Understanding and Expanding the Conditioning Mechanisms in "High-Resolution Image Synthesis with Latent Diffusion Models"

Neil Chulani
Math 179, Harvey Mudd College
Attending Pomona College
njcf2022@mymail.pomona.edu

Abstract

"High-Resolution Image Synthesis with Latent Diffusion Models" introduces a new generative model for image synthesis called Latent Diffusion Models (LDMs). These models are built on a project called "Stable Diffusion", which aims to improve the flexibility of diffusion models as conditional image generators by combining their generative power with cross-attention mechanisms. The goal of this project is to replicate and understand the results of a state-of-the-art generative model that maps language/text inputs to comprehendible images. Specifically, I aim to understand the conditioning mechanisms that are used to begin and guide the image generation process. To demonstrate such an understanding, I will build on the current image generation and implement an audio to image model.

1 The Methods Behind High-Resolution Image Synthesis

In order to begin to understand the conditioning mechanisms used, we must get a high level understanding of the methods in High-Resolution Image Synthesis. This will allow us to take an informed dive into the intricate process of using different forms of input to begin the image generation process.

1.1 Perceptual Image Compression

The perceptual image compression model used is a specific type of autoencoder trained using a combination of a perceptual loss and a patch based adversarial objective. Perceptual loss is a measure of how well the reconstructed image matches the original in terms of visual similarity, such as texture, contrast, and color. This type of loss function is used rather than pixel-space losses such as L2 or L1 objectives since it better captures human perception of image quality. The patch-based adversarial objective is a method of training the autoencoder to generate more realistic images by using a discriminator network that provides feedback on the realism of the reconstructed images. This helps to avoid blurry reconstructions that might result from relying solely on pixel-space losses. The process as a whole ensures that the autoencoder produces compressed images that are visually similar to the original images, while also maintaining local realism and avoiding blurry artifacts.

1.1.1 Encoding in Perceptual Image Compression

An image can be thought of as a matrix in which each entry is a 3x1 vector, such that each entry stores RGB values. Thus, an image $x \in \mathbb{R}^{H \times W \times 3}$ in RGB space can be encoded by encoder $\mathcal{E}$, which transforms x into latent, compressed form $z = \mathcal{E}(x)$. We also define $\mathcal{D}$, the decoder which reconstructs the latent image form to a relatively accurate approximation original, which includes the details that are relevant to human perception:

$$
\begin{aligned}
z &= \mathcal{E}(x) \\
\mathcal{D}(z) &= \mathcal{D}(\mathcal{E}(x)) \\
\mathcal{D}(z) &= \tilde{\boldsymbol{x}}
\end{aligned}
$$

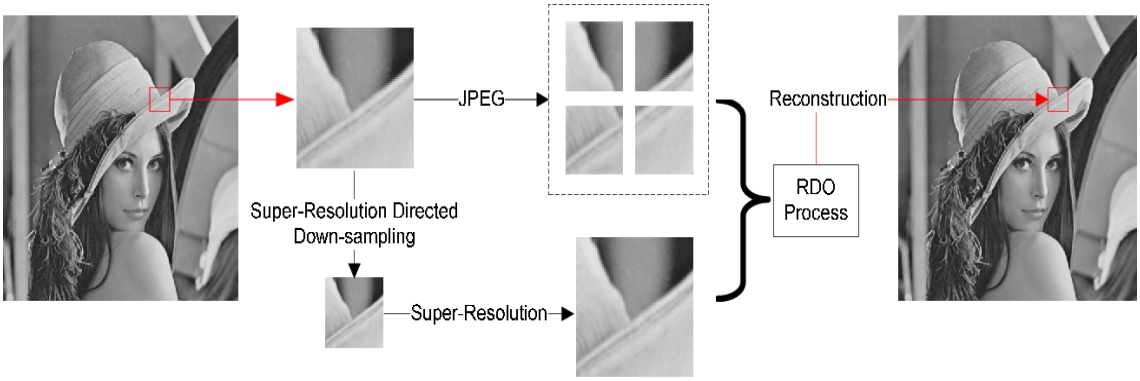


Figure 1: Visualization of Perceptual Image Compression (source)

1.2 Latent Diffusion Models

Latent diffusion models are an extension of diffusion models, which are a key to image generation. Diffusion models work by iteratively adding noise to an initial image and then removing the noise in a controlled manner, while preserving the structure and content of the image. By applying this process multiple times, we can generate controlled, high-quality images that are based on the initial image, the noise we introduce, and the de-noising mechanism. When we combine diffusion models with the perceptual image compression model that was previously discussed, we can create a diffusion model with access to images in latent space, thus, latent diffusion models. These models function in a lower dimensional environment due to the removal of imperceptible details of an image due to the compression, making them more efficient since they only need to focus on the salient features of data.

The purpose of porting diffusion models to latent space is to reduce the dimensions in which computations must be performed, subsequently reducing the time and complexity of image generation. This is key to being an applicable, state of the art high-resolution image synthesis project.

1.3 Conditioning Mechanisms

Conditioning mechanisms allow for the diffusion models, which were previously discussed, to model conditional distributions of the form $p(z|y)$, where y is an input that is not an image, such as text or semantic maps. To make diffusion models more flexible as image generators, the authors augment the UNet backbone with a cross-attention mechanism and introduce a domain-specific encoder that projects y to an intermediate representation. The encoder τ_θ is used to project y to an immediate representation which is then mapped to the intermediate layers of the UNet using a cross-attention layer, by implementing Attention(Q, K, V) = softmax $(\frac{QK^T}{\sqrt{d}}) \cdot V$, where

$$Q = W_Q^{(i)} \cdot \varphi_i\left(z_t\right), K = W_K^{(i)} \cdot \tau_\theta(y), V = W_V^{(i)} \cdot \tau_\theta(y)$$

- $\varphi_i\left(z_t\right) \in \mathbb{R}^{N \times d_\epsilon^i}$ is an intermediate representation of the UNet
- $W_V^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i}$, $W_Q^{(i)} \in \mathbb{R}^{d \times d_r}$ and $W_K^{(i)} \in \mathbb{R}^{d \times d_r}$ are learnable projection matrices. These matrices are obtained during training and are the weights that are used in the project. For our purposes, we use the pre-trained weights that are provided, which are trained on the LAION-5B dataset which consists of 5.85 billion CLIP-filtered image-text pairs. [1]

1.3.1 Background on UNet

UNet is a neural network architecture that is, at a high level, composed of an encoder and decoder network. The purpose of the encoder network is to extract high-level features from the input image through convolutional layers, which consist of computing dot products between a filter and input data. The decoder network then gradually restores the spatial resolution of the image through upsampling and more convolutional layers. Thus, using UNet along with diffusion models allows for the creation of high-level feature maps that summarize relevant information that an image must have based on different forms of inputs through conditioning mechanisms.

2 Getting the Code Running on CPU

For the midterm, a major task beyond understanding the method was preparing and executing the code for latent diffusion models. The README states that a requirement for running the code is a GPU with at least 10GB of RAM. While this is an incredibly low requirement for the project that is being done, I have a MacBook Air with an M1 chip, which doesn't have a dedicated GPU. As a result, I had to make edits to the following files to ensure the project could run entirely on a CPU: "ddim.py, plms.py, modules.py, knn2img.py, sample_diffusion.py, txt2img.py." With these modifications, the expected time for text-to-image synthesis increased significantly from 15 seconds on a GPU to about 2 hours running solely on a CPU. Though this is an unfortunate increase in time, it is expected and understandable given the circumstances. Here are some examples of images generated with input text (provided below each group of images), produced locally with the modified code:

Figure 2: Snorlax in the Rain

[1] https://huggingface.co/CompVis

Figure 3: Dog Painting in the Style of A Starry Night

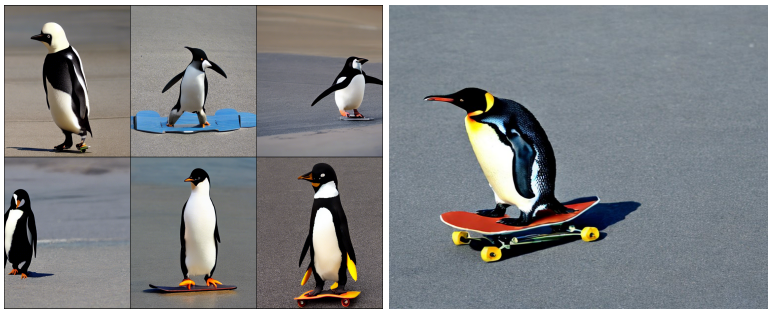


Figure 4: A Photograph of a Penguin on a Skateboard

Figure 5: A Photograph of an Astronaut Riding a Horse

Figure 6: Nature Artwork

Figure 7: Cat in a Hat

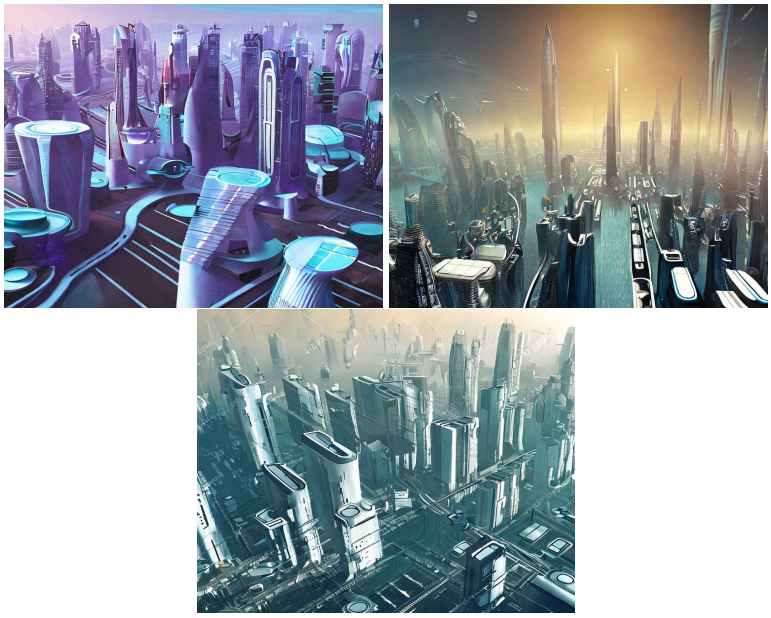


Figure 8: Futuristic City

Figure 9: Turtles Flying

3 Expanding The Conditioning Mechanisms

At this point, we have successfully replicated the results from High-Resolution Image Synthesis with Latent Diffusion Models and have obtained a high level understanding of the methods used in the image generation process. In order to move forwards with the understanding of conditioning mechanisms, we will take a closer look at how they work and then build off of the existing implementation to create and audio to image model.

3.1 A Closer Look at Conditioning Mechanisms

As discussed in section 1.3, the encoder τ_θ is a critical component of the conditioning mechanisms used in the text to image process. The immediate representation $\tau_\theta(y)$ is a dense vector with fixed length, meaning that it captures all relevant features of the text (dense), and contains a set amount of scalar values no matter what the input is (fixed length). Thus, this vector can be thought of as a compressed representation of the input text. This compressed representation is then used as input to the cross-attention layer, which, as previously discussed, combines it with the feature maps generated by the UNet backbone to guide the generation process of the high-resolution image.

3.1.1 Encoder Example

Let's say we input the text prompt "a bright red sports car driving on a highway". τ_θ is trained to create a vector,

$$[0.21, -0.14, 0.87, 0.76, -0.92, 0.33, 0.45, 0.19, -0.66, -0.58]$$

for example, which contains scalar values that represent different aspects of the input. Each element in the vector would correspond to a different feature such as the color of the car, size of the car, or the location of the car. Thus, each value of the vector $\tau_\theta(y)$ can be thought of as an abstract numerical representation of certain aspects of the text such as the color of an object, the size of an object, or the location of an object in a scene. This vector is then used to guide the image generation process.

3.2 Audio as a Conditioning Mechanism

The conditioning mechanism implemented in the supplemental code is built for text, so if we wanted to adapt the image generation code to work with audio as input instead, we would need to replace the text encoder τ_θ with an audio encoder that can produce effective dense, fixed length vectors. To do this, we could use a pre-trained audio feature extractor such as the VGGish model to extract the relevant audio features and then feed the features into a domain specific encoder such as a CNN or an RNN to map the input to a fixed-length vector. In theory, this vector could then replace the output of τ_θ and be used as the guide for the image generation process.

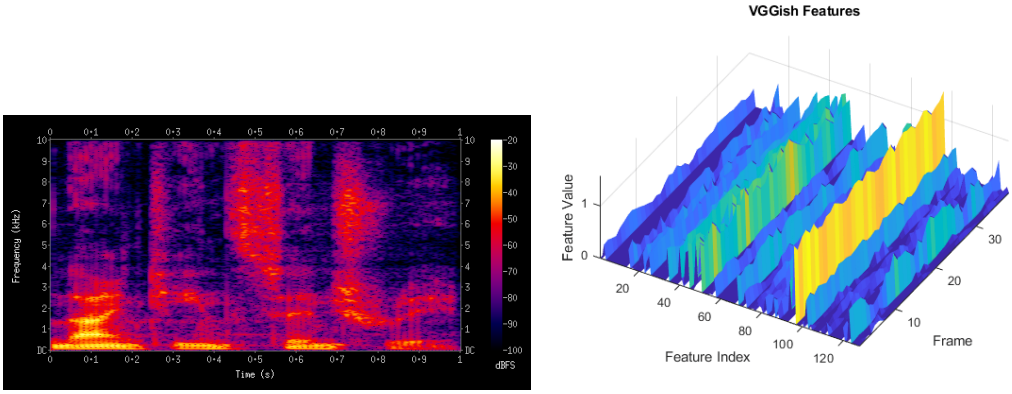
3.2.1 The VGGish Model

The VGGish model is a powerful tool for audio analysis, built upon a pre-trained convolutional neural network (CNN) specifically designed to capture the salient features from an audio signal. It is trained on a large-scale dataset called AudioSet, which contains millions of audio clips labeled with thousands of different sound classes. This extensive training allows the model to effectively recognize and process a wide variety of sounds.

Before the audio is fed into the network, it is preprocessed to create spectrograms, which are visual representations of the audio signal's frequency content over time. This is typically done using short-time Fourier transform (STFT) or Mel-frequency cepstral coefficients (MFCCs). Once the spectrograms are generated, the VGGish model chunks the input audio into segments of 0.96 seconds each to ensure manageable input sizes and to capture local temporal information.

Each chunk is then fed through a series of convolutional and pooling layers within the CNN, which are responsible for extracting higher-level features from the spectrograms. Convolutional layers apply filters to the input data to identify patterns and structures, while pooling layers reduce the spatial dimensions of the feature maps, enhancing the model's robustness and computational efficiency.

After passing through these layers, the resulting feature maps are flattened into a one-dimensional vector and passed through a fully connected layer. This process produces a final 128-dimensional embedding vector that represents the most important features of the input audio segment.



(a) An example of a spectrogram

(b) Feature extraction visualization

Figure 10: VGGish

To use the output, we would need to train our own encoder to map the 128-dimensional embedding vector to a fixed-length vector that can be used instead of the existing text encoder. This would not only be a tedious process, but it would require us to have a set of embedding vector to fixed length vector pairs that we could use for training, which we do not have. Thus, we will implement a different method of using audio to begin the image generation process.

3.3 Implemented Method

In the process of implementing an audio-to-text layer, I opted to use Google's Speech Recognition API instead of developing a brand new encoder. This powerful tool converts spoken language into written text by employing two primary components: the Acoustic Model and the Language Model.

The Acoustic Model focuses on transforming the audio signal into a sequence of phonemes, the basic sound units in a language. It has been trained on a diverse range of data sources, including the Google Voice Search dataset, YouTube videos, and phone call recordings. This extensive training allows the model to recognize various accents, dialects, and speaking styles, thus enhancing its performance and adaptability.

The Language Model, on the other hand, is responsible for converting the phoneme sequence produced by the Acoustic Model into a coherent sequence of words. It is trained on a wide variety of online text data, such as books, articles, and other written content, enabling it to capture the intricacies and nuances of language. One of the key functions the Language Model uses to determine the most likely sequence of words is the argmax function. By analyzing the input phoneme sequence and the probabilities assigned to each possible word sequence, the argmax function selects the sequence with the highest probability, effectively transcribing the spoken language into text.

Once the text has been transcribed, we can feed it into the original encoder τ_θ. By performing this process we can begin the image generation process with audio input while maintaining the accuracy and efficiency of High-Resolution Image Synthesis with Latent Diffusion Models.

4 Audio to Image Examples

Note: Under each image is a link to the audio used as input. Each recording is a different person's voice.

Figure 11: Audio File

Figure 12: Audio File

Figure 13: Audio File

5 Conclusions

This project successfully replicated the results from "High-Resolution Image Synthesis with Latent Diffusion Models," leading to a foundational understanding of conditioning mechanisms that explain how different forms of input can guide image generation. By modifying the existing code to run fully on the CPU, a wide variety of images were generated based on text inputs. Building upon the text-to-image code, an audio-to-text layer was added, which expanded the project's capabilities by generating a diverse range of images based on audio samples from different people. This achievement showcases the versatility and potential of the implemented models for various applications.

6 Literature Review

This literature review focuses on the foundational work and key papers upon which the paper "High-Resolution Image Synthesis with Latent Diffusion Models" is built. We explore the developments and concepts that have led to the creation of Latent Diffusion Models (LDMs) for high-quality image synthesis.

Key Papers and Concepts:

1. Denoising Diffusion Probabilistic Models (DDPMs): Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. arXiv preprint arXiv:1503.03585.

   This paper introduces the concept of diffusion models for unsupervised learning, specifically the Denoising Diffusion Probabilistic Model (DDPM). It outlines a method to train a deep neural network to perform denoising by leveraging nonequilibrium thermodynamics. This work forms the basis for the development of Latent Diffusion Models in the target paper.

2. Denoising Score Matching: Vincent, P. (2011). A connection between score matching and denoising autoencoders. Neural Computation, 23(7), 1661-1674.

   Vincent's paper discusses the connection between score matching and denoising autoencoders. The concept of denoising score matching forms a key component in the development of diffusion models and the LDM proposed in the target paper.

3. Generative Adversarial Networks (GANs): Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. Advances in Neural Information Processing Systems, 2672-2680.

   This seminal paper introduces Generative Adversarial Networks (GANs), a powerful generative modeling technique that has been widely used for high-quality image synthesis. The LDM paper references GANs to provide context and comparison to their own approach.

4. Variational Autoencoders (VAEs): Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. ArXiv preprint arXiv:1312.6114.

   Kingma and Welling's paper presents Variational Autoencoders (VAEs), another generative modeling technique that has been used for image synthesis. VAEs serve as another point of comparison for the proposed LDMs in the target paper.

5. High-Quality Image Synthesis with GANs: Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. ArXiv preprint arXiv:1710.10196.

   Karras et al.'s paper demonstrates the potential of GANs for high-quality image synthesis by introducing a progressive growing technique. This paper provides a point of comparison for the performance of LDMs in the target paper.

6. Style-Based GANs: Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4401-4410.

   This paper presents a style-based generator architecture for GANs, further improving the quality of synthesized images. The style-based GAN incorporates a mapping network to convert input noise vectors into intermediate latent codes, which are used to modulate the activations of the generator. This method allows for better control over the generated images' style and content while also enhancing image quality. The target paper, "High-Resolution

Image Synthesis with Latent Diffusion Models," compares the performance of LDMs with state-of-the-art GANs, including style-based GANs, to demonstrate the effectiveness of LDMs in generating high-quality images.

This literature review has explored the key foundational papers and concepts that have contributed to the development of Latent Diffusion Models for high-resolution image synthesis. The research in denoising diffusion models, generative adversarial networks, and variational autoencoders has provided the basis for LDMs' novel approach to image synthesis. By understanding the context of these works, we can appreciate the advancements and contributions made by the target paper, "High-Resolution Image Synthesis with Latent Diffusion Models."

7 References

[1] compvis/stable-diffusion. GitHub, https://github.com/compvis/stable-diffusion.

[2] Iashin AI. "VGGish Model - Pretrained audio feature extractor for TensorFlow." Iashin AI, n.d., https://iashin.ai/video_features/models/vggish/.

[3] Laion. "Laion | Perceptual Losses and Beyond." Laion, 11 Dec. 2020, https://laion.ai/blog/laion-5b/.

[4] ModeratePrawn. "fixing broken test." GitHub, 28 Feb. 2022, https://github.com/ModeratePrawn/stable-diffusion-cpu/pull/1/files.

[5] Rombach, Robin, et al. "High-Resolution Image Synthesis with Latent Diffusion Models." arXiv preprint arXiv:2204.06515 (2022).

[6] Simplilearn. "Speech Recognition in Python - Tutorial with Example." Simplilearn, 13 Oct. 2021, https://www.simplilearn.com/tutorials/python-tutorial/speech-recognition-in-python.

Note: the paper that this is based on, "High-Resolution Image Synthesis with Latent Diffusion Models", is written by Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjőrn Ommer.