# CSCI 1430 Final Project Report: Text-to-Image Synthesis By Separating "Verbal" From "Nonverbal" Information Using Residual Auto-Encoder

*Homemade DALL·E*: Koyena Pal, Luke West
Brown University

## Abstract

*Text-to-image synthesis is the idea of expressing a scene in text format and having a program generate an image based on the text description. Many text-to-image generation models utilize a huge dataset, a lot of computation power and time to train and generate good images. We illustrate an approach that uses significantly less resource and is able to generate decent images from our custom Shapes data.*

## 1. Introduction

Our project idea stems from an initial curiosity of understanding how to generate an image based on a natural language text caption given to the program. For instance, if we say, "there is a room with a window above the sofa," is it possible to generate an image of a sofa with the window on top of it and the surrounding image having a wall? This led us to a research community that uses machine learning approaches to text-to-image generation. One of the current most talked about models within this domain is Ramesh et al's model, DALL·E 2 [10], which is a new AI framework that can create realistic pictures from a description in natural language. It is a two-stage model: the first stage generates a CLIP [9] embedding given a text caption and the second stage decodes the outputted image embedding and generates the resulting image. To train its encoder-decoder arrangement, it takes about 900M images (around 650M for training its encoder and 250M for training its decoder). Its decoder architecture uses 3.5 Billion parameters. The predecessor of this model, Ramesh et. al's model, DALL·E [11], uses 10 Billion parameters in its training architecture.

Since these models have huge parameter and dataset sizes, we looked into other models that are more cost-saving. One such model is Zhou et al's framework, LAFITE [16]. It is a zero-shot text-to-image generation that has about 1% model size and training data size relative to the large DALL·E model. Upon comparing the model, we noticed a weakness in LAFITE's generated outputs. As shown in Figure 1, LAFITE's output for a triangle object looks more like a circle unlike DALL·E's output, which is clearly a set of triangular items. However, the former's output still had the color and object accurate. Hence, we decided to focus on training a text-to-image generation for shapes by creating our own shapes dataset and designing our architecture that has fewer parameters to train so that it can be feasible to run in our Google Cloud accounts. We believe that if our model does not test well on shapes, then it would not be able to produce a decent output for more complicated scenes. Hence, in this project, we worked on creating a model that can generate images for shape-related text captions. To summarize, our contributions are as follows:

- Created a Shapes Dataset ranging from squares to stars with different fill colors, positions, and background colors (size: about 2500).
- Designed and implemented a text-to-image shape generator model that uses 10M parameters, which is 99.9 % smaller than the large DALL·E model.

## 2. Related Work

With respect to text-to-image generation, there have been previous works on training GANs [3] on public image captioning datasets to produce text-conditional images [12, 13, 14, 15, 17]. Another approach is by adapting VQ-VAE [7] to train autoregressive transformers on text token sequences and then on image tokens [11, 2, 1]. Other works have applied diffusion models, either training in the continuous form [6] or discrete form [4] with text encoders to handle text captions.

Related to this task is the creation of newer datasets that aims to aid in further research of such topics. Generally, MS-COCO [5] is a popular choice of data. Point-Tuset et al. recently created a Localized Narratives dataset [8]. It has image annotations with verbal descriptions and mouse tracking. This allows for controlled image captioning which can be helpful to create text-to-video generation.

Figure 1. Generated Images for text caption: “a triangle black clock. a black clock in the shape of a triangle” *Left:* DALL·E *Right:* LAFITE

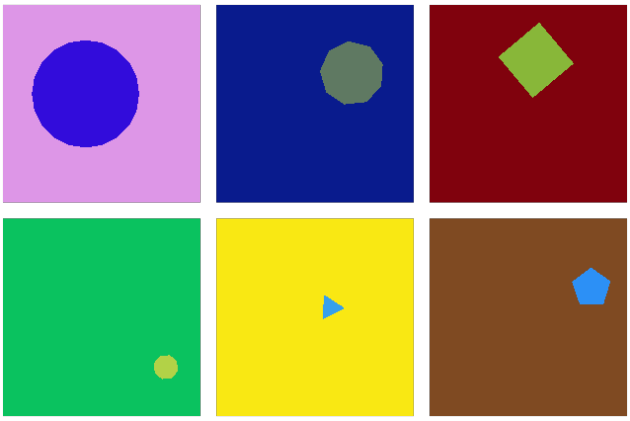

Figure 2. Subset of the Custom Shapes Dataset

## 3. Method

In this section, we first go over how we created our shapes dataset and move on to describing our model architecture.

### 3.1. Shapes Dataset Creation

To generate different shapes, we used different fill colors, background colors, size and orientations of the shapes in question. The shapes considered ranges from squares to stars. There are about 2500 of them and they have been generated with the help of members of the following GitHub Repository: 2DGeometricShapesGenerator. Figure 2 showcases the subset of Shapes that we generated.

### 3.2. Model Architecture

Many images can be captioned the same, and the same caption describes many images. Directly training a model pipeline to create images from text embeddings therefore seemed less feasible - we would want many captions for each image, and many images for each caption, which is not easy to find in real-world datasets. Additionally, such a model would not be inherently stochastic - the same caption would always lead to the same output image. In other words, the true underlying relation from captions to images is not a function, and so it seemed misguided to try to train a function on captions to imitate that relation. Instead, we realized we could *force* this relation to look more like a function by additionally parameterizing it with an abstract characterization of the image that captures everything about it that a verbal caption ignores. So, this characterization would encode everything about the image that a human would not mention in a caption. Together, the characterization, or “nonverbal” information, referred to as a latent image, along with the “verbal” sentence embedding, should map to an approximately unique output image. The latent image is created by a series of image convolutions that produce a 1024-dimensional vector, only a fraction of the size of the input image. This is concatenated with a text embedding to produce a vector that (hopefully!) fully characterizes the input image Figure 3 illustrates what our full model looks like. In the bottleneck of this architecture, just before the text embedding is added, the model is ideally too narrow to learn both “verbal” and “nonverbal” content. Hence, the auto-encoder should learn to rely on the supplied sentence embedding to recreate the image.

Once training is complete, the leftmost column (“encoder”) of the architecture shown in 3 is thrown away. To produce novel images, we pass text to the text embedding model, and input a randomized 1024-element latent image vector. The hope is that any possible latent image corresponds to some abstract characterization of an image described by the given caption. This hope is somewhat inspired by linear algebra — if the 1024 dimensions of the latent image vector are *just enough* to capture every possible abstract image layout associated with an arbitrary caption, then the components of the latent image should be akin to a vector space basis, in which *every* possible linear combination of those components produces a unique valid element of that space.

### 3.3. Training Regime

On our Shapes dataset, we initially used a slightly downsized version of the model shown in 3, where the latent image vector was only of size 54. This was because any image in the shapes dataset could be characterized with far, far less information than a real-world photo. In fact the number 54 was based on an estimation of the number of bits represented by any Shapes image — $2^8 = 256$ bits for each channel of the background and foreground colors, $\approx 2^7$ bits for the $x$ and $y$ coordinates of the shape, $\approx 2^7$ bits for the size of the shape, and perhaps a few extra bits for its rotation, for a total of around 72 bits. We then chose to round down to 54 elements because each element is not actually a bit, but only approximates one with certain activation functions and normalization layers, e.g. sigmoid activations.

We also used a vastly simplified text embedding model, using only the words “Circle, Star, Triangle, Square, Pentagon, Hexagon, Heptagon, Octagon, Nonagon”, and map-

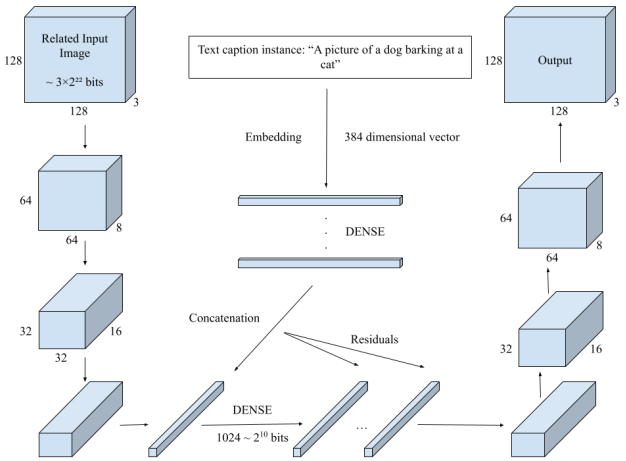


Figure 3. Architecture Overview

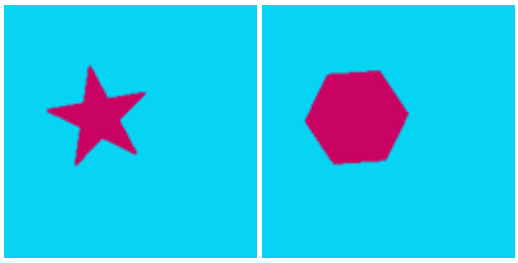

Figure 4. Augmented Training Pair

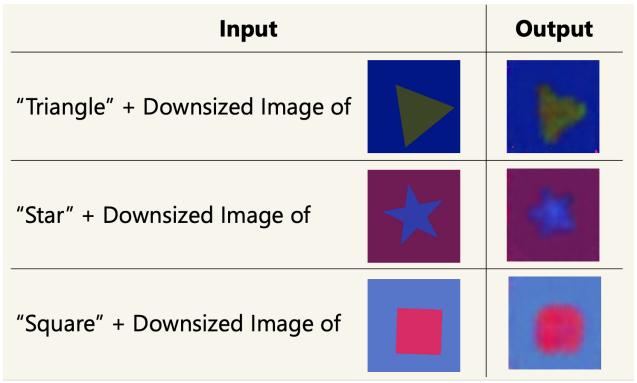


Figure 5. Generated Results On Matching Latent Image/Caption Pairs

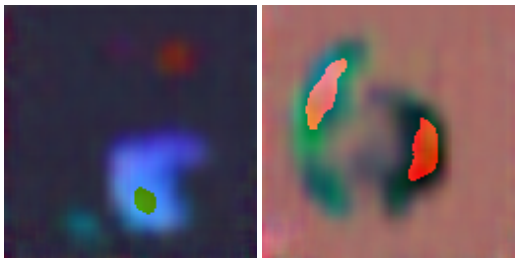

Figure 6. Generated Results on Randomized Latent Images (caption inputs from left: "Square", "Hexagon")

ping these to a 4-element vector. Each image was passed to the model both as an input and a desired output, and loss was measured with Mean Squared Error. We considered using a Generative Adversarial Network model, with an adversarial discriminator serving as the loss function, but decided against it given our time constraints. We also realized that the latent image should be able to encode perturbations like rotations, zooms, skews, etc., and thus an adversarial network might not theoretically be necessary even when training on a real-world photo dataset.

After our first few rounds of results, we augmented our training regime by generating shapes images in pairs, with the figure depicted being identical in every way (size, color, rotation, etc.) saved for its actual shape. An example of such a training pair is shown in 4.

All training was performed on a GCP VM instance using a single NVIDIA Tesla T4 GPU.

## 4. Results

Figure 5 illustrates the generated images from our model that was trained using our Shapes dataset. While testing our model, our input image was downsized to 54 pixels. We were still able to replicate what the actual image looked like, so the model at least performed decently as an autoencoder. However, when tasked with generating *new* images, by randomizing the latent image, it definitely failed. For the most part, we just got formless blobs.

Training this model took about 1.5 hours for 500 epochs in total. We attempted to expand the use of this model further by using the MS-COCO dataset. We replaced our text embedding framework with a (pre-trained BERT model) to take advantage of transfer learning. Due to time constraints, we were not able to get this model to produce a decent output (500 epochs would have taken 2-4 days). Nevertheless, this experience led us to think of a way to train our models even better. After a few epochs on the COCO dataset, we noticed that the model was consistently ignoring all input from the embedding — on a given latent image, a meaningful embedding produced identical input to a randomized one.

On further reflection, we realized that on both the shapes dataset and the COCO dataset, the model's path towards minimizing MSE only led it to functioning as a poor autoencoder, and it was not learning to take information from the caption embedding. So, it was not sufficient to just restrict the latent image space and hope the model would take information from the embedding to improve its output — to see good results, we would need to *force* it to *forget* certain information from the latent image, only to regain it from the caption embedding.

This line of thought led to a new training regime, previously referred to as the augmented training regime as in

Figure 7. Generated Results on Latent Image of Star with Caption of “Hexagon”

Figure 4. In this new strategy, we would give the model a shape image matching its provided caption on only half the training pairs. On all other pairs, we would *lie* to the model about the shape actually depicted in its input image, yet expect the model to give us the shape described in the caption. For instance, as in Figure 4, we would supply it with an image of a star, but caption it as a hexagon, and calculate the mean squared error of its output with respect to the image of the hexagon. These results were much more encouraging, as shown in Figure 7. Not only was the model able to learn to replicate the shape described, irrespective of the actual shape input, but the images generated from random latent image data were more impressive, as shown in Figure 8. It also performed quite well on latent images taken from actual shape images that it hadn’t seen before, shown in Figure 9. We think it may have performed even better on randomized latent images if we had restricted its latent image space — the augmented training was performed on the full 1024-dimensional model, and so the latent image space was surely too big, but we ran out of time to re-train it.

We were also able to form images from randomized text embeddings of the given shape, which often resembled interesting intermediates of the other training shapes. One odd thing we noticed was the consistent underperforming on “Triangle” inputs, and we really don’t know how to explain this. Given a randomized latent image and caption input of “Triangle”, only occasionally did the the output images resemble any sort of shape at all — the example given in Figure 8 was the best result we found.

### 4.1. Future Work

Given the success of the augmented training regime, we believe we could apply it to real-world photo data given a slightly altered task. If the caption embedding instead was only a list of nouns from a given set, describing the objects present in the image, we might prepare a dataset of paired images where one object in each pair is different in the two images, but expect the resulting image to always match the provided caption. This might be able to produce novel images of the objects from randomized latent image data, if the training dataset had enough coverage.

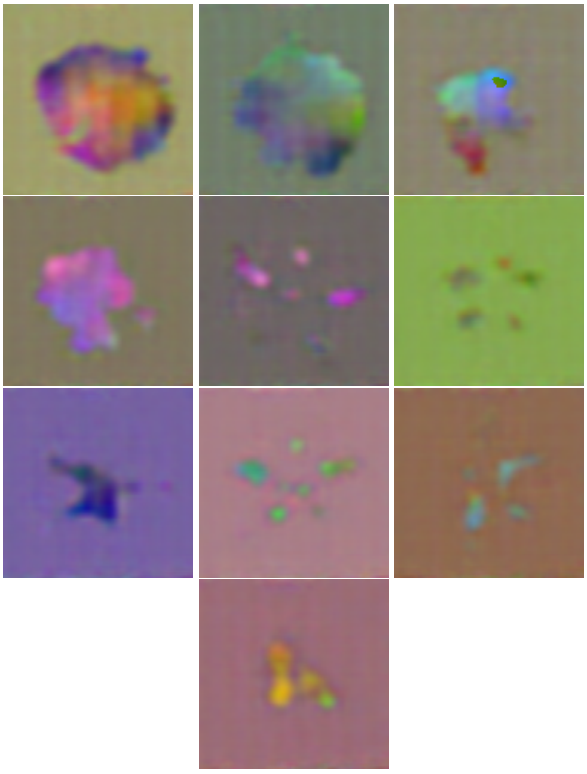

Figure 8. Generated Results on Randomized Latent Images (captions from top left: “Heptagon”, “Heptagon”, “Square”, “Square”, “Star”, “Star”, “Star”, “Star”, “Star”, “Triangle”)

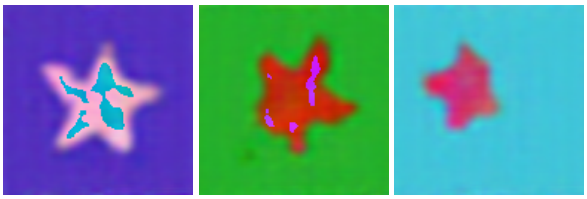

Figure 9. Generated Results using Latent Images from Unseen Shape Data, given caption “Star”

### 4.2. Societal Discussion

One of the potential ethical concerns talked about in the swap critique is that we could potentially destroy the jobs of professional artists since the model could create the wanted art instead. While this can be true, the current state of the model is far from destroying jobs. On the other hand, this model can be used as a complementary equipment for artists to use to create simple shapes or objects to save time. Since this project is inspired from DALL·E, another ethical concern that was discussed was a privacy issue. What happens if the image of a face that is generated by DALL·E actually exists in real life? How can we prevent this from happening? We were not sure about how it can be prevented entirely. However, we could have an approved database of faces of individuals who consented to their faces being use in third

parties. If such a face does not appear in the database, the generated image can be scrapped before the end-user can see the image. Apart from faces, people are also concerned with generating fake news using this generator mechanism. To prevent this from happening, this generator could be limited to certain trusted parties where we have some level of guarantee that they are not using this generator to create fake data. The news come from people who are not from the trusted parties, we would err to believing that they are spreading fake news. Lastly, in the swap critique, we were asked if this model was trained on any dataset apart from COCO. We did do this by creating our own Shapes dataset. Since shapes don't hold any discriminatory value, other concerns of racial bias (which is present in DALL·E 2) is not applicable here.

## 5. Conclusion

To create a cost-saving text-to-image generation model, we design a residual auto-encoder and train our model on our 2500-image dataset, which consists of various shapes in different colors, backgrounds, and position. By creating a decent text-to-image shape generator model, we are able to see that it is possible to have a good model that is 99.9% times smaller than the large DALL·E model. Further, it is possible on this dataset to have the model learn shape information only from the caption embedding, suggesting similarly separated learning might be possible on real-world photo data. The impact of this project is to further the direction of generating plausible images with as few data examples and model size as possible.

## References


[1] Armen Aghajanyan, Bernie Huang, Candace Ross, Vladimir Karpukhin, Hu Xu, Naman Goyal, Dmytro Okhonko, Mandar Joshi, Gargi Ghosh, Mike Lewis, and Luke Zettlemoyer. Cm3: A causal masked multimodal model of the internet, 2022. 1

[2] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. Cogview: Mastering text-to-image generation via transformers, 2021. 1

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1

[4] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis, 2022. 1

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. 1

[6] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2022. 1

[7] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2017. 1

[8] Jordi Pont-Tuset, Jasper Uijlings, Soravit Changpinyo, Radu Soricut, and Vittorio Ferrari. Connecting vision and language with localized narratives. In *ECCV*, 2020. 1

[9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1

[10] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. 1

[11] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021. 1

[12] Ming Tao, Hao Tang, Fei Wu, Xiao-Yuan Jing, Bing-Kun Bao, and Changsheng Xu. Df-gan: A simple and effective baseline for text-to-image synthesis, 2020. 1

[13] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks, 2017. 1

[14] Hui Ye, Xiulong Yang, Martin Takac, Rajshekhar Sunderraman, and Shihao Ji. Improving text-to-image synthesis using contrastive learning, 2021. 1

[15] Han Zhang, Jing Yu Koh, Jason Baldridge, Honglak Lee, and Yinfei Yang. Cross-modal contrastive learning for text-to-image generation, 2021. 1

[16] Yufan Zhou, Ruiyi Zhang, Changyou Chen, Chunyuan Li, Chris Tensmeyer, Tong Yu, Jiuxiang Gu, Jinhui Xu, and Tong Sun. Lafite: Towards language-free training for text-to-image generation, 2021. 1

[17] Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis, 2019. 1

# Appendix

## Team contributions

We both contributed equally in the project. While we primarily did the following individually, for all the "individual" tasks, we discussed and brainstormed our issues together.

**Person 1** This person primarily worked on conducting literature reviews, writing progress and final reports, as well as comparing models present within the literature reviews (for instance, see Figure 1)

**Person 2 (capstone student)** This person worked on designing, implementing, and training the model displayed earlier, porting it to GCP, and adapting it to the COCO dataset (see Figure 3). They also wrote the pre-processing code to handle incremental generation of COCO data via a Python iterator in a manner compliant with the Tensorflow/Keras API, and in such a way that only a small portion of the images are loaded into memory at any given time. For their capstone, this person decided to also create the dataset and code for the augmented training regime, and to analyze the performance of the model on this regime compared to the initial regime.