Text-to-image with artistic style generation

Vincent Gariepy
University of Montreal
vincent.gariepy.1@umontreal.ca

Enzo Daval
University of Montreal
enzo.daval@umontreal.ca

Adrien Chabaud
University of Montreal
adrien.chabaud@umontreal.ca

Abstract

This paper investigates text-to-image machine learning models that have an extra added feature of also letting the user select an artistic style to copy to the generated image. This has the advantage of making the images look much nicer, as generally even the best trained text-to-image models have some noticeable flaws in their images. Adding an artistic filter would brush out the flaws to make the image more appealing to the viewer. With our project we train a model on a brand new dataset of annotated images that also have some art style applied to them. The results are being compared with a baseline model of a pre-trained text-to-image model followed by a pre-trained art style transfer model. Although our training ressources were limited, our results are promising that trained on the correct dataset, a text-to-image model can also learn artistic style from text only. All of the code necessary to creating our dataset, training our model and testing it versus our baseline is available in our Github repository (Project).

1 Introduction

The inspiration for our project comes from the website : **Dream by WOMBO**[1]. It lets the user generate any image using words to describe it, while also having 17 different art styles to choose from. Since this project is not open source, the focus will be to be able to recreate it. This project is inspiring since after looking at different text-to-image models, we have found that although they all thrive to get the most realistic image that matches the most accurately the input text, they leave behind beauty in most cases. This is why we think combining a text-to-image model with an artistic style transfer model, will help us create a model that can generate beautiful images.

There are many ways to approach this type of text-to-image generation and we thought it would be interesting to implement two different methods. The first method would simply be to use two different pre-trained models, one of them for text-to-image generation and the other for art style transfer to this image. The next method would be to train a text-to-image model on a new dataset that we create. This dataset is composed of annotated images that already have an art style applied to them. Since our training time and resources are limited we have a *small dataset of 3000 airplane images with only 4 different art styles to choose from*. Within this dataset we have set aside a small percentage of the images as a test set. We will apply the test set to both models and see if our own trained model can produce similar results to a robust concatenation of pre-trained two models. The goal of this would be to show that it is possible to train a single model to generate art style images from text, and that with more time and better hardware, the model could be trained on a bigger dataset of all types of images to generate any type of image, not just airplanes.

Due to limited resources, we have partially trained the a zero-shot Text-to-image model from the work of (Ramesh et al., 2021) with 3 different depths : 2, 6 and 12 layers. Although our dataset was small and training was limited, early results are promising with the the model clearly distinguishing the different art styles and produces nicer artwork the more layers there are.

2 Related work

Machine Learning approaches to text to image synthesis mainly started with the work of (Mansimov et al., 2015) whose aim was to introduce a model generating images from natural language captions. As a base to generate images, the authors used the Deep Recurrent Attention Writer (**DRAW**) introduced by (Gregor et al., 2015) which aim

[1] https://app.wombo.art/

was to combine novel spatial attention mechanism mimicking the foveation of the human eye with a sequential variational auto-encoding framework allowing an iterative construction of complex images like a person would do if asked to draw or paint something. In their paper (Mansimov et al., 2015) extended the DRAW model to iteratively draw patches with an attention to the relevant words given in the description. Once generated, the images are refined by a *deterministic Laplacian pyramid adversarial network* explained by (Denton et al., 2015). Based on those works (Reed et al., 2016) demonstrated that using a generative adversarial network rather than a recurrent variational auto-encoder was improving image fidelity but also zero-shot generalize to held-out categories. Based on those methods (Ramesh et al., 2021) use a transformer that autoregressive models text and image tokens as a single stream of data, they assume that with a sufficient data and scale, their approach is quite competitive to domain-specific models when evaluated in a zero-shot fashion.

When looking for a text-to-image neural network model, the work of (Nichol et al., 2021) on a brand new model called **GLIDE** was very interesting. The authors introduce a new approach that is innovative in many ways to help achieve photo-realism image creation from text. Also this model is faster than all the others tried (*1 minutes to generate an image, against 10 minutes for other models*). With all these factors it will play a crucial role in our first model as it will be able to generate many images quickly to be able to be evaluated on a test set without too much computation time.

With regards to artistic style transfer, a paper by (Lin et al., 2021a) goes over a very interesting approach called *Laplacian Pyramid Network*. We chose this paper because it deeply describes how artistic transfer works, how it can be implemented and trained, and which is the best model according to it. It has a lot of value for our project because it processes many parts around artistic style transfer which is one of the two main parts of the model we will try to implement. For example, we can find a part dedicated to the datasets details, which is crucial because we have to understand how to set up our own dataset, or another part on the comparison of the different existing methods of transfer, allowing us to choose the one that best fits our needs. Finally the paper concludes by giving us, according to its different tests, which method and model is the best for artistic style transfer.

3 Methods

The model we used for the text-to-image part was the GLIDE's one (Nichol et al., 2021). We used notebooks provided on the Github page (GLIDE) of the model to understand how it works and to implement it in our project. GLIDE (Guided Language to Image Diffusion for generation and Editing) is a model composed of diffusion which had noise to the initial picture; Classifier-free guidance which is a technique for guiding diffusion models that does not require a separate classifier model to be trained; And CLIP guidance which is an approach for learning joint representation between text and images, with the dot product of the image encoder and the caption encoder.

"an old car in a snowy forest"

Figure 1: An example of the the inpainting skill of the GLIDE model.

One of the strengths of this model is its inpainting skills. In fact, unlike most of the text-to-images models, GLIDE doesn't create images but merges different ones to satisfy the prompt. For example, the model is able to recognize which word of the prompt is the main topic, and will choose an image of its dataset which corresponds to the main topic. Then, the model will modify this image according to the other elements of the prompt. It is able to recognize which pixel is from which element of the image, so it can easily modify the initial image. The model can then quickly modify the background of an image or add new elements (see figure 1.

The model used for the art style part comes from (Lin et al., 2021b), it uses an approach

called Laplacian Pyramid Network. We used the hands-on code from the associated github page (PaddleGan, a) allowing us to easily use the pre-trained models. They also explain how to train our own models (PaddleGan, b) but this approach has been moved aside and the decision to move on with pre-trained models has been taken.

Since the main part of our project is to be able to have a single model that can do text-to-image with and art-style, the choice of our model would be crucial. After, many tries and different models briefly tested we decided to go forward with the Zero-shot Text-to-Image generation model from (Ramesh et al., 2021), also called the DALL-E model. Their approach is very interesting since it is a two step method that has a VAE (variational auto-encoder) and a transformer. The VAE is used to compress each image into a grid of 32 x 32 x 8192 image tokens. This reduces the context size of the transformer by a factor of 192 without a large degradation in visual quality. They then concatenate the encoded text tokens with the encoded image, which is passed to the autoregressive transformer as a single stream of data (see figure 3). The transformer can model the joint distribution over the text and image tokens. With this type of approach learning the relation between the text and image is much simpler for the model.The VAE has an encoder-decoder architecture while the transformer is a decoder only sparse transformer.

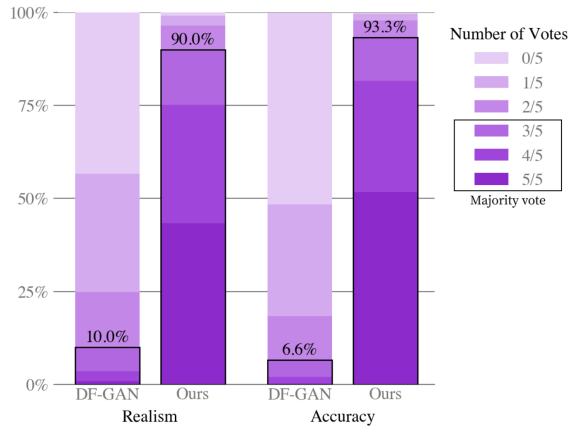


Figure 2: Human evaluation of the model (evaluated zero-shot without temperature reduction) vs DF-GAN on captions from COCO dataset.

Both the VAE and the transformer are trained separately on the same dataset. Doing so ensures that each part of the model performs well, which is the case. Multiple test show that the new model outperforms most models when tested in a zero-shot fashion. When tested with human evaluation as compared to DF-GAN by (Koh et al., 2020), the DALL-E model was found the most realistic looking image 90.0% of the time and the best matching image 93.3% of the time (see figure 2). To train this model we based our code off their official github page (DALL-E).

4 Experiments

4.1 Dataset

Since the goal of our project is to train one model that can do text-to-image generation with art style transfer, we would need a database of labeled data that has an art style applied to them. However, as expected, this sort of dataset does not explicitly exist so one main part of our project would be to create such a dataset to be able to train our model on it. Since our training resources (time and hardware) were limited we did not want to train our model on a big dataset such as COCO dataset (COCO - Common Objects in Context)[2] with over 200k labeled images. We decided we would try and train our model on a specific category of images, such as “cats” or “dogs”, this would make our dataset smaller and more specific. This will help us show that if this model can generate art style images of this category then it can be extrapolated and trained on all categories to generate any art style image. After going through many categories of the COCO dataset we have decided to pick the “airplane” category, since it has just under 3000 images that are very clear and well labeled.

After having picked our images, we have to apply an art style to them so our model can be trained on them. We used code from a Google Collab script from PaddleGAN (PaddleGan, a) that applied on of the following four art styles to any image : “circuit”, “ocean”, “starrynew”, “stars”. The pre-trained model is based on the Laplacian Pyramid Network by (Lin et al., 2021a). By modifying the code slightly, we were able to iteratively go through all the images and randomly pick a style to apply to them. We also changed the image captions to include the type of art style in words so our model does not need a reference

[2] https://cocodataset.org

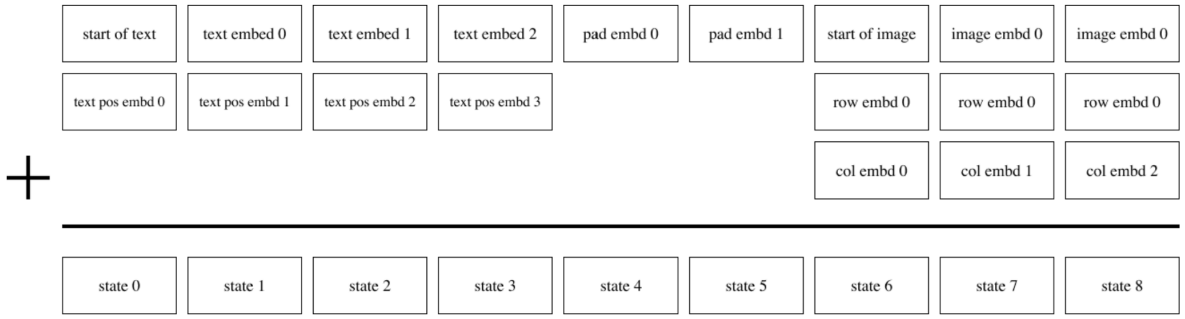


Figure 3: An example of the embedding scheme for the transformer with a maximum text token length of 6.

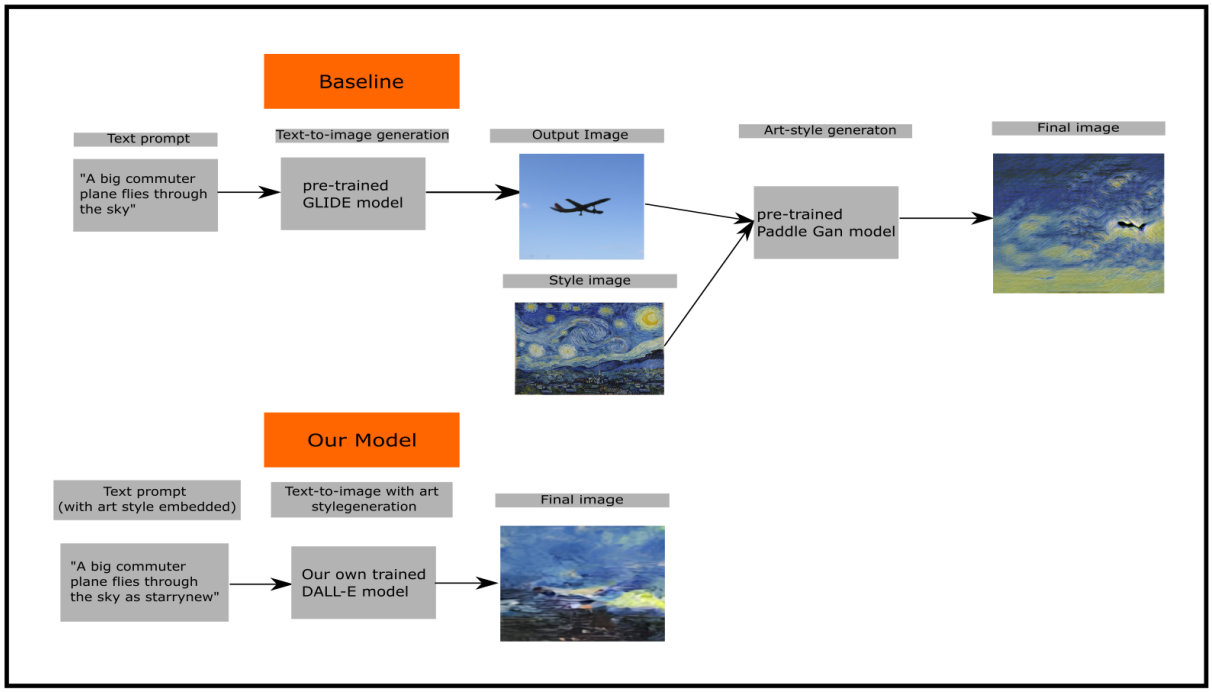


Figure 4: An overview of the input-output of the pre-trained baseline model and our model.

image. After doing so, we now had a dataset of 3000 images of labeled airplanes with art style applied to them. We split up this dataset into 2000 training images, 600 validation images and finally 400 test images which will be used to compare our different model architectures. Here is an example of one image from our dataset with its caption (see figure 5). All our code needed to manipulate and create our custom dataset is available on our Github repository.

Figure 5: "An AirFrance jet airplane flying in the sky with starrynew style"

4.2 Baseline

As mentioned previously, we are comparing two model architectures. One of which is pre-trained and the other trained by us. Therefore, the first model architecture which is a combination of the GLIDE text-to-image model and Laplacian Pyramid Network will be used as our baseline. We will run this model on the test set and then compare the result with our own model. Our goal will not be to outperform the pre-trained model as it has had

longer training time on a bigger dataset, but simply to see if the results of our model can be competitive with it.

4.3 Evaluation method

As an evaluation method, we decided to use an image similarity API (DeepAI) using a computer vision model already trained to compare two images. This API takes two images as input and returns a value, showing the similarity between two given images. The value is showing the distance between the input images. The best score to be obtained is '0' showing that two images are identical.

The evaluation will be done on our small test dataset of 400 images, each of our models will try and generate the 400 images from their captions. We will be able to compare the similarity scores to the actual images between both models.

4.4 Experimental details

For the model configuration of the text-to-image part, we went through different kinds of models before finding the GLIDE one. Firstly, we tried the Big-Sleep model (Wang, a), but it was too difficult to impose our own dataset on the model. We then tried the Deep-Daze model (Wang, b), but the time needed to generate one image was too high (about 10 minutes). We finally found a model which was fast enough to create a small dataset of our own images, that we could use for the next part of the project.

On the Art Style part, the pre-trained models from (Lin et al., 2021b) have been used all along, because it did not show any malfunction and was doing exactly what we expected it to do and produce. The first part was to get to know the models to be able afterhand to pipe the output of the text-to-image model to the input of the art style model. The pre-trained model allows us to select between four different styles, to simplify this step, only the 'stars' style has been used as all styles will give a consistent output. The model, to generate the output, needs to be set with the style selected, be given the style image and the input image (See input figure 6 and the associated result figure 7).

The second part consisted of putting the text-to-image and the art style together to be able to generate an image from a caption and directly applying a style to it (See the generate image given as input to the art style (given "an oil painting of a corgi" as caption), figure 8 and the associated result figure 9).

Figure 6: Test 1 Art Style Input

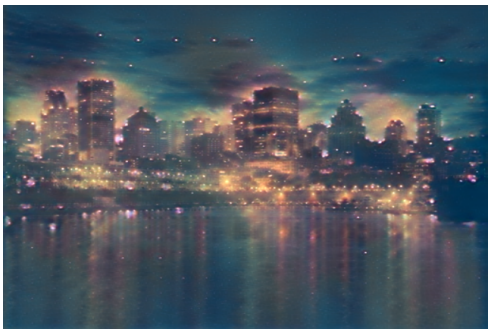


Figure 7: Test 1 Art Style Output

Figure 8: Test 2 Art Style Input

The most challenging part of our project was to find the best way to train our model, DALL-E from (Ramesh et al., 2021) with the limited resources we had. For reference, their VAE was trained on 64 16 GB NVIDIA V100 GPUs, with a per-GPU batch size of 8, resulting in a total batch size of 512. It is trained for a total of 3,000,000 updates.While their transformer was trained on 1024, 16 GB NVIDIA V100 GPUs and a total batch size of 1024, for a total of 430,000 updates. We did not have access to such hardware and were limited by

Figure 9: Test 2 Art Style Output

the free training account Google Colab[3]. This gave us access to different resources every time, usual Nvidia Tesla T4 or K80. The limited memory on a single GPU meant that we were limited in our batch size, since images take up a lot of space. We were also limited on the depth of the model we trained since the more layers we had the more memory was needed on the GPU. Lastly, we were limited by the training time since Colab would only let us train for 3-6 hours at time.

One solution to this problem that we found was that it was unnecessary for us to train our own VAE. The only task of this model was to compress an image so it could be concatenated with the text tokens, so we could use a pre-trained VAE. For this we used the authors pre-trained OpenAI VAE.

To train our model, the authors included the instructions and code within their official Github repository (DALL-E). We used much of their code and modified what was needed to train correctly on our dataset, the Colab notebook that was used by us can be found in our Github repository.

All these constrained meant that it would be impossible for us to fully train a model and hope for ideal results. We also did not know the ideal number of layers and other hyperparameters that would give us the best results for our task. In our case, the batch size and the number of layers were linked together since they both used up the memory of the GPU. We started by training a small model of only 2 layers, with such small model we were able to have a batch size of 6. After training this model for 24 hours (around 50 epochs), we increased the number of layers to 6 and had to reduce the batch size to 4. We also trained this model for 24 hours (around 50 epochs), and then increased to the maximum amount of layers we could : 12, this was done with a batch size of 2. Training this model for 24 hours was only around 20 epochs. In the paper from (Ramesh et al., 2021), the authors mention that the ideal number of layers for them was 64, however for us this was not possible.

Our thought with this type of training, is to see if we can find a link between the number of layers and the accuracy of the text generation. There is a flaw that we decided to train a certain number of hours, so our deeper layer has seen less of the data since the training was slower. However, after testing all three different depths with our baseline model, it should be enough as a sort of proof of concept. That we can in fact train a text-to-image model on a specific dataset with art style for that model to be able to do both task simultaneously.

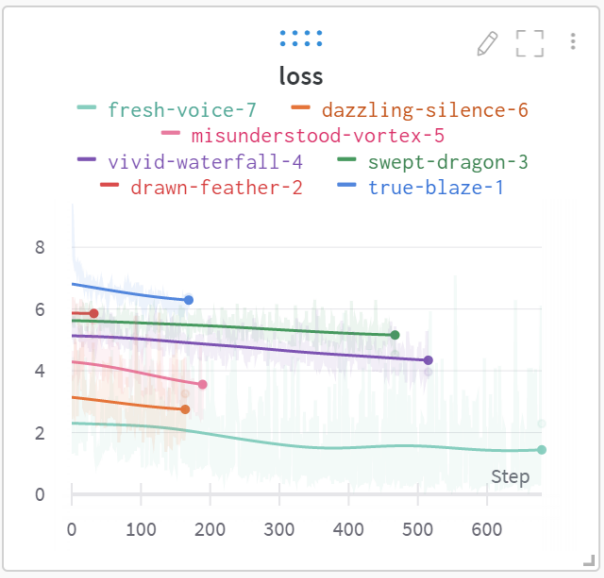


Figure 10: The loss graph of the model of depth 6 over 24 hours. Each different color represents a different training session. We can see the loss steadily decreasing as training increases.

To keep track of our training we used the website https://wandb.ai/ since it connected smoothly with Google Colab and helped us see how our training was going. It also let us resume training from where we were when Colab interrupts

[3] https://colab.research.google.com/

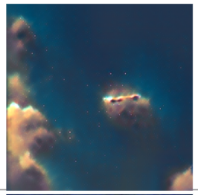
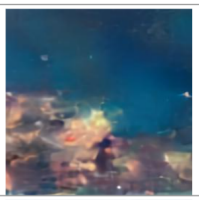
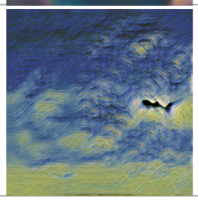
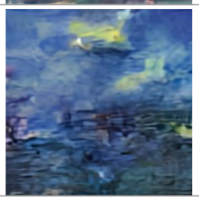
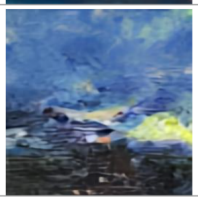
	Baseline	2 layers (50 epochs)	6 layers (50 epochs)	12 layers (20 epochs)
"A really big airplane that is flying in the sky as stars"				
"a big commuter plane flies through the sky as starrynew"				

Figure 11: Comparison of two captions and the images generated by our baseline vs our 3 different depth models.

our training. The most useful feature is keeping track of our loss, this could help us immediately see if our model was learning correctly and already gave us a good idea of how the loss of each model compared. For example, when training our 2 layer model, the loss seemed to hit its lowest point at around 2.5 and would not go any lower. However, once we trained the 6 layer model (see figure 10) it was able to go much lower to 1.5, and would have kept getting lower if longer training was possible.

4.5 Results and analysis

Before testing our models using the DeepAI similarity API, we can visually compare example of images with our baseline to see if some noticeable patterns can be found (see figure 11). At first glance, we can see that the art style is very well translated from text to image, however the description of the image itself is much harder to see. As we increase the depth of the model, the precision of the art style increases. For the stars style, the pink "clouds" are less random and form a shape resembling a plane. For the starrynew style, there is a better definition of the ground (darker) and the sky (lighter) while also having what looks like a white-ish plane for the 12 layer model. These images are not clear or ideal, but they show that with the right training resources a great text-to-image with art style model can be achieved.

In the figures 12 & 13, different things are noticeable, firstly the average of each series are fairly close (31.81 for the model with 12 layers against 32.46 for the model with 2 layers) but compared to our baseline model that achieved a distance of 18.6, the results are clear that the generated models with 2 layers or even with 12 layers are still not outperforming the baseline. Secondly, we can see that with the model with 12 layers, the distribution varies a lot less to the extremes. Something to be considered as well is that the minimum value obtained with the model containing 2 layers is higher compared to the one with 12 layers (i.e 24 for the 2 layers model against 20 for the one with 12 layers).

In overall, the fact of adding layers to the model do enhance the results obtained in the test phase when comparing images produced by the model to the ones from the baseline but the difference is still slight. Thus, we can clearly see that even if the difference between 2 layers and 12 layers on the distances side is slight, on an observation point (See 11, the difference is more clear, by the way that the Art Style is more smoothly applied in general.

5 Conclusion

In conclusion, we saw that the art style transfer is what makes the learning very difficult and time-consuming. One of the main obstacles is that the more the model has layers the better it becomes, but it requires a lot of resources. However, we are satisfied with the result because our model understands very well the art-style asked in the prompt, which was our main goal.

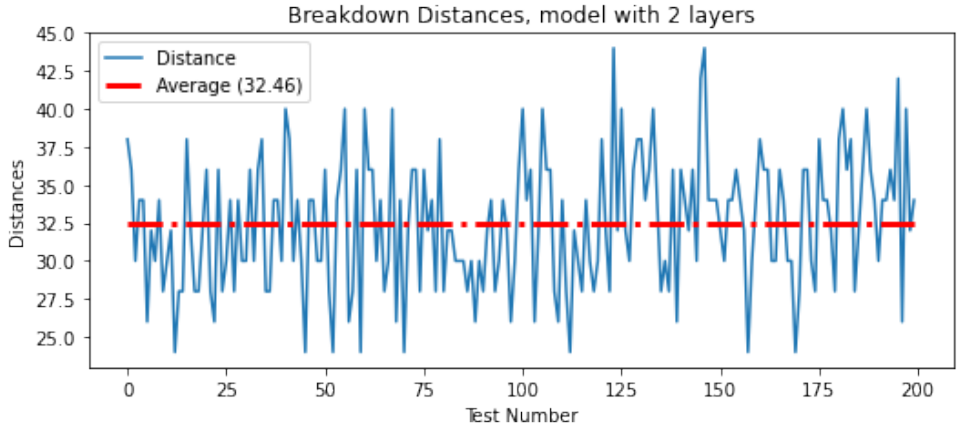


Figure 12: Breakdown of tests distances using a model with 2 layers

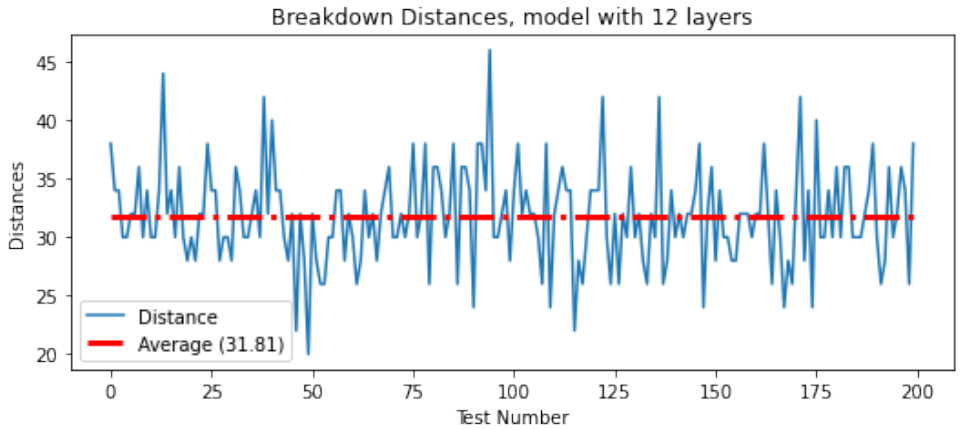


Figure 13: Breakdown of tests distances using a model with 12 layers

References

DALL-E. Dall-e. https://github.com/lucidrains/DALLE-pytorch.

DeepAI. Image similarity. https://deepai.org/machine-learning-model/image-similarity.

Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. 2015. Deep generative image models using a laplacian pyramid of adversarial networks.

GLIDE. Glide. https://github.com/openai/glide-text2im.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation.

Diederik P Kingma and Max Welling. 2013. Auto-Encoding variational bayes.

Jing Yu Koh, Jason Baldridge, Honglak Lee, and Yinfei Yang. 2020. Text-to-image generation grounded by fine-grained user attention.

Tianwei Lin, Zhuoqi Ma, Fu Li, Dongliang He, Xin Li, Errui Ding, Nannan Wang, Jie Li, and Xinbo Gao. 2021a. Drafting and revision: Laplacian pyramid network for fast high-quality artistic style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5141–5150.

Tianwei Lin, Zhuoqi Ma, Fu Li, Dongliang He, Xin Li, Errui Ding, Nannan Wang, Jie Li, and Xinbo Gao. 2021b. Drafting and revision: Laplacian pyramid network for fast high-quality artistic style transfer.

Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2015. Generating images from captions with attention.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*.

A Oord, O Vinyals, and K Kavukcuoglu. 2017. *Neural discrete representation learning*.

PaddleGan. a. Paddlegan/lap_style.md. https://github.com/PaddlePaddle/PaddleGAN/blob/develop/docs/zh_CN/tutorials/lap_style.md.

PaddleGan. b. paddlepaddle. https://www.paddlepaddle.org.cn/.

Project. Project$_github$..

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with VQ-VAE-2.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models.

Phil Wang. a. Big sleep. https://github.com/lucidrains/big-sleep.

Phil Wang. b. Deep daze. https://github.com/lucidrains/deep-daze.