

# Tech Explainer

## DIFFUSION MODELS: HOW TEXT-TO-IMAGE GENERATION WORKS

### INTRODUCTION

Image generation has reached unprecedented quality and scale in the short time that the technology has been accessible to the general public. From futuristic art exhibits to deceptive deepfakes, AI-generated images have been the subject of much awe and scrutiny over the last few years. But how exactly does it work?

When humans are asked to draw an object based on verbal instructions, we engage in an imaginative process. We start by understanding what is described by the words in the instructions. Perhaps our brains simultaneously begin visualizing the object described by the words as we read or hear them. Some people may recall many examples of the object described and draw an image that combines features of the various examples in their memory, whereas some people may draw the first example of the object that comes to mind. A person who has never seen an example of the object described may still be able to compose a believable drawing based on what they know about the object from the description.

Generative AI models that generate images from users' text prompts do not share this human imaginative process, yet are still capable of creating new, never-before-seen images based on user prompts.

AI image-generation models such as DALL-E<sup>1</sup> are comprised of neural networks that are trained on large quantities of training data that include both images and text. As a result of the training process, the models contain mathematical representations of patterns and associations between language and images that are then deployed to generate a new image when a prompt is inputted. A common misunderstanding of AI image generation is that the model constructs an image by combining specific features gleaned from snippets of stored training data piece by piece in a deterministic process—this is not how it works. At a high level, the image generation process can be broken down into two parts: first, the model must “understand” a text prompt to guide the image generation process and, second, the model must use that “understanding” to create a clear image. These two parts involve two distinct model architectures, which have different training processes and different inference steps. The first part of the process uses a transformer-based text encoder, similar to those used in models that generate text, such as GPT models described in our previous *Tech Explainer*<sup>2</sup>. The second part of the process involves a

---

1 DALL-E is a model developed by Open AI that specializes in generating images from text captions. *DALL-E: Creating Images from Text*, OPENAI (Jan. 5, 2021), <https://openai.com/index/dall-e/>.

2 David J. Kappos & Sasha Rosenthal-Larrea, et al., *Tech Explainer: How ChatGPT Understands Context: The Power of Self-Attention* (Feb. 2024), <https://www.cravath.com/a/web/25fvkMDn6Q8MyAtaPpsLf2/8BaHMZ/cravath-tech-explainers-how-chatgpt-understands-context-022024.pdf>.

different type of model called a “diffusion model”, which represents a significant development in the quality and scale of image generation<sup>3</sup>. In this *Tech Explainer*, we will walk through the technical steps of text-to-image generation using text encoder neural networks with diffusion models.

## A PICTURE IS WORTH A THOUSAND WORDS

In order to generate images based on text prompts, a model must be able to interpret the text prompt to guide the image generation process. AI models cannot “understand” language instructions the way humans do, so they must convert the words in the prompt into a series of mathematical representations of the various features of each word in the prompt. Written text must first be converted into a string of “tokens” (generally, a word or part of a word), which correspond with “token embeddings” (a set of numbers that represent attributes of each word or part of a word, which were “learned” during the training process)<sup>4</sup>. Then, these embeddings are passed through a transformer-based text encoder (a type of neural network). The text encoder produces a numerical representation that captures the meaning of the entire text prompt, which can be thought of as a map of the relationship between the words, syntax and context of the text prompt. These text encoders are trained on pairs of images and corresponding text<sup>5</sup> (e.g., an image of a green apple, paired with the text “a granny smith apple”) to capture the relationship between words and their corresponding images.<sup>6</sup>

The result of this training process is that text becomes “encoded” in terms of images or, put another way, the text is represented in a multidimensional vector—e.g., a set of hundreds or thousands of component numbers referred to as “weights”—that correspond roughly to the various features of the text and image pair<sup>7</sup>, such that the weights associated with a given text sequence closely correspond to the weights associated with the image with which it was paired, as illustrated below. In the resulting set of weights, the numbers represent both the text and attributes of an image that corresponds with the text.

---

3 Note, however, that diffusion models are not the only way to generate images. Other techniques, such as Generative Adversarial Networks (“GANs”), were and are still used for certain image generation tasks, but diffusion models are the common architecture used in popular image generation applications because of their ability to generate a wide range of highly realistic images with less training.

4 Please refer to our previous *Tech Explainer* for a more in-depth description of tokens and token embeddings for transformer models. See David J. Kappos & Sasha Rosenthal-Larrea, et al., *Tech Explainer: How ChatGPT Understands Context: The Power of Self-Attention* (Feb. 2024), <https://www.cravath.com/a/web/25fvkMDn6Q8MyAtaPpsLf2/8BaHMZ/cravath-tech-explainers-how-chatgpt-understands-context-022024.pdf>.

5 Text encoders do not necessarily need to be trained on text-image pairs, but several commonly used image generators take this approach. DALL-E and Stable Diffusion use text encoders trained on text-image pairs. Other image generators, such as Google’s Imagen use large language models, which are trained on text alone and not trained on images. See Chitwan Saharia, et al., *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding* (May 23, 2022), <https://arxiv.org/pdf/2205.11487>.

6 Note that just as text encoders represent text in a way to predict the corresponding image, image encoders represent images in a way to predict the corresponding text description. These two concepts are two sides of the same coin, and some models are trained to do both. For purposes of text-to-image generation, the text embeddings are the most important, and will be the focus of this *Tech Explainer*. But note that there are many use cases for image-to-text generation as well, such as image classification and computer vision.

7 The individual weights within the encoding are somewhat inscrutable, as they are derived through training, and often do not map neatly to particular identifiable attributes of the word or image. There are additional weights that are adjusted through training that are beyond the scope of this *Tech Explainer*.

To illustrate, we will walk through a simplified example. Let’s say we have a photograph captioned “peaceful lake”; this caption and photograph would form a text and image pair used in training. The tokens, embeddings and encodings could appear as follows:


	Text	Image
Training Data	peaceful lake	
Tokens	[peace] [ful] [lake]	N/A
Embedding <sup>8</sup>	[0.12, -0.45, 0.87, . . .]#peace [0.15, 0.56, 0.23, . . .]#ful [-0.34, 0.89, 0.18, . . .]#lake	[0.13, -0.87, 1.51, . . .]
Encoding	[0.71, 0.55, -0.42, . . .]	[0.70, 0.52, -0.48, . . .]

FIGURE 1 Illustrative table showing training data and corresponding tokens, token embeddings and text encodings.

The photo above could be described in a multitude of ways by a human observer, but the model training process is limited to the data that is supplied to it; in this case, the two pieces of data in the pair—the photo and the text caption “peaceful lake”—are ingested by the model.

The numbers in the encodings map the text and images to a point in latent space—we can conceptualize this as a graph. The order and values of the weights determine their position on the graph. For example, the first number in the text encoding example above (0.71) aligns with the first number in the image encoding (0.70). Through training, the coordinates of the encodings of corresponding captions and images are adjusted to be closer in position on the graph. Similar text sequences (*e.g.*, “peaceful lake” and “tranquil lake”) can be expected to have encodings that are close to each other, and similar images (*e.g.*, a photo of Lake Tahoe and a photo of Lake George) can be expected to have encodings that are close to each other as well. Conversely, training moves the positions of unrelated text and images further apart on the graph.

During training, the model is given a set of training images and training text. The model is prompted with a particular text prompt from the set of text captions included in the text-image pairs used for training, and tasked with selecting the image that corresponds with the text from within the set of training images. The model generates an output of what it determines is the corresponding image

<sup>8</sup> The numbers (or “weights”) provided in this table and in later examples are purely illustrative values for the purpose of simplifying the examples. Token embedding values can be many more decimal places long, and each embedding consists of more than three numbers.

by taking the dot product (*i.e.*, matrix multiplication, illustrated below) of the training text encoding and the training image encoding, and picking the image that produces the highest dot product (*i.e.*, the highest number answer resulting from the matrix multiplication calculation, compared to others multiplied by the same text encoding). The output is evaluated against the known correct text-image pair, and then the model’s weights are adjusted so that the model generates a more desirable or “accurate” output based on the same prompt.<sup>9</sup> For example, if the model is given “peaceful lake” text input, and the model predicts an image of a sunflower—that would be an inaccurate output, and the model’s weights would be adjusted to move the encodings of the “peaceful lake” text and sunflower image further from each other. The goal of these mathematical adjustments is to detect errors in prediction and to tune the weights to minimize error (or “loss”) across all training examples for the given task. This ingestion, testing and adjustment process is iterated until the model’s weights capture the association between these words and pictures that optimizes the output (in our simplified example, correctly identifying the training image of the lake corresponds with the training caption “peaceful lake”).




		Training images and image encodings		
		 [-0.87, -0.30, 0.42, . . .]	 [0.70, 0.52, -0.48, . . .]	 [0.35, -0.89, -0.30, . . .]
Training text and text encodings	“sunflower” [-0.90, -0.35, 0.40, . . .]	<b>1.06</b> Dot product of “sunflower” and image of a sunflower	-1.00 Dot product of “sunflower” and image of a peaceful lake	-0.12 Dot product of “sunflower” and image of a baby deer
	“peaceful lake” [0.71, 0.55, -0.42, . . .]	-0.96 Dot product of “peaceful lake” and image of a sunflower	<b>0.98<sup>10</sup></b> Dot product of “peaceful lake” and image of a peaceful lake = 0.71*0.70 + 0.55*0.52 + -0.42*-0.48 + . . .	-0.12 Dot product of “peaceful lake” and image of a baby deer
	“baby deer” [0.38, -0.85, -0.33, . . .]	-0.21 Dot product of “baby deer” and image of a sunflower	-0.02 Dot product of “baby deer” and image of a peaceful lake	<b>0.99</b> Dot product of “baby deer” and image of a baby deer

FIGURE 2 Illustrative table showing the relationship between the text encoding and image encoding based on three example text-image pairs.

In the above hypothetical example, the dot product of the encoding of “peaceful lake” with the encoding of an image that is *not* a lake is a lower number, while the product of matrix multiplication of the encodings of “peaceful lake” with the image embeddings of the photo of the peaceful lake returns a high value of 0.98. Text inputs with similar meaning or overlapping concepts, such as “pond” or “river”, may result in a higher dot product value than a completely unrelated word such as “sunflower” or “baby deer” because, over a training process that includes millions of images of bodies of water, the

<sup>9</sup> Weights are adjusted during training through a process called “backpropagation” and “gradient descent”. This is a complex, iterative process that is outside the scope of this *Tech Explainer*.

<sup>10</sup> For example, the first number in the text encoding is multiplied with the first number in the image encoding, plus the second number in the text encoding multiplied with the second number in the image encoding, etc.

concept of a body of water will have found its way into the embeddings, but the value will not be as high as the original caption of “peaceful lake”, which was part of the original training pair. Similarly, “sunflower” and the sunflower image have the highest value, and “baby deer” and the baby deer image have the highest value in their respective rows.

Through this iterative training process, the models’ weights can go through millions of small adjustments from training on hundreds of millions of combinations of image-text pairs.<sup>11</sup> In image generation, a picture is worth not only a thousand words, but also several thousand numbers, which direct an AI model to determine which words statistically are more or less likely to correspond with said picture.

We have just described at a high level the encoder training process for one image-text pair. At inference time (when a user inputs a new text prompt into a model), the model’s weights are already optimized by the various image-text data from the training process. Text prompts are tokenized, and token embeddings are passed through the encoder model to produce a vector output. This vector output is based on the training process utilizing multiple text-image pairs. The text encoder output is analogous to an “understanding” of the text prompt, but there is no image stored in the encoding itself, and there is no *image* created corresponding to these numbers yet—the actual output image is generated in the next step, the diffusion model.

## DIFFUSION MODELS FOR IMAGE GENERATION

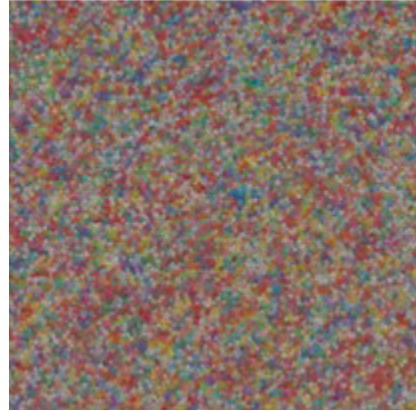
Now that we have a vectorized representation of the text, the text encoding, we can discuss how it is used as the input to a diffusion model to produce a final image. Whereas a text generation model creates outputs by generating one word or part of a word at a time based on context, diffusion models create outputs through a series of gradual alterations applied to an entire image, as a whole, in order to create the final output image.

Diffusion models (or, more formally, “denoising diffusion probabilistic models”) repurpose the concept of “diffusion” from physics. In physics, thermodynamics researchers use diffusion to predict the movement of particles from an organized state to a more random state (low to high entropy) and, vice versa, from a random state to an organized state. In image generation, machine learning researchers cleverly use a similar mathematical concept to create diffusion models, which generate the output image by “predicting” how an initial random state turns to a structured output—a clear and believable image. We can think of a clear image as containing a high concentration of organized information (low entropy), and the noisy image as one where much of the original information has been randomized and lost (high entropy).

The diffusion model takes as its input the text encoding from the user prompt along with randomly initialized noise (“Gaussian noise”). To the human eye, Gaussian noise would just look like a grainy image with random pixel colors, like a staticky television screen with no discernible image. This grainy image is the starting point of the image to be generated. The purpose of the diffusion model is to “find” an image within the noise by slowly adjusting the pixels until they resemble a clear image based on the user’s prompt, a process called “denoising”.

To illustrate, the following image approximates Gaussian noise:

---



---

**FIGURE 3** An approximation of Gaussian noise generated by using DALL-E with the prompt “Please generate an image completely filled with colorful Gaussian noise.”<sup>12</sup>

Before we dive into the denoising steps at inference time (when a user generates an image from a new prompt), we will first review the training process for diffusion models, so it is easier to understand what these models are trained to do.

Because we do not typically have “noisy” images laying around, the first step of training is to prepare a set of training images based on the training data. We do this by incrementally adding “noise” to the training data image until we have a set of images of varying clarity and noise. Each step of adding more noise is called a “timestep”, and this process of adding noise gradually is called “forward diffusion”. To simplify our example, say we have four timesteps<sup>13</sup>: at timestep 1, we have our original image (a clear image from our training dataset text–image pair); at timestep 2, we add some “noise” to the image, meaning a random selection of the original image’s pixels will have their color values altered by random amounts, resulting in modified colors throughout the image; at timestep 3, the model adds even more noise to the image from timestep 2, creating a noisier version of the image, but the vague outline of the original image is still discernable; and at timestep 4, the image’s pixels have so much random noise that the final noisy image no longer bears meaningful relation to the original image.

---

<sup>12</sup> The authors selected this response among a number of other outputs as the example that best visually resembled Gaussian noise.

<sup>13</sup> We have simplified to four timesteps for simplicity and to help readers visualize the difference between each timestep. The number of timesteps in reality is much higher, and there would be many gradations between each of these four sample images shown in Figure 4. For example, OpenAI’s research paper provides an example with 1,000 timesteps. See Prafulla Dhariwal & Alex Nichol, *Diffusion Models Beat GANs on Image Synthesis* (June 1, 2021), <https://arxiv.org/pdf/2105.05233>.

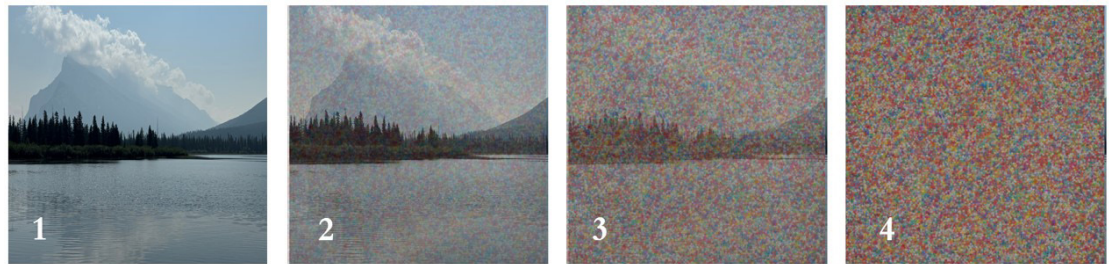
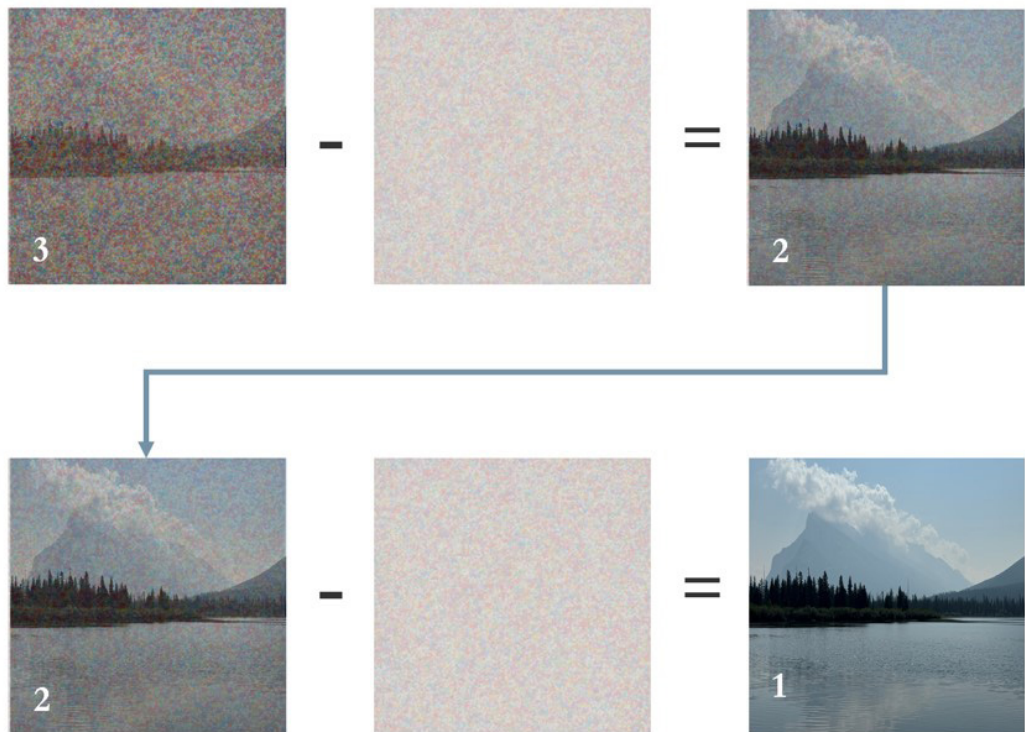


FIGURE 4 An approximation of forward diffusion in four timesteps, with progressively more noise.

Each timestep has an associated noisy image; the model’s training objective is to reverse this process, starting with the noisy image that we have identified with a timestep number, “predicting” the noise that was added through each of the previous timesteps, and subtracting that noise to reveal the original clear image. The training process optimizes the image prediction by adjusting the weights in the diffusion model neural network until the predicted image matches the original training data image. We would start, for example, by inputting the image from timestep 3 into the model, with the objective of getting the model to output the original training image from timestep 2. The model does this by predicting what “noise” was added between timestep 2 and 3, and subtracting that noise from timestep 3—if it predicted the noise accurately, the resulting image should be the timestep 2 image. Because each image is associated with a timestep value, the model takes into account how many steps the input text is away from the original timestep. A higher timestep would suggest more noise must be removed, since a predefined incremental amount of random noise is added with each timestep.<sup>14</sup>

---

<sup>14</sup> Note that the amount of noise added at each timestep does not need to be linear (i.e., the images do not need to increase by the same amount of noise in each timestep). The rate at which noise is added just needs to be predefined by some consistent formula (the “noise schedule”). For example, research by OpenAI showed that it is more effective to add noise more slowly at the earlier timesteps and faster at the later timesteps, because the marginal benefit of having many similar noisy photos was low, and it was inefficient to train on later timestep images. See Alex Nichol & Prafulla Dhariwal, *Improved Denoising Diffusion Probabilistic Models* (Feb. 18, 2021), <https://arxiv.org/pdf/2102.09672>.



**FIGURE 5** An illustration of the subtraction of noise from timestep 3 to reveal the image in timestep 2, and subtraction of noise from timestep 2 to reveal the image in timestep 1.

Just as the noise is added incrementally in the forward diffusion stage, noise is removed incrementally in small amounts, timestep-by-timestep in the denoising stage of training. The model takes the image at timestep 3, for example, and predicts the “noise” that was added.<sup>15</sup> Then, the model subtracts this noise from the image, leaving the image at timestep 2. Lastly, the model predicts what noise was added between timestep 1 and timestep 2 and subtracts that noise from the image in timestep 2, leaving the clear image at timestep 1.

<sup>15</sup> One advancement in diffusion models is the use of “latent variable space”, which moves away from tracking images at the pixel level, and instead encodes images by semantic concepts. This has lowered training costs and enabled faster inference speeds. See Rombach, et al., *High-Resolution Image Synthesis with Latent Diffusion Models* (Apr. 13, 2022), <https://arxiv.org/pdf/2112.10752>.



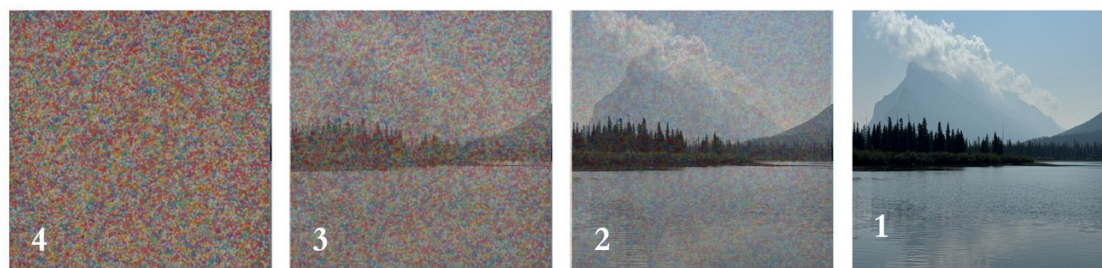


FIGURE 6 An approximation of reverse diffusion in four timesteps, which are the steps the model would take to predict the clear image from the noisy input.

At the end of training, the weights of the diffusion model are adjusted such that we would expect to be able to input a noisy image from, for example, timestep 3, and the model would be able to accurately remove the noise and predict the original clear image at timestep 1.<sup>16</sup> While this process can be completed with just the image inputs (basic diffusion), it can also be paired with text classifiers to “guide” the reverse diffusion toward the original image (“guided diffusion”). Researchers found that training the diffusion model with classifier guidance improves the precision of the prediction.<sup>17</sup> In other words, if we started with the image in timestep 3 and trained the denoising model along with the “peaceful lake” caption, the model would likely produce an output that is closer to the original image if it is told the end image contains “peaceful lake” than if the model was not given this additional context. This “guided diffusion” concept is central to how text prompts are able to “guide” diffusion models to generate the desired output in text-to-image generation applications.

## USING DIFFUSION MODELS TO GENERATE NEW IMAGES

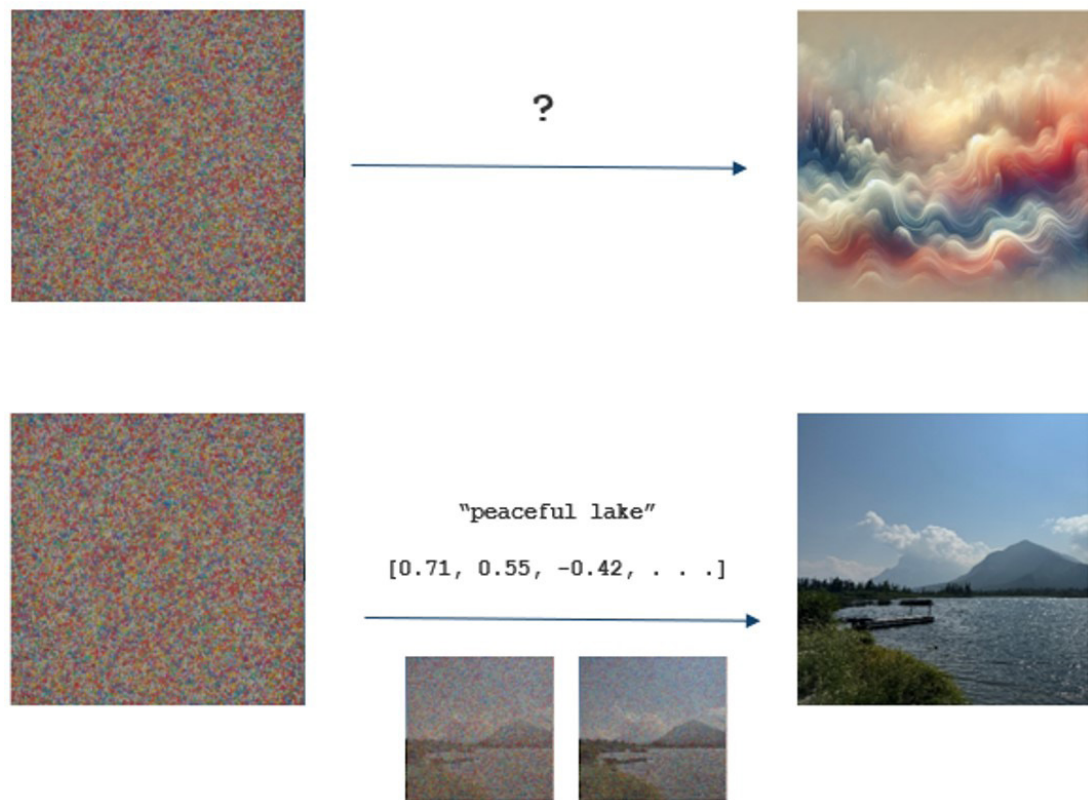
Now that we have covered how diffusion models are trained on existing images, we can discuss how the diffusion model can generate a new image that was not actually made from adding noise to a training image.

At the inference stage, we are asking the model to generate a clear image that did not previously exist, where there is no preexisting timestep 1 image (or noisy timestep images, for that matter). Yet, the model follows the same process as it would if it were to denoise a training image: incremental removal of predicted noise until it reaches the first timestep, a clear image. The model takes the starting point image, “predicts” what the “added” noise might be, and then subtracts it from the image.

In order to generate an image that is described by the user’s prompt rather than generating a random image, the model needs to take into account the text encodings discussed in the earlier section. The text encoding of the user’s prompt guides the diffusion model at each timestep such that the information contained in the encoding influences the prediction of the denoised image. On the other hand, as illustrated below, if the input to the diffusion model is completely random, the generated image would also be random, because there is no contextual instruction guiding the diffusion process.

<sup>16</sup> These training outcomes are in large part reliant on a convolutional neural network called a UNet and several additional layers of neural networks that control the diffusion process, which are beyond the scope of this paper.

<sup>17</sup> Prafulla Dhariwal & Alex Nichol, *Diffusion Model Beat GANs on Image Synthesis* (Jun. 1, 2021), <https://arxiv.org/pdf/2105.05233>.



**FIGURE 7** An illustration of the impact of text classifiers in diffusion model inference. The image on the top right is intended to represent a random image, but was actually generated using DALL-E by uploading the image of Gaussian noise with the text prompt “Please generate an interpretation of what image might emerge from this Gaussian noise.”

After denoising is complete, the resulting image should resemble the subject of the user’s prompt. The denoised image is subsequently passed through another neural network to further refine the image into a high resolution image, and this high resolution version of the diffusion model output is the image the user receives as the generative AI output.

### LEGAL SIGNIFICANCE

Image generation has reached unprecedented quality and scale in the short time that the technology has been accessible to the general public. From futuristic art exhibits to deceptive deepfakes, AI-generated images have been the subject of much awe and scrutiny over the last few years. The legal issues presented by generative AI, and AI image generation in particular, are complex and evolving. This *Tech Explainer* aims to present a nuanced explanation of how image generators produce images that correspond with user prompts, and demystify how they can produce new, never-before-seen works. By walking through the technical steps, we hope to quash the oversimplification that AI image generators are merely collaging together partial copies of training images, and emphasize that user prompts are integral to the images generated by AI.

In the short time that they have been publicly accessible, diffusion models have raised many complex questions about IP rights. Indeed, the U.S. Copyright Office has denied copyright registrations for AI-generated works, and there is ongoing litigation regarding alleged violations of copyrights and name, image and likeness rights by AI-generated images.<sup>18</sup> Plaintiffs in copyright litigation have alleged infringement in both training and inference: copying of training images to compile them into the training dataset, encoding of images with paired text, training the diffusion model to computationally produce images and producing outputs that resemble copyrighted works.<sup>19</sup>

As AI image generation technologies become more advanced, and theories of infringement are further developed, lawyers must understand how they work in order to advise on legal matters involving these issues and make well-informed arguments when advocating for their clients.

---

<sup>18</sup> *Thaler v. Perlmutter*, 1:22-cv-01564 (D.D.C. 2022) (challenging the Copyright Office’s decision denying registration for plaintiff’s AI-generated work); see also Copyright Registration Guidance: Works Containing Material Generated by Artificial Intelligence, 88 Fed. Reg. 16190 (Mar. 16, 2023) (emphasizing the requirement of human authorship and stating that works containing AI-generated material will depend on “how the AI tool operates and how it was used to create the final work”).

<sup>19</sup> See *Getty Images (US), Inc. v. Stability AI, Inc.*, 1:23-cv-00135-JLH (D. Del. 2023).

Sasha Rosenthal-Larrea

+1-212-474-1967

srosenthal-larrea@cravath.com

Lucille Dai-He

+1-212-474-1860

ldaihe@cravath.com

**CRAVATH, SWAINE & MOORE LLP**

**NEW YORK**

Two Manhattan West

375 Ninth Avenue

New York, NY 10001

T+1-212-474-1000

F+1-212-474-3700

**LONDON**

CityPoint

One Ropemaker Street

London EC2Y 9HR

T+44-20-7453-1000

F+44-20-7860-1150

**WASHINGTON, D.C.**

1601 K Street NW

Washington, D.C. 20006-1682

T+1-202-869-7700

F+1-202-869-7600

This publication, which we believe may be of interest to our clients and friends of the firm, is for general information only. It should not be relied upon as legal advice as facts and circumstances may vary. The sharing of this information will not establish a client relationship with the recipient unless Cravath is or has been formally engaged to provide legal services.

© 2025 Cravath, Swaine & Moore LLP. All rights reserved.