Greninja in the style of the Sprite Embedding

elegant, highly detailed, centered, artstation, smooth, sharp focus". I removed the 3D parts since 3D is inherently a different style than the embeddings. The results are below. These ones are much better than the previous ones most likely because of the more detailed description and inclusion of the characteristics keywords.

Figure 34: Whitey in the style of the Galar Embedding

Figure 35: Whitey in the style of the Modern Embedding

Figure 36: Whitey in the style of the Classic Embedding

Figure 37: Whitey in the style of the Sprite Embedding

3.5 Experiment 5

In this experiment, I trained my own embedding for the Pokémon Lucario (see figure 38) locally on my machine. This was feasible since training an embedding is not as expensive as fine-tuning or training the whole model. I trained the embedding on 8 images of Lucario, one of which is below in figure 38. This embedding is different from the previous ones since it is trained on much fewer images and of one subject only. So while the previous embeddings were "style" embeddings, this one is a "subject" embedding, i.e. when using it in prompts the usage should be as a subject in the sentence and not a style. Below are three of the results of this embeddings training. Prompt 1: "A (lucariov003) Pokemon, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus" (lucariov003 is the name of the embedding); figure 39. Prompt 2: "A (lucariov003) Pokemon firing an aura sphere, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus"; figure 40. Prompt 3: "A white (lucariov003) Pokemon, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d"; figure 41. The results are okay, but they could be alot better. They definitely show however that the model learned the embedding somewhat since they all resemble Lucario and Lucrio was not mentioned anytime explicitly in any of the prompts. My favorite is the last one. "

Figure 38: Lucario

Figure 39

Figure 40

3.6 Experiment 6

The last experiment was all about going back to the prompt engineering lab and attempting to get the best possible results from the base SD v1.5 using prompt engineering. One of the biggest things that I learned about that boosted the quality of the generations was negative prompting; you write keywords that you **do not** want to be represented in the output image. This is extremely useful since you can write some keywords in the negative like "ugly" and "low quality" to avoid those concepts in your generations. Another great learning was that you can write "art by <artist-name>" and it would attempt to generate art in the style of the artist specified. You can also combine different artists. Combining all these learnings along with playing with different parameter values and searching for good seeds resulted in the below awesome final results. All the generations below have a negative prompt of: "ugly, tiling, abnormal, poorly drawn hands, poorly drawn feet, poorly drawn face, out of frame, mutation, mutated, extra limbs, extra legs, extra arms, disfigured, deformed, cross-eye, body out of from, blurry, bad art, bad anatomy, blurred, text, watermark, grainy, weird colors,

Figure 41

blind, worst quality, low quality, normal quality, oversaturated, low-res, kitsch, disconnected limbs, floating limbs, long neck, long body". First two images (43, 42) are the baseline Pokémon Bulbasaur and Greninja. It seems like they have come a long way from the first generation! The green monkey is a generation of Anything-V3, a different model which I was trying in the end, which also generated great outputs. Anything-V3 a version of SD v1.5 that is fine-tuned on anime pictures and that can be attributed to why it might be generating good results much easier than SD v1.5. The rest is miscellaneous experimentation with image generation.

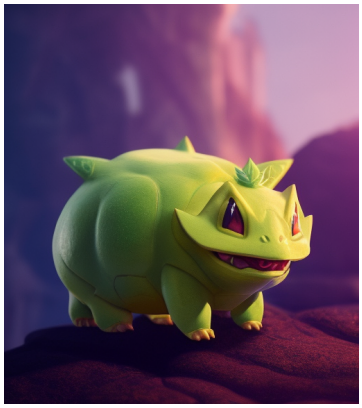

Figure 42: "a (bulbasaur) pokemon by beeple and James Gilleard and Justin Gerard, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d"

Figure 43: "a (greninja) pokemon by beeple and James Gilleard and Justin Gerard, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d"

Figure 44: "A grookey (Pokemon) by beeple and James Gilleard and Justin Gerard"

Figure 45: "a Sobble pokemon by beeple and James Gilleard and Justin Gerard, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d"

4 Conclusion

In summary, I learned a lot about deep generative models and Stable Diffusion in specific. I learned how to fine-tune SD and about Textual Inversion and embeddings and how to train them in addition to being exposed to many other ideas throughout my research (which I will mention in the future work section). I also experimented a great deal with all the previously mentioned parameters of the model generation and was able to go from low quality generations to extremely high quality ones that could pass for real Pokémon.

Figure 46: “a grimmsnarl pokemon by beeple and James Gilleard and Justin Gerard, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d”

Figure 47: “a sobble pokemon by beeple and James Gilleard and Justin Gerard, ornate, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d” (variation)

Figure 48: ‘a chansey by samdoesart and James Gilleard and Justin Gerard, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d, 4K, 8K, ultra HD”

Figure 49: “a chansey pokemon coming out of its ((pokeball)) and hitting brock by samdoesart and James Gilleard and Justin Gerard, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d, 4K, 8K, ultra HD” (kind of a random result)

5 Future Work

Future work includes: More experimentation with Anything-V3 model and other large-scale fine-tuned versions of SD. Experimentation with img2img features that can use a base image and then use the prompt and the generation of the model to modify that base image. Explore the outpainting feature, where you draw over a section of the image and you can generate new things inside that area with new prompts. Experiment with Dreambooth as a fine-tuning alternative to Textual Inversion, which could work much better for the case of fine-tuning the model on only one Pokémon. Research and experiment with Textual Inversion more and try to train better embeddings. And lastl but definitely not least, more prompt engineering and learning how to make better and better prompts.

Figure 50: ‘a ((pokeball)) and hitting brock, by samdoesart and James Gilleard and Justin Gerard, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d, 4K, 8K, ultra HD”

Figure 51: “a ((pokemon monster)), by samdoesart and James Gilleard and Justin Gerard, dynamic, particulate, intricate, elegant, highly detailed, centered, artstation, smooth, sharp focus, octane render, 3d, 4K, 8K, ultra HD”

Acknowledgement

I would like to thank Professor Yu for suggesting Stable Diffusion as an alternative to GANs and generally for the amazing course! I would also like to thank my cousins Yousof and Amr for helping me brainstorm better prompts and giving me feedback on the generated output images.

References

[1] Justin N. M. Pinkney. *Pokemon BLIP captions*. https://huggingface.co/datasets/lambdalabs/pokemon-blip-captions/. 2022 (cit. on p. 1).

[2] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. *An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion*. 2022 (cit. on pp. 1, 6).

[3] fireYtail. *SD-concepts-library/pokemon-gens-1-to-8 · hugging face*. 2022 (cit. on pp. 1, 7).

[4] fireYtail. *SD-concepts-library/pokemon-modern-artwork · hugging face*. 2022 (cit. on pp. 1, 7).

[5] fireYtail. *SD-concepts-library/pokemon-classic-artwork · hugging face*. 2022 (cit. on pp. 1, 7).

[6] fireYtail. *Commits · SD-concepts-library/pokemon-rgby-sprite*. 2022 (cit. on pp. 1, 7).

[7] Michael B. Hedge, Morgan Nelson, Thomas Pengilly, and Michael Weatherford. “PokéGAN: P2P (Pet to Pokémon) Stylizer”. *SMU Data Science Review*. 10th ser., vol. 5, no. 2 (2021) (cit. on p. 2).

[8] Devi Acharya, Rehaf Aljammaz, Beth Oliver, and Mirek Stolee. *Gotta Generate ’em All! Pokemon (With Deep Learning)*. 2019 (cit. on p. 2).

[9] Justin Pinkney. *How to fine tune stable diffusion: How we made the text-to-pokemon model at lambda*. Oct. 2022 (cit. on pp. 2, 3).

[10] Foong. *How to fine-tune stable diffusion using textual inversion*. 2022 (cit. on p. 2).

[11] *SD hypertextbook*. 2022 (cit. on p. 3).

[12] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. *Elucidating the Design Space of Diffusion-Based Generative Models*. 2022 (cit. on p. 3).

[13] huggingface. *Textual inversion*. 2022 (cit. on p. 7).

[14] Justin Kleiber. *Pokegan: Generating fake pokemon with a generative adversarial network ...* June 2020.

[15] Tyler Folkman. *How To Create Unique Pokémon Using GANs — by Tyler Folkman — Towards ...* 2020.

[16] Digvijay Yadav. *Pokemon Generator Gans*. July 2020.
[17] theRoughCode. *TheRoughCode/pokegan: Pokemon generator using the power of gans.*
[18] Arham Lodha. *My journey through gans for pokemon generation*. July 2020.
[19] Menghanibhanvi. *Pokemon Generation Using Gans (pytorch)*. Nov. 2020.
[20] Mehta. *Generating New Pokemons using Gans — by Jatin Mehta — becoming ... - medium*.
[21] Rayfield. *AI generated pokemon sprites with GPT-2*. 2020.
[22] LambdaLabs. *Lambdalabs/SD-image-variations-diffusers · hugging face*.
[23] ExponentialCookie. *R/stablediffusion - [tutorial] "fine tuning" stable diffusion using only 5 images using textual inversion.*
[24] zoru22. *R/stablediffusion - I got stable diffusion to generate competent-ish leavannies w/ textual inversion!*
[25] External_Quarter. *R/stablediffusion - let's discuss best practices for finetuning.*
[26] P.w. *Fine tuning stable diffusion images with cross attention control*. Sept. 2022.
[27] Joy Zhang. *I spent $15 in dall·e 2 credits creating this AI image, and ... - medium*. 2022.
[28] Kvpratama. *Pokemon Images Dataset*. Aug. 2020.
[29] Lambda. *Lambdalabs/pokemon-blip-captions · datasets at hugging face*.
[30] llSourcell. *Pokemon GAN/data at master llSourcell/pokemon gan*.
[31] PokeAPI. *Sprites/sprites/pokemon at master pokeapi/sprites*.
[32] *Pokémon sprites: Archive of pokémon images from every game*.
[33] *Sprite packs*.
[34] AI Tutorials. *Embeddings in stable diffusion (locally)*. Oct. 2022.
[35] AUTOMATIC1111. *AUTOMATIC1111/stable-diffusion-webui-feature-showcase: Feature showcase for stable-diffusion-webui*.
[36] AUTOMATIC1111. *Negative prompt · automatic1111/stable-diffusion-webui wiki*.
[37] AUTOMATIC1111. *Textual inversion · automatic1111/stable-diffusion-webui wiki*.
[38] Benny Cheung. *Stable diffusion training for personal embedding*. 2022.
[39] Mohamad Diab. *Cdn.openart.ai*. 2022.
[40] Jennifer Doebelin. *How to train textual inversion - stable diffusion ai — deepfake yourself! embeddings and low memory*. Oct. 2022.
[41] Linaqruf. *Linaqruf/Anything-v3.0 · hugging face*. 2022.
[42] Mauwii. *Overview*. Dec. 2022.
[43] Mauwii. *Textual inversion*. Oct. 2022.
[44] *SD Resource Goldmine*. 2022.
[45] u/pxan. *R/stablediffusion - how to get images that don't suck: A beginner/intermediate guide to getting cool images from stable diffusion*. 2022.
[46] u/stinkykoala314. *R/stablediffusion - parameters for textual inversion on automatic1111?* 2022.
[47] u/ts4m8r. *R/stablediffusion - what is the difference between each sampling method?* 2022.

6 Appendix

This appendix will include some of the best final results that I could not include in the main report.

Figure 52

Figure 53

Figure 54

Figure 55

Figure 56

Figure 57

Figure 58

Figure 59

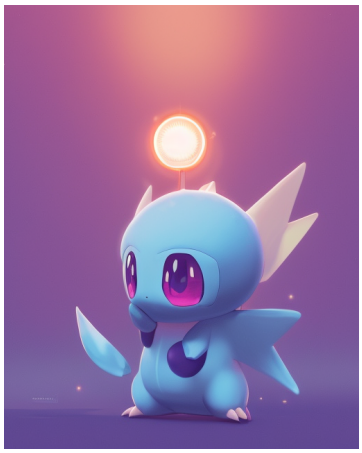

Figure 60

Figure 61

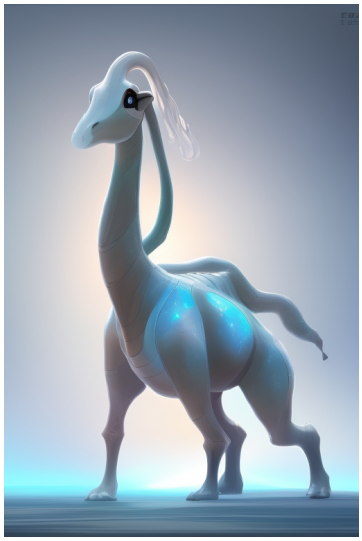

Figure 62

Figure 63

Figure 64

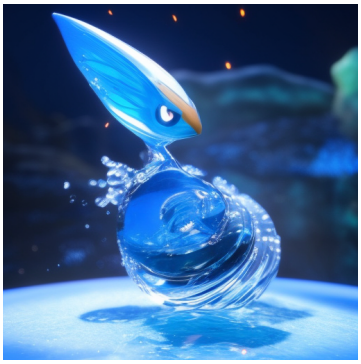

Figure 65

Figure 66

Figure 67

Figure 68

Figure 69

Figure 70

Figure 71

Figure 72

Figure 73

Figure 74

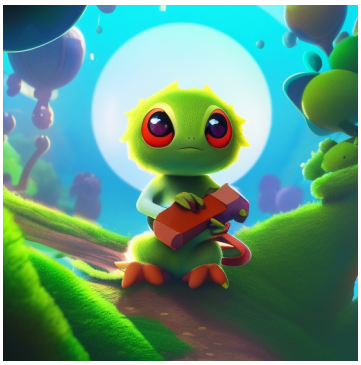

Figure 75

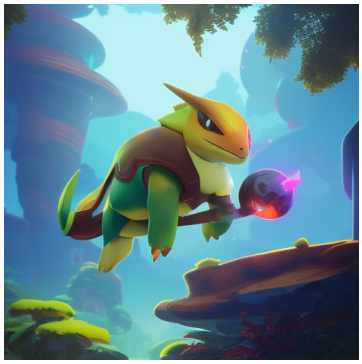

Figure 76

Figure 77

Figure 78

Figure 79

Figure 80

Figure 81

Figure 82

Figure 83