Text to Image Generation

Daniel Vadranapu
Department of Robotics
University of Buffalo
dvadrana@buffalo.edu

Anurag Hruday Pirangi
Department of Robotics
University at Buffalo
anuraghr@buffalo.edu

Abstract

In this project, our objective is to develop an advanced AI system capable of creating vivid and expressive images based on textual descriptions. The system will take text inputs and generate corresponding images that faithfully capture the essence and details described in the text. Our approach involves leveraging the Latent Diffusion Model, where we aim to optimize model parameters and fine-tune its performance.Before diving into the Latent Diffusion Model, we're ensuring a solid understanding of Generative Adversarial Networks (GANs) through a step-by-step process. We began with a basic GAN implementation using the MNIST dataset, successfully generating digit images. Subsequently, we progressed to more complex datasets, such as COCO, to generate images representing various classes. Additionally, we explored the synergy of language and vision by training a BERT encoder and a CNN for generating images from textual inputs. This integration allowed us to bridge the semantic gap between textual descriptions and visual representations. As we move forward, we refined our model by fine-tuning the Latent Diffusion Model and rigorously testing its performance. During our testing phase of the Latent Diffusion Model, we encountered several issues that shed light on areas for potential improvement. These insights are invaluable for refining the model's performance and addressing its limitations which will be discussed further in the report.

1 Introduction

In recent years, there has been a surge of interest in developing artificial intelligence (AI) systems capable of generating images from textual descriptions. This endeavor has been fueled by seminal works such as the Generative Adversarial Networks (GANs) introduced by Goodfellow et al. [4]. GANs have revolutionized the field of generative modeling by providing a framework for training generative models via an adversarial process, wherein a generator network learns to produce realistic samples while a discriminator network learns to distinguish between real and generated samples.Building upon the foundation laid by GANs, researchers have made significant strides in generating images from complex datasets such as the Common Objects in Context (COCO) dataset [7]. This dataset contains a diverse range of images depicting everyday scenes and objects, making it an ideal testbed for evaluating the capabilities of image generation models.Furthermore, the synergy between language and vision has been explored through the integration of advanced language models like BERT [3] and convolutional neural networks (CNNs) [6]. By training these models to understand textual inputs and generate images, researchers have sought to bridge the semantic gap between textual descriptions and visual representations. We begin by elucidating the foundational concepts of GANs and progress to more complex datasets like COCO. We then explore the integration of language and vision through BERT and CNNs. Finally,

we leverage the Latent Diffusion Model [5] for image generation and iteratively refine our approach based on testing and experimentation.

2 Work Done

2.1 GAN on MNIST Dataset

Developed a basic generator Convolutional model which uses transposed convolution layer which takes random noise input and outputs a number image. Later, the generated input image is passed into the Discriminator model where we use a BCE loss for predicting the image is real or fake. After, several hours of training the model generated images which are close to the real data and can't be classified by the discriminator. The generated images closely resemble authentic data, displaying intricate digit patterns and details. The MNIST dataset was fairly simple and the complexity in the data is less and the model was able to generate easily. Sample architecture can be found below [4].

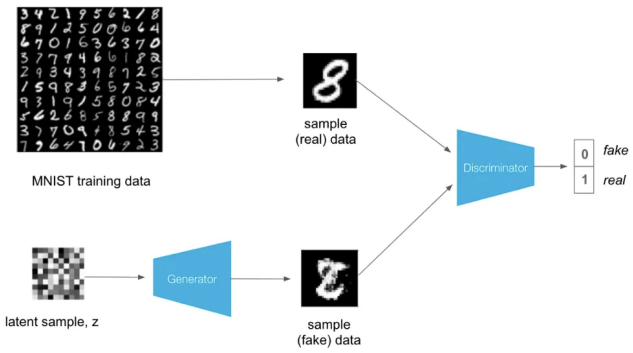

Figure 1: Block Diagram Illustrating the GAN Architecture

2.2 cGAN on AFHQ Dataset

Traditional GANs generate data without any control over the required output. While cGAN allows for controlling the generation of the data by conditioning on information either the information can be a class labels, attributes or contextual data. This helps the model generate data which is more diverse and useful to the conditioning information [10]. The Generator takes random noise and the conditioning input and generates the data, while discriminator does the same as the above basic GAN itself.

$$\min_G \max_D E_{x\sim p_{data}(x)}[\log D(x|y)] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)|y))] \quad (1)$$

where x is the real data, y is the conditioning variable, z is the random noise, $G(z|y)$ is the generated data given the condition y, and $D(x|y)$ is the probability that x is real given the condition y.

Discriminator tries to maximize the probabilty of correctly classifying the real and generated data as fake, given the condtion y. While Genearator tries to minimize the probability that the discriminator correctly classified generated data. The Model was trained on COCO dataset for 100 epochs and the progress of generation of data for the cat class can be seen below.

Figure 2: Progression of Training

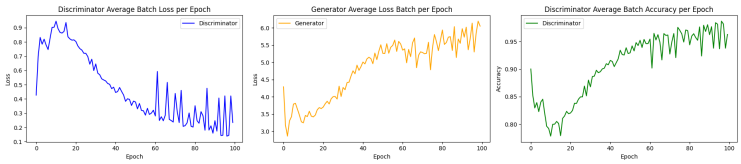

Figure 3: Results

2.3 cGAN with BERT Text Encoder for Text-to-Image Synthesis

This project involves developing a Conditional Generative Adversarial Network (GAN) to generate images from textual descriptions. The GAN architecture includes a BERT-based Text Encoder, a Generator, and a Discriminator.

2.3.1 Data Handling and Transformations

The COCO dataset is utilized, specifically the train2017 subset, which contains approximately 118,000 images along with their corresponding annotations. The `COCODataset` [7] class is used to load and preprocess image-caption pairs from the dataset's annotations. Images are resized to 256x256 pixels, converted to tensors, and normalized with a mean and standard deviation of 0.5 for each channel. PyTorch's `DataLoader` is used to handle batching and shuffling, with a batch size of 32, to ensure efficient data loading during training.

2.3.2 Model Architecture

The model architecture consists of the following components:

Text Encoder: Uses a pre-trained BERT model to convert text into feature vectors. BERT is chosen for its ability to capture rich semantic information from text, which is crucial for understanding the context and details described in the captions. The Text Encoder has a total of 109,482,240 parameters.

Generator: Composed of a fully connected layer followed by three deconvolutional (transposed convolutional) layers. The architecture is designed to map the 768-dimensional text embeddings to image space: the fully connected layer maps the text features to a 256-dimensional feature map of size 16x16. The subsequent deconvolutional layers progressively upsample this feature map to generate a 256x256 image. These layers include converting 256 channels to 128 channels, followed by ReLU activation, converting 128 channels to 64 channels, followed by ReLU activation, and a final layer converting 64 channels to 3 channels (RGB), followed by Tanh activation to ensure the pixel values are in the range [-1, 1]. This design allows the generator to create high-resolution images that maintain the semantic details provided by the text descriptions. The Generator has a total of 51,055,811 parameters.

Discriminator: Consists of five convolutional layers with LeakyReLU activations and batch normalization. The discriminator is tasked with distinguishing between real and generated images: each layer progressively reduces the spatial dimensions while increasing the depth, extracting hierarchical features from the images. The layers include an initial layer converting 3 channels to 64 channels, subsequent layers increasing the number of channels (64 to 128, 128 to 256, 256 to 512) while halving the spatial dimensions, and a final layer outputs a single value indicating the probability that the input image is real. LeakyReLU is used to avoid dead neurons, and batch normalization helps stabilize training by normalizing activations. The Discriminator has a total of 2,766,529 parameters.

2.3.3 Training

The adversarial loss for both the Generator and Discriminator is given by the Binary Cross-Entropy Loss:

$$\mathcal{L}_D = -E_{x \sim p_{data}}[\log D(x)] - E_{z \sim p_z}[\log(1 - D(G(z)))]$$

$$\mathcal{L}_G = -E_{z \sim p_z}[\log D(G(z))]$$

The Binary Cross-Entropy Loss is chosen because it is well-suited for binary classification tasks, where the discriminator's job is to classify images as real or fake. This loss function effectively measures the difference between the predicted probabilities and the actual binary labels (real or fake).

Adam optimizer is used for both the Generator and Discriminator with a learning rate of 0.0002 and betas (0.5, 0.999). Adam is selected for its adaptive learning rate capabilities, which helps in converging faster and handling the sparse gradients typically encountered in GAN training.

The training process involves alternating updates to the Generator and Discriminator, minimizing their respective losses. The Generator aims to create realistic images that can fool the Discriminator, while the Discriminator learns to distinguish between real and generated images. Training was conducted on the train2017 subset of the COCO dataset for 20 epochs, with a batch size of 32. Training losses are logged using TensorBoard, and model checkpoints are saved at each epoch. This approach ensures that the Generator and Discriminator improve iteratively, leading to the production of high-quality images that accurately reflect the input textual descriptions.

The total number of parameters in the models are as follows: Text Encoder: 109,482,240 parameters, Generator: 51,055,811 parameters, Discriminator: 2,766,529 parameters, Overall: 163,304,580 parameters.

2.3.4 Results

Figure 4 shows the training progress over 20 epochs, logging the Generator (G) and Discriminator (D) losses at various steps. The Generator's loss initially fluctuated but showed an increasing trend over time, while the Discriminator's loss remained very low. Despite the increasing Generator loss, the model demonstrated the ability to generate diverse images from textual descriptions, showcasing the potential of the Conditional GAN architecture. The increasing Generator loss suggests areas for further improvement and optimization. The following factors could contribute to the observed results: Mode Collapse: The Generator might be producing a limited variety of images, causing the Discriminator to easily distinguish between real and fake images. Imbalance in Training: The Discriminator could be learning faster than the Generator, making it difficult for the Generator to catch up and produce convincing images.

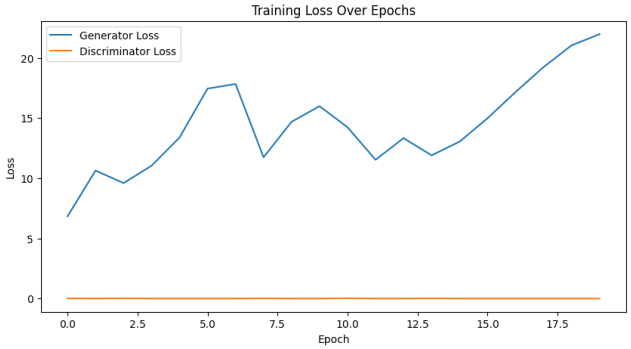

Figure 4: Training Loss over Epochs

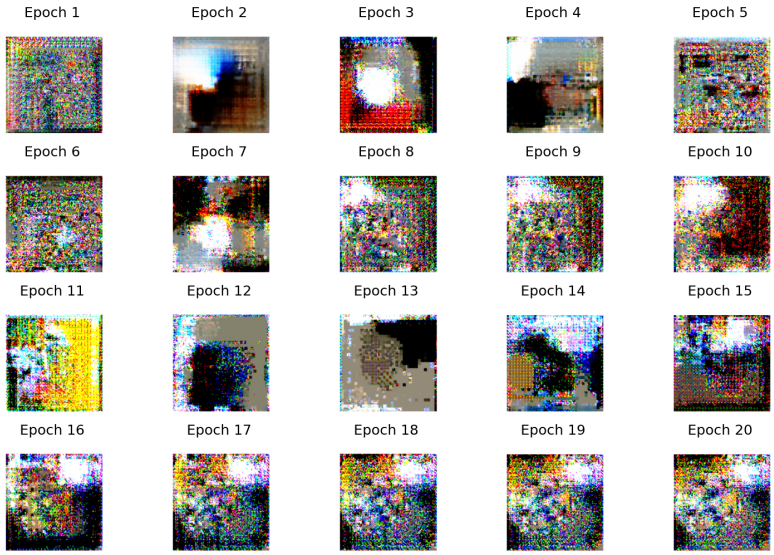

Figure 5: Progression of training epochs

Figure 5 illustrate the progression of generated outputs from the Conditional GAN model over 20 epochs. Initially, the images are very noisy and lack recognizable structure, as seen in Epoch 1. By Epoch 2, there is a slight reduction in noise, but clarity is still lacking. As training progresses, especially by Epochs 7 and 8, some structure begins to emerge, although the images remain abstract. From Epochs 9 to 12, the generated images show more distinct patterns and less noise, indicating the model is learning features, but they are still not coherent. This trend continues through Epoch 15, with some improvement in texture and pattern, but the outputs are far from realistic. In the later epochs, specifically from Epochs 16 to 20, there is a noticeable improvement in stability and pattern definition, yet the images still lack clarity and detail. The model's generator struggles to create realistic images from text prompts, possibly due to dataset complexity and training dynamics.

3 Latent Diffusion Tweaking and Testing

Implemented the existing Latent Diffusion model [13] which is good at generating high-resolution images. The LDM model works well, However, during testing, we encountered a notable issue: the model's performance significantly deteriorated when presented with input containing typographical errors. While the LDM excels in generating high-quality images under ideal conditions, it with

input containing errors. In such instances, the model tends to produce erratic, noisy images that lack coherence and fidelity to the intended content. This inconsistency underscores a crucial limitation of the current implementation. Despite its overall effectiveness, the model's robustness to input variations, such as typographical errors, remains inadequate. As a consequence, addressing this issue is imperative to enhance the model's reliability and versatility across diverse input scenarios. Moving forward, strategies to improve the model's resilience to input variability merit exploration. This could involve augmenting the training data with samples containing typographical variations, implementing more robust error-handling mechanisms, or enhancing the model architecture to better handle noisy inputs. By addressing these challenges, we can strive to elevate the LDM's performance and broaden its applicability in real-world scenarios.

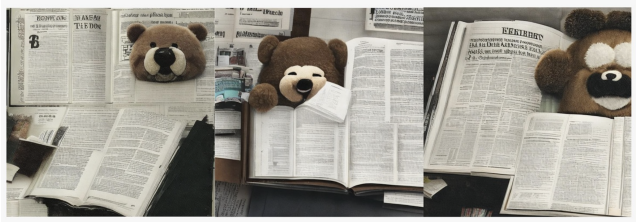

Figure 6: Prompt: A happy bear reading a newspaper

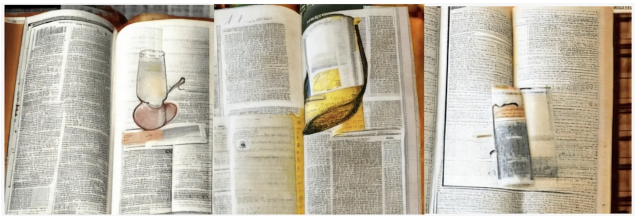

Figure 7: Prompt: A happy beer reading a newspaper

In the above Figure 6, The model has predicted a happy bear with newspaper accurately, while in the Figure 7, when the spelling was changed the model has just stitched the two texts with respective images rather than giving them separately. This is one of the area of improvement which can be done to improve the model performance on such edge cases.

4 Training Resources

Testing the Latent Diffusion Model (LDM) demands substantial computational resources due to its sophistication and size. The current state-of-the-art models, rich in parameters, place considerable demands on hardware. For LDM testing, a minimum configuration of 32GB of CPU RAM and a graphics card with at least 20GB of memory is necessary. Additionally, the pretrained model itself occupies approximately 11GB of storage. To effectively process the LDM model, a robust GPU is essential. At present, a minimum requirement of one A100 GPU is necessary to handle the computational workload efficiently. This GPU provides the necessary power to execute the complex computations required by the LDM, ensuring optimal performance during testing and evaluation.

5 Challenges

While working on the Conditional GAN model for text-to-image synthesis, we encountered several challenges that significantly impacted our progress. One of the biggest hurdles was achieving a balanced training process between the Generator and Discriminator. Often, the Discriminator would become too strong too quickly, easily distinguishing real images from generated ones. This led to a zero loss for the Discriminator and minimal improvement in the Generator's ability to produce realistic images. We had to experiment with various strategies, such as adjusting learning rates, trying different optimizers, and incorporating techniques like gradient penalty to ensure that both models improved together.

Interpreting and encoding textual descriptions into meaningful feature vectors was another complex challenge. Our text encoder, based on a pre-trained BERT model, needed to capture the nuances and details of the descriptions to provide useful input for the Generator. We spent considerable time fine-tuning the text encoding process and validating the quality of the generated feature vectors to ensure they were rich and informative enough to guide the image generation.

Handling the COCO dataset, with its large size and diversity, posed significant preprocessing and data handling challenges. Efficiently loading and preprocessing the data to ensure smooth training was crucial. This involved resizing images, normalizing them, and organizing data into batches. We had to carefully plan and optimize our approach to manage computational resources effectively and handle the large dataset without running into memory issues.

Tuning the hyperparameters for the model was a critical and time-consuming task. Finding the right learning rates, batch sizes, and architecture for both the Generator and Discriminator required extensive experimentation and validation to see what worked best.

Ensuring stability during GAN training was inherently challenging due to the adversarial nature of the models. The Generator aims to create images that can fool the Discriminator, while the Discriminator aims to correctly identify real versus generated images. This setup can lead to oscillations, mode collapse, or failure to converge. We implemented techniques like Wasserstein GAN (WGAN) and used specific loss functions to stabilize the training, which required careful integration and testing.

Lastly, training deep learning models, especially GANs, is computationally intensive. We had to ensure that our available GPU resources were utilized efficiently while avoiding bottlenecks. This often involved optimizing code for parallel processing, managing data loading efficiently, and ensuring that the training process could run for extended periods without interruption. These challenges required a combination of theoretical understanding, practical experimentation, and iterative refinement to develop a robust Conditional GAN model capable of generating images from textual descriptions. Overcoming these issues not only improved the model's performance but also provided valuable insights into the complexities of GAN training and text-to-image synthesis.

6 Fair Use of GANs

Generative Adversarial Networks (GANs) hold significant potential across various fields, from creating realistic images to generating synthetic data for research. However, their powerful capabilities come with substantial ethical considerations. One of the main concerns is the potential for misuse, such as generating deepfakes or misleading synthetic media, which can harm individuals and undermine trust. To counteract this, developers and researchers should implement safeguards like watermarking generated content and ensuring compliance with data privacy laws [12, 11, 9].

Additionally, addressing biases in training data is crucial to avoid reinforcing harmful stereotypes or discrimination through generated content. Ethical guidelines and regulations are necessary to prevent misuse, and ongoing discussions

with ethicists, lawmakers, and the public can help create robust standards for GAN use [12, 11]. By adhering to these principles, we can harness the benefits of GANs while mitigating associated risks, ensuring their development and deployment serve the greater good.

7 Future Work

As outlined in Section 3, enhancing the Latent Diffusion Model (LDM) involves refining its ability to comprehend textual context, thereby enabling more accurate results. This refinement includes addressing typographical errors, grammatical inconsistencies, and contextual nuances to improve the model's output precision.

Architectural adjustments are essential for better accommodating the complexities of textual input, enhancing the LDM's ability to interpret contextual cues. Such adjustments may include incorporating mechanisms for error detection and correction, integrating contextual understanding modules, or refining attention mechanisms to focus on relevant textual features. By fine-tuning the architecture in this manner, we can bolster the LDM's performance and ensure it produces more reliable and contextually coherent results, even under challenging input conditions. This approach represents a crucial step toward advancing the capabilities of LDMs and enhancing their practical utility across various applications [2, 8, 1].

References

[1] Wang J. Li X. Chang, Y. Advanced architectures for latent diffusion models in text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3456–3465, 2022.

[2] Qiusheng Zheng Chenyang Li, Long Zhang. Utilizing latent diffusion model to accelerate sampling speed and enhance text generation quality. *Electronics*, 13(6):1093, 2024.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2014.

[5] Jonathan Ho, Priya Jaini, and Pieter Abbeel. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, 2020.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

[8] Hoffmann H. Zhu X. Franke B. Markus, M. Improving latent diffusion models with enhanced textual context understanding. *Journal of Machine Learning Research*, 24(112):1–20, 2023.

[9] K. Martin. Tackling bias in ai: Ensuring fairness in machine learning models. *Ethical AI Journal*, 2021. `https://www.ethicalaijournal.com/tackling-bias-in-ai`.

[10] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[11] A. NG. Responsible ai: Guidelines for ethical ai development. *AI Ethics Review*, 2020. `https://www.aireview.com/responsible-ai-guidelines`.

[12] N. Patel. The ethical considerations of generative adversarial networks. *Journal of AI Ethics*, 2023. https://www.journalofaie.com/ethical-considerations-of-gans.

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*, 2022.