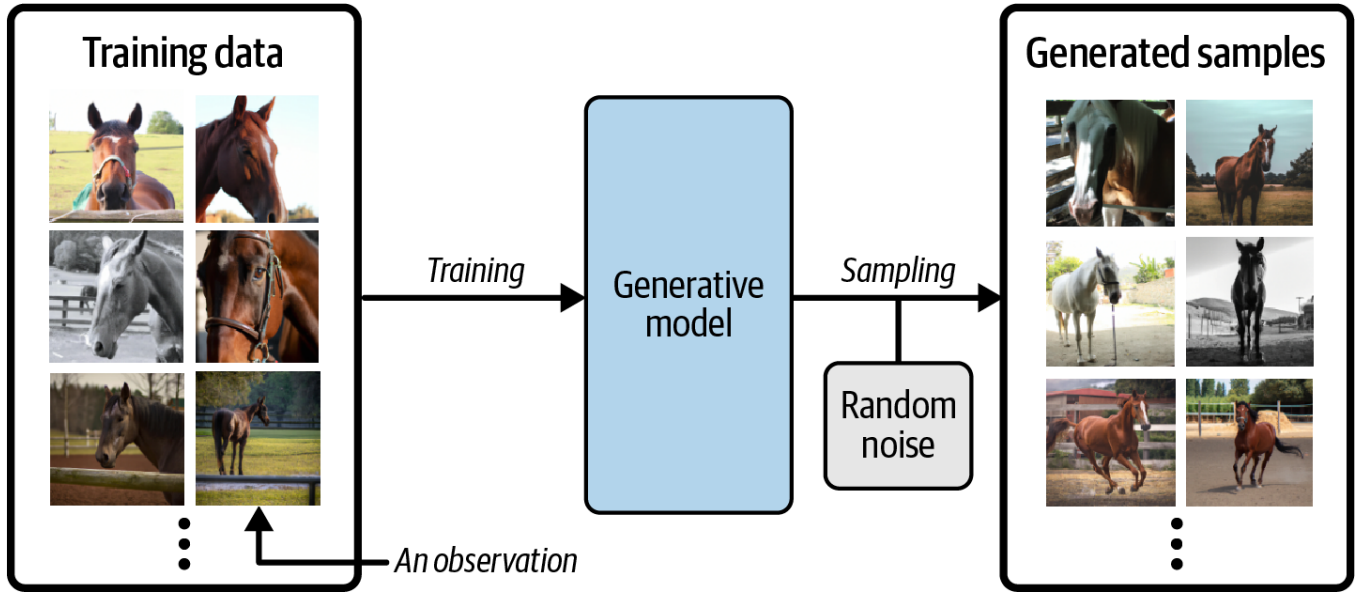
Generative AI

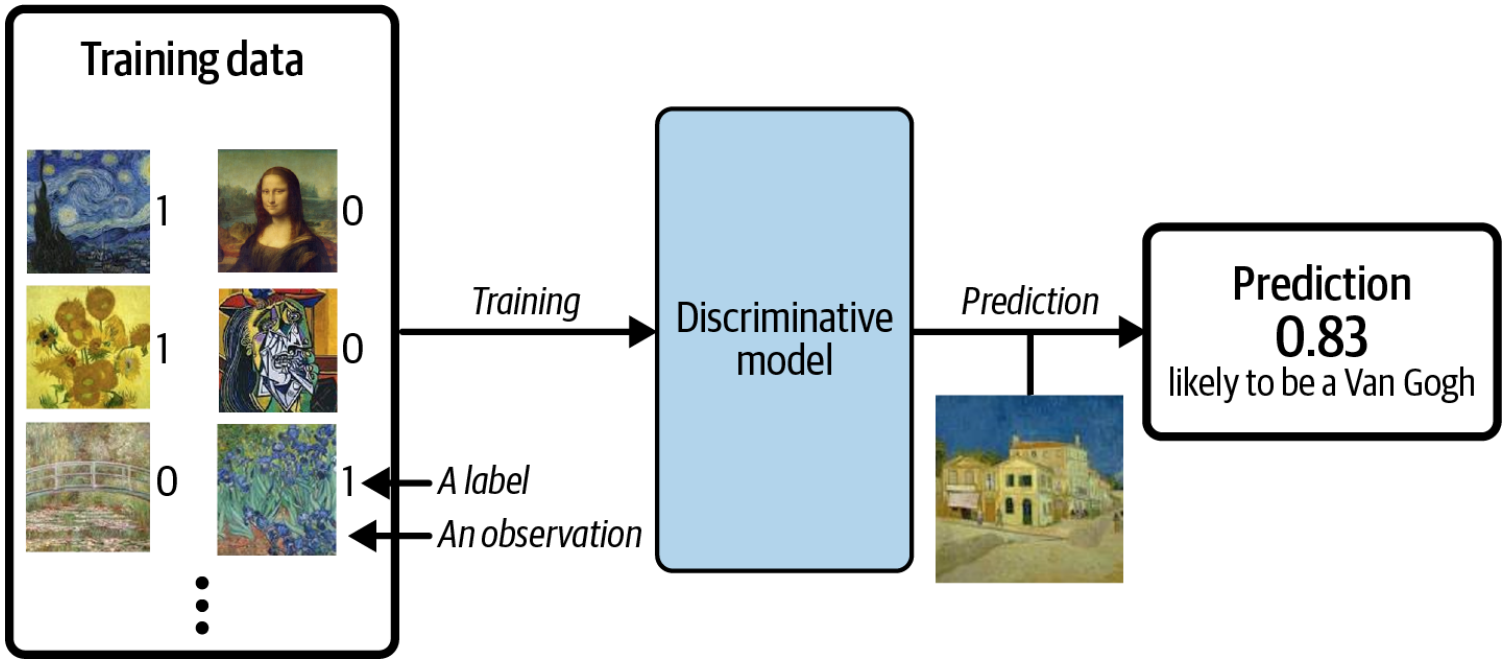
What is Generative AI?



Generative modeling is a branch of machine learning that involves training a model to produce new data that is similar to a given dataset.

Generative & Discriminative

Discriminative modeling estimates
*That is, discriminative modeling aims to model the probability of a label **'y'** given some observation **'x'.***



Probabilistic vs Deterministic

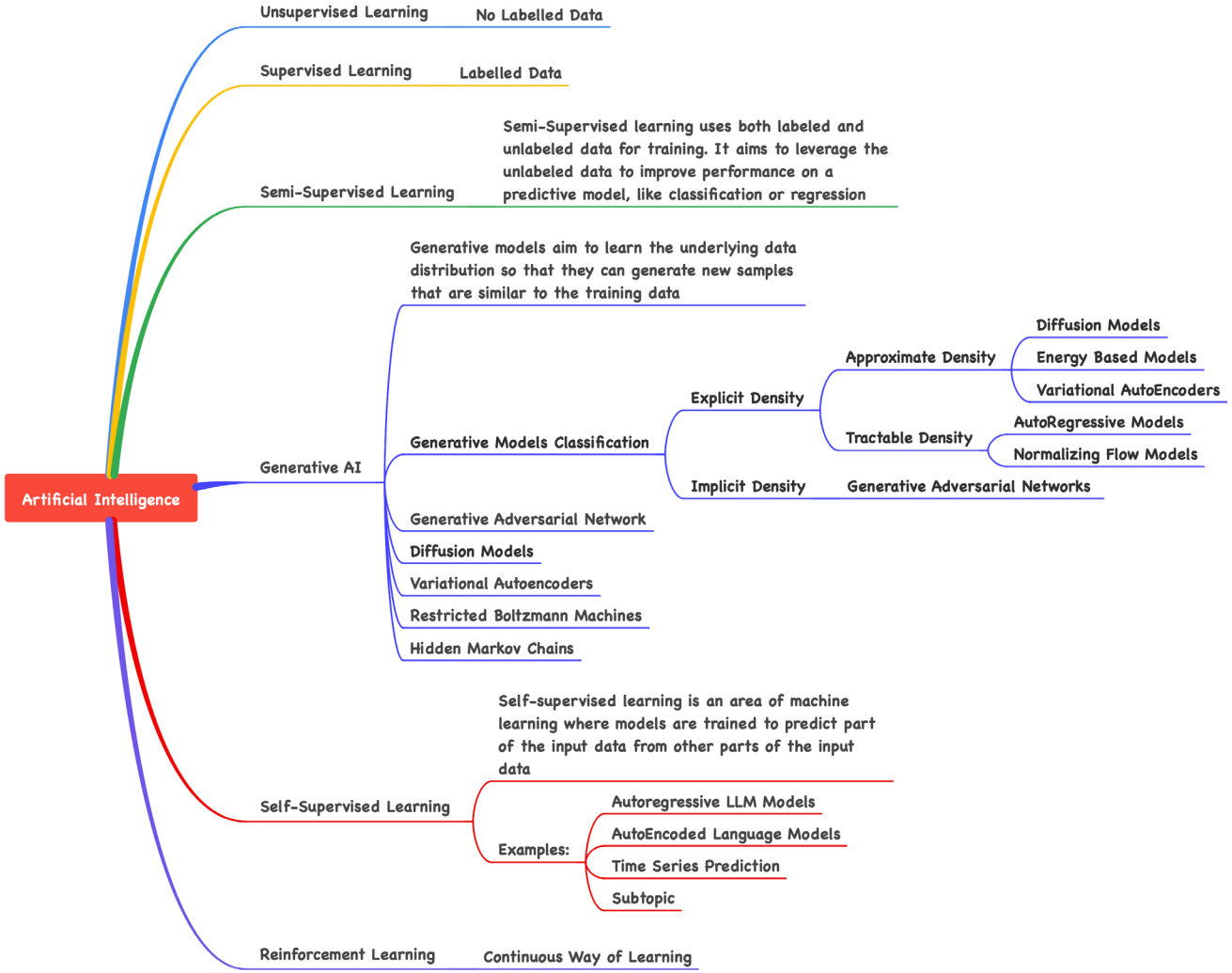
bepec CT Programs

- Probabilistic models account for uncertainty and randomness. These models provide a probability distribution over possible outcomes.

- Given the same input, deterministic models will always produce the same output. These models do not account for uncertainty or randomness; they make a specific prediction.

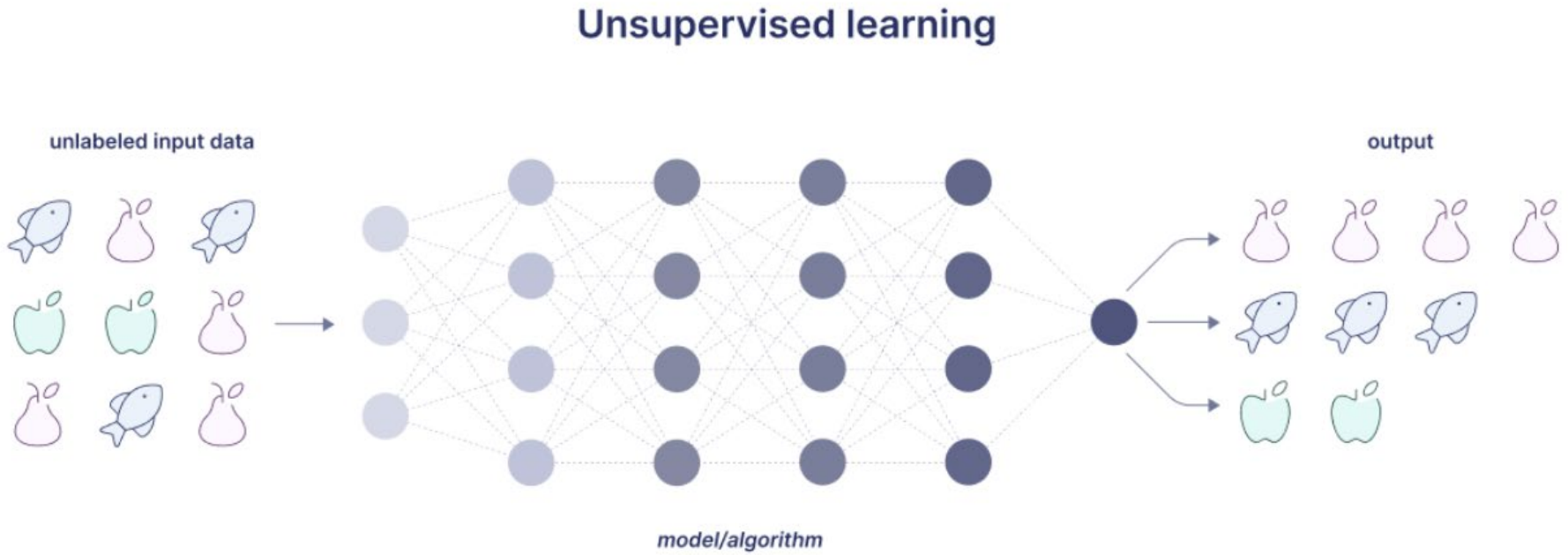
Types of Artificial Intelligence

bepec CT Programs



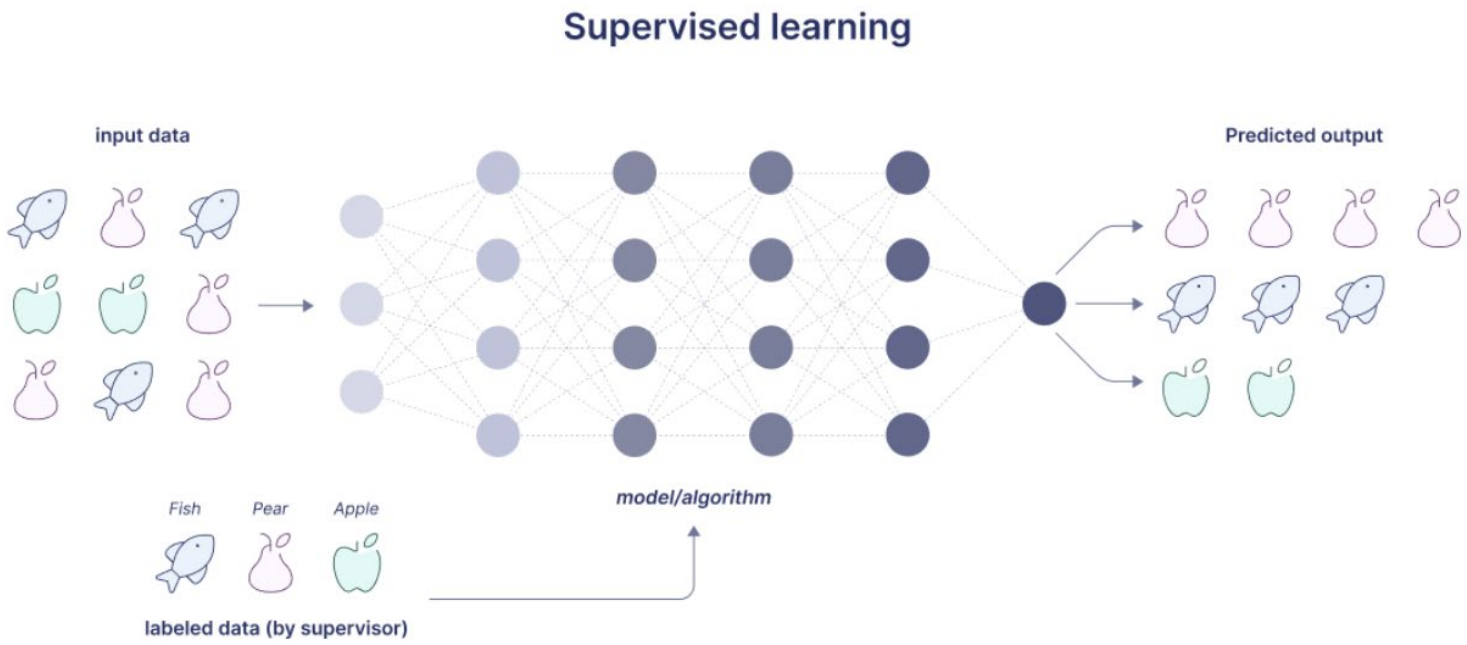
Types of Artificial Intelligence

bepec CT Programs



Types of Artificial Intelligence

bepec CT Programs



Gen AI Tools

bepec CT Programs

1. *ChatGPT*
2. *Dall.E2*
3. *Stable Diffusion*
4. *Bard*
5. *MidJourney*
6. *Claude*
7. *Synthesia*
8. *Github Copilot*
9. *Cohere Generate*
10. *Murf*

Applications of GenAI

bepec CT Programs

1. The University of Pennsylvania deployed a generative AI model to simulate the spread of COVID-19 and test different interventions
2. Generating high-quality medical images. Hospitals can employ generative AI tools to enhance the traditional AI's diagnostic abilities. This technology can convert poor-quality scans into high-resolution medical images with great details
3. A team of researchers experimented with Generative Adversarial Network (GAN) models to extract and enhance features in low-quality medical scans, transforming them into high-resolution images
4. This approach was tested on brain MRI scans, dermoscopy, retinal fundoscopy, and cardiac ultrasound images, displaying a superior accuracy rate in anomaly detection after image enhancement.
5. Google's AI-powered Med-Palm 2 was trained on the MedQA dataset and achieved an 85% accuracy rate while answering relevant medical questions

Applications of GenAI

bepec CT Programs

- Designing and generating new molecules with desired properties that researchers can later evaluate in lab settings
- Predicting properties of novel drug candidates and proteins
- Generating virtual compounds with high binding affinity to the target that can be tested in computer simulations to reduce costs
- Forecasting side effects of novel drugs by analyzing their molecular structure
- ProteinSGM, that can generate novel realistic proteins after studying imagery representations of existing protein structures.
- Extract data from patients' medical records and populate the corresponding health registries. Microsoft is planning to integrate generative AI into Epic's EHR.
- Transcribe and summarize patient consultations, fill this information into the corresponding EHR fields, and produce clinical documentation. Microsoft's Nuance integrated generative AI tech GPT-4 into its clinical transcription software.
- Generate structured health reports by analyzing patient information, such as medical history, lab results, scans, etc.

Applications of GenAI

bepec CT Programs

- Navina, a medical AI startup, built a generative AI assistant that helps doctors tackle administrative duties more efficiently. This tool can access patient data, including EHRs, insurance claims, and scanned documents, give status updates, recommend care options, and answer doctors' questions. It can even generate structured documents, such as referral letters and progress notes.
- A team of German researchers built an AI-powered model, GANerAid, to generate synthetic patient data for clinical trials. This model is based on the GAN approach and can produce medical data with the desired properties even if the original training dataset was limited in size.

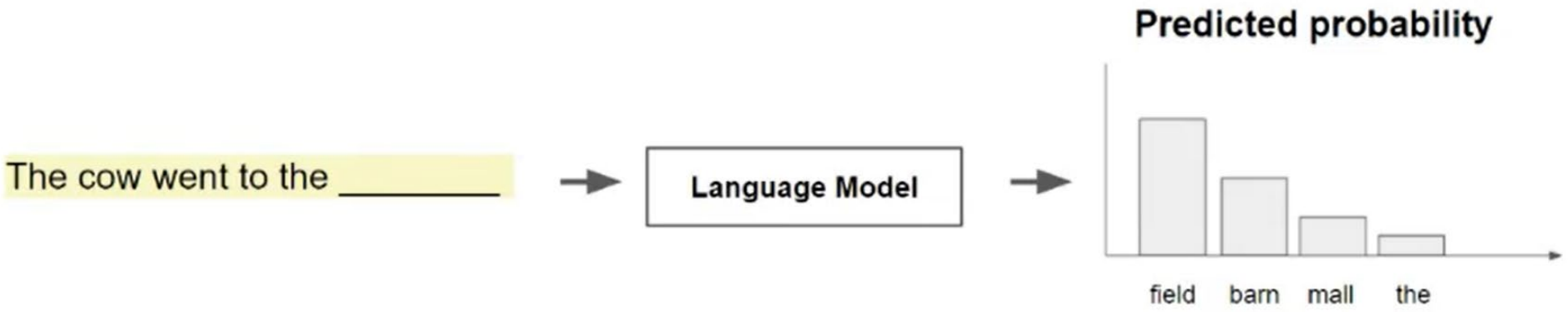
Different Language Models

bepec CT Programs

- N-Gram Language Models
- Hidden Markov Chains
- RNNs/LSTMS
- Transformer Language Models
  - BERT(Bidirectional Encoder Representations from Transformers)
  - GPT(Generative Pre-Trained Transformer)
  - T5(Text-to-Text Transfer Transformer)
  - RoBERTa
  - XLNet

Language Models

bepec CT Programs



Language Models

bepec CT Programs

- Language models are hard to train due to its Sequential Nature.
- But meaning of the word changes due to context in Language

Generative Models

bepec CT Programs

Hallucination

bepec CT Programs

Hallucination (artificial intelligence)

Article Talk

From Wikipedia, the free encyclopedia

In the field of artificial intelligence (AI), a **hallucination** or **artificial hallucination** (also called **confabulation**[1] or **delusion**[2]) is a response generated by an AI which contains false or misleading information presented as fact.[3][4][5]

For example, a hallucinating chatbot might, when asked to generate a financial report for a company, falsely state that the company's revenue was $13.6 billion (or some other number apparently "plucked from thin air").[6] Such phenomena are termed "hallucinations", in loose analogy with the phenomenon of hallucination in human psychology. However, one key difference is that human hallucination is usually associated with false *percepts*, but an AI hallucination is associated with the category of unjustified responses or beliefs.[5] Some

Hallucination

bepec CT Programs

Model	Estimated Size	Hallucination rate
GPT-4	Unknown (>= 175 Billion)	8.39%
GPT-3 Davinci-003	175 Billion	9.59%
ChatGPT-3.5 Turbo	Unknown	13.99%
Dolly	Small	Too poor to quantify
Alpaca	Small	High
Fine-Tuned* Alpaca (research-only)	Small	17.95%

Generative Models

bepec CT Programs

- Generative Models are great at creativity, but they lack Accuracy due to Hallucinations.

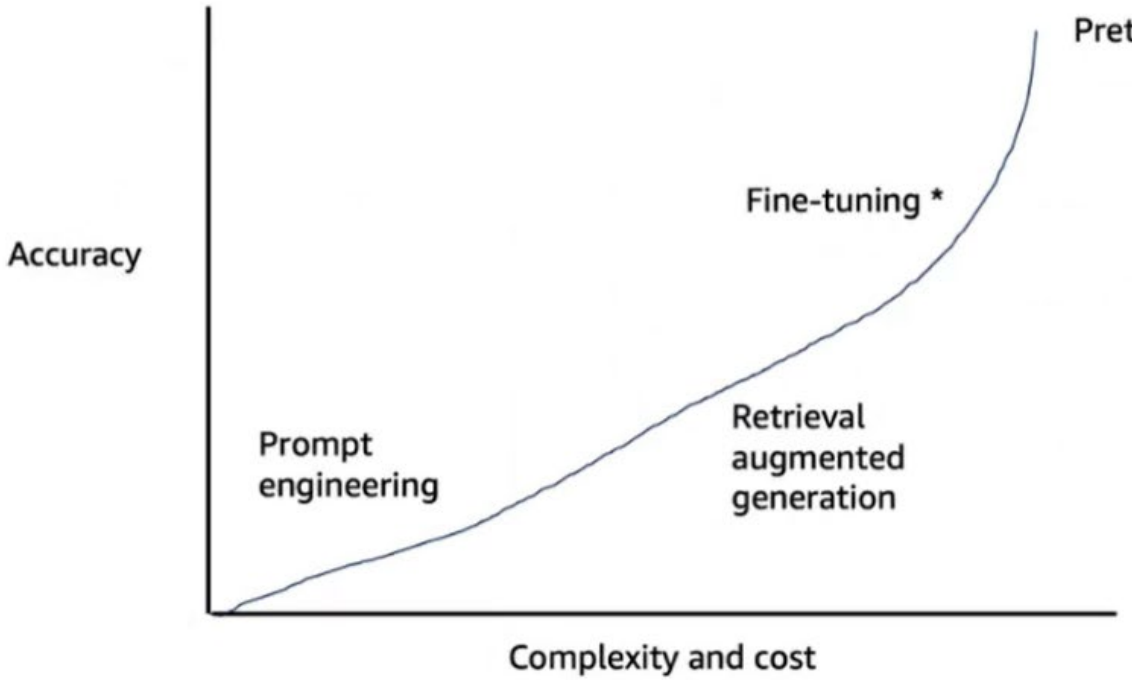
Model Parameters

bepec CT Programs

	Total Parameters*	Total Data Size (in tokens)
GPT-3	175B	300B
GPT-NeoX	20B	472B
OPT-66	66B	300B
BLOOM	176B	366B
BloombergGPT	50B	569B

Accuracy

bepec CT Programs



5 Principles of Prompt

bepec CT Programs

- Give Direction
- Specify Format
- Provide Examples
- Evaluate Quality
- Divide into Small Chain of Tasks

Retrieval Augmented Generation

bepec CT Programs

- Retrieval augmented generation or RAG is an architectural approach that can improve the efficacy of large language model (LLM) applications by leveraging custom data. This is done by retrieving relevant data/documents relevant to a question or task and providing them as context for the LLM. RAG has shown success in support chatbots and Q&A systems that need to maintain up-to-date information or access domain-specific knowledge

Retrieval Augmented Generation

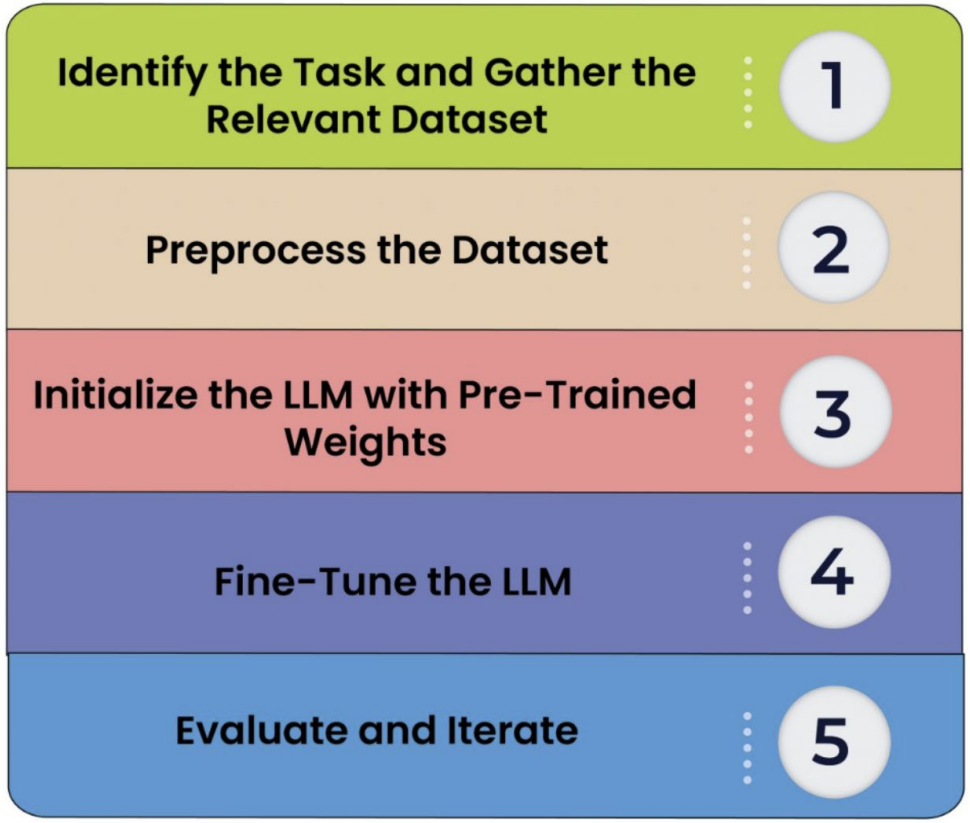
bepec CT Programs

- Retrieval augmented generation or RAG is an architectural approach that can improve the efficacy of large language model (LLM) applications by leveraging custom data. This is done by retrieving relevant data/documents relevant to a question or task and providing them as context for the LLM. RAG has shown success in support chatbots and Q&A systems that need to maintain up-to-date information or access domain-specific knowledge

Fine-Tuning

bepec CT Programs

- Adapting a pre-trained LLM to specific data sets or domains



Pre-Training

bepec CT Programs

- Training an LLM from scratch

Method	Definition	Primary use case	Data requirements	Advantages	Considerations
 **Prompt engineering**	Crafting specialized prompts to guide LLM behavior	Quick, on-the-fly model guidance	None	Fast, cost-effective, no training required	Less control than fine-tuning
 **Retrieval augmented generation (RAG)**	Combining an LLM with external knowledge retrieval	Dynamic data sets and external knowledge	External knowledge base or database (e.g., vector database)	Dynamically updated context, enhanced accuracy	Increases prompt length and inference computation

Pre-Training

bepec CT Programs

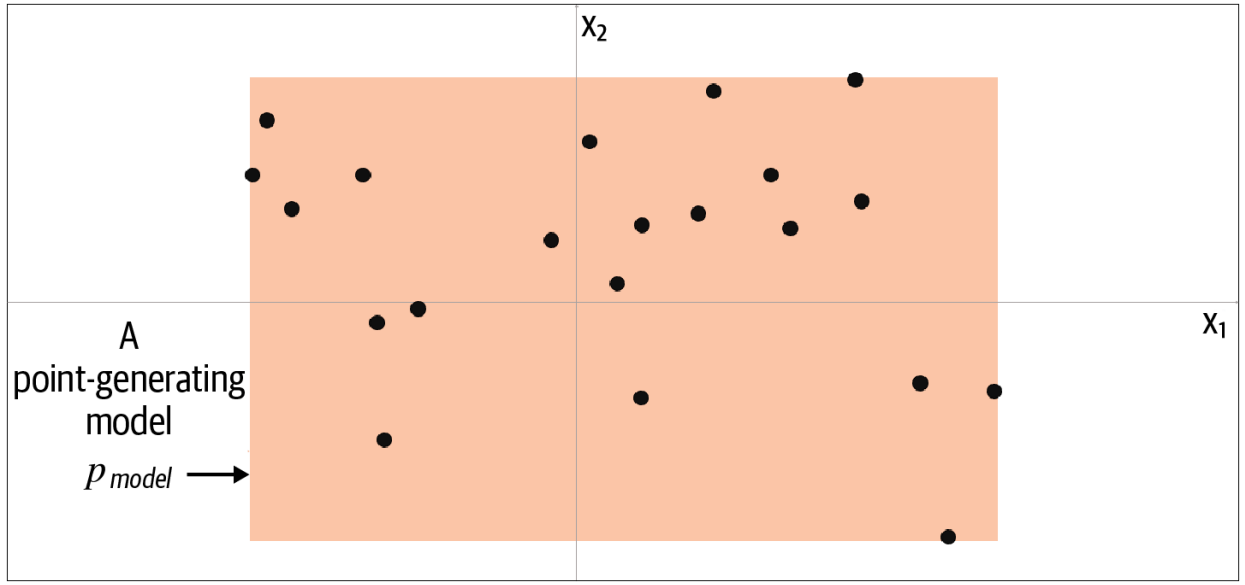
- Training an LLM from scratch

 Fine-tuning	Adapting a pre-trained LLM to specific data sets or domains	Domain or task specialization	Thousands of domain-specific or instruction examples	Granular control, high specialization	Requires labeled data, computational cost
 Pre-training	Training an LLM from scratch	Unique tasks or domain-specific corpora	Large data sets (billions to trillions of tokens)	Maximum control, tailored for specific needs	Extremely resource-intensive

Generative Model Framework

bepec CT Programs

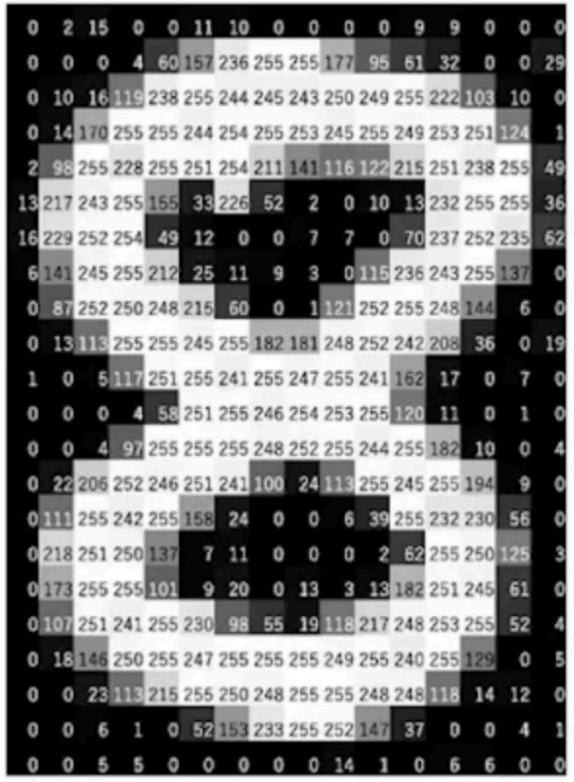
- We have a dataset of observations **X**.
- We assume that the observations have been generated according to some unknown distribution, **Px.**
- We want to build a generative model Pm that mimics **Px**. If we achieve this goal, we can sample from **Pm** to generate observations that appear to have been drawn from **Px.**



Representation Learning

bepec CT Programs

Kanth: Hey, where are you?
Person X: I'm in corner seat at with White hair, Sunglasses with white shirt, blue jeans and hat.
Kanth: Hey, I saw you!



Representation Learning

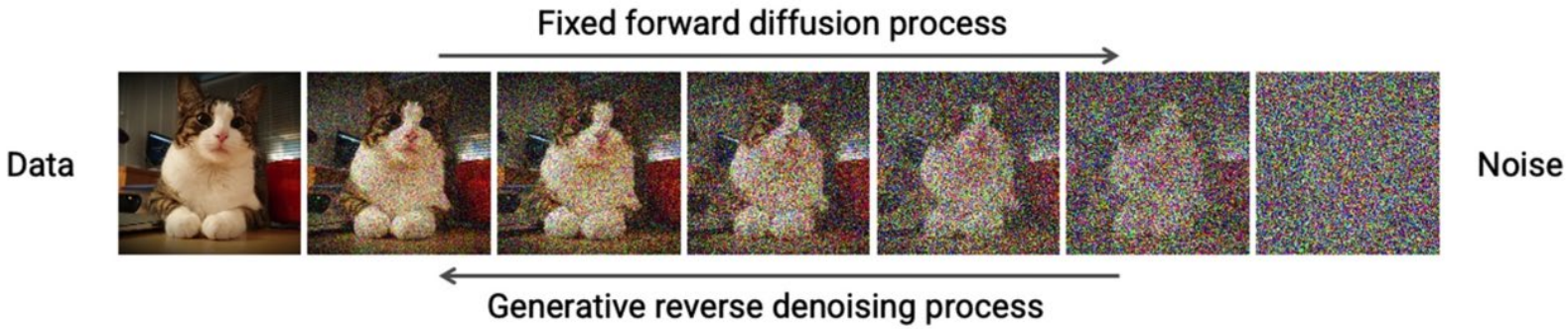
bepec CT Programs

- The core idea behind *representation learning*.
- Instead of trying to model the high-dimensional sample space directly, we describe each observation in the training set using some lower-dimensional *latent space.*
- Then learn a mapping function that can take a point in the latent space and map it to a point in the original domain.
- In other words, each point in the latent space is a *representation* of some high-dimensional observation.

Diffusion Models

bepec CT Programs

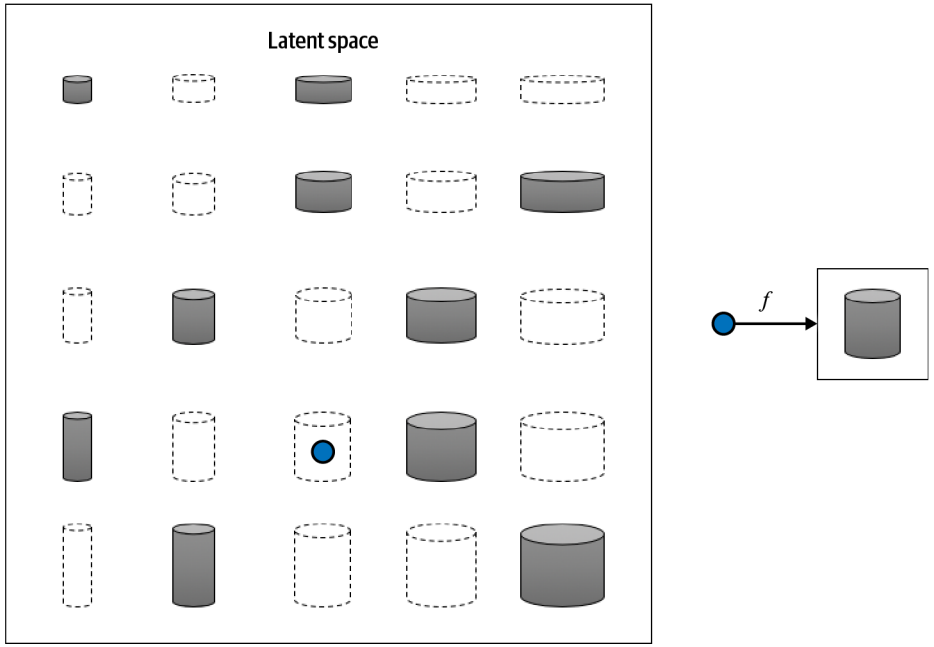
- The core idea behind a denoising diffusion model is simple
- we train a deep learning model to denoise an image over a series of very small steps.
- If we start from pure random noise, in theory we should be able to keep applying the model until we obtain an image that looks as if it were drawn from the training set.



Representation Learning

bepec CT Programs

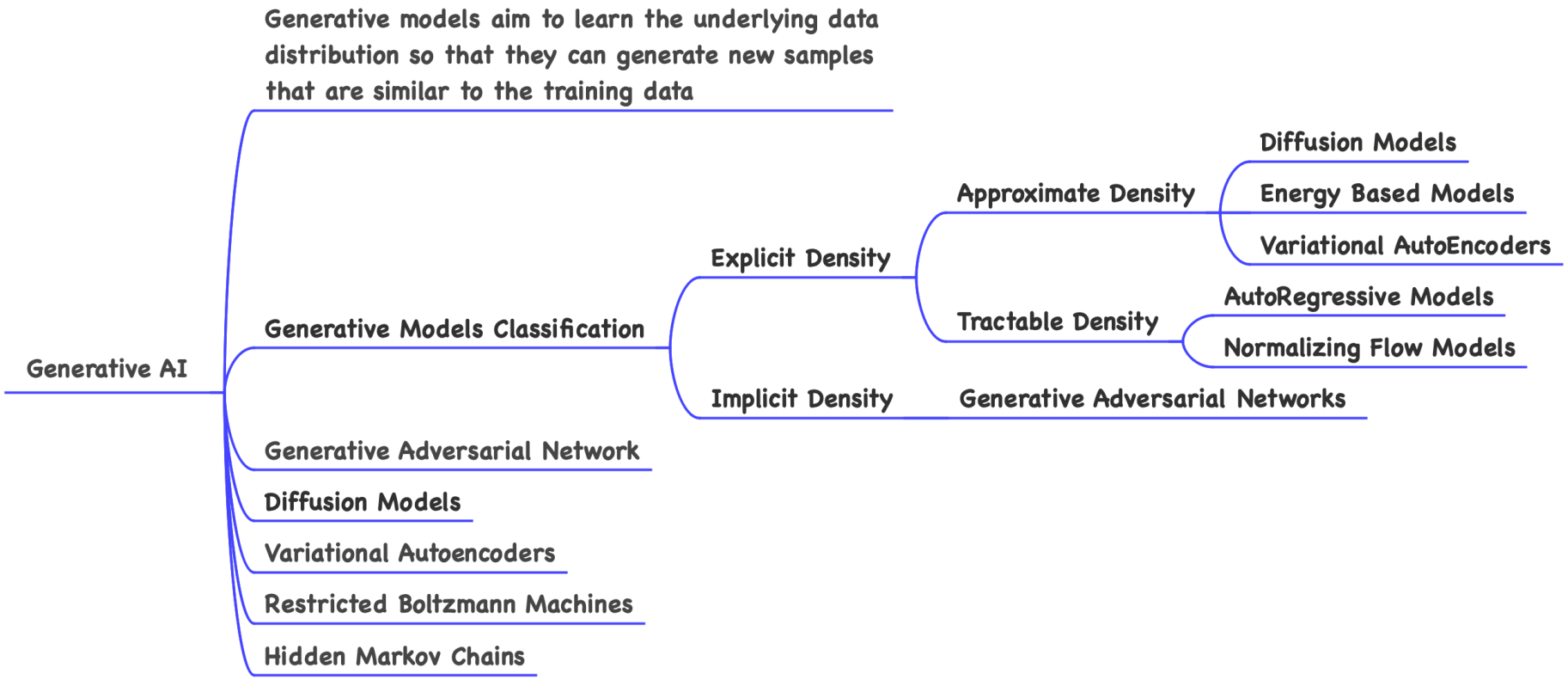
- The core idea behind *representation learning*.
- Instead of trying to model the high-dimensional sample space directly, we describe each observation in the training set using some lower-dimensional *latent space.*
- Then learn a mapping function that can take a point in the latent space and map it to a point in the original domain.
- In other words, each point in the latent space is a *representation* of some high-dimensional observation.



Different Generative Models

bepec CT Programs

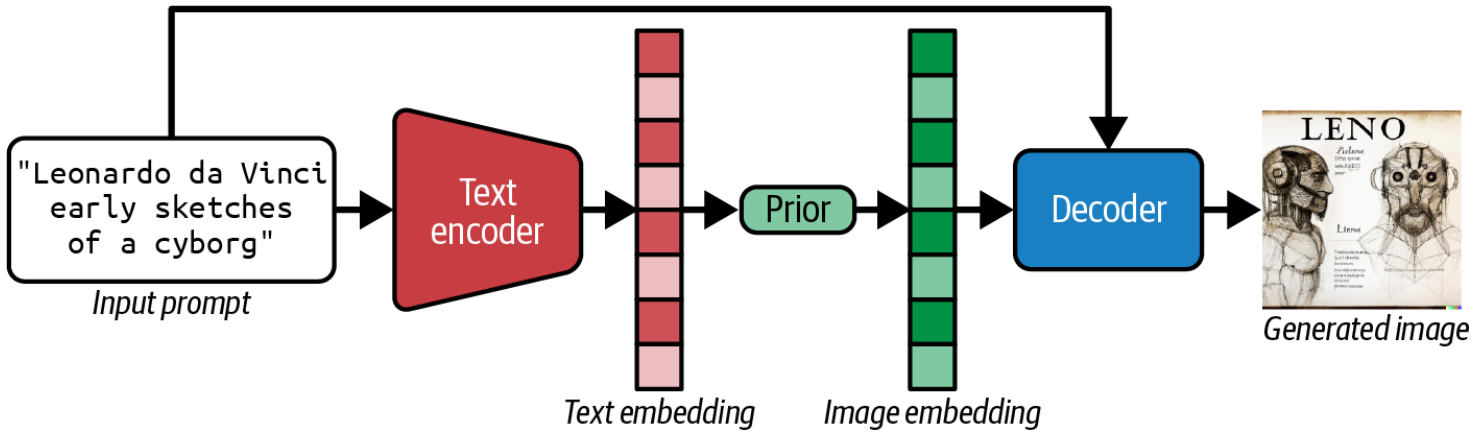
- Mapping Function is Changing



MultiModal Learning

bepec CT Programs

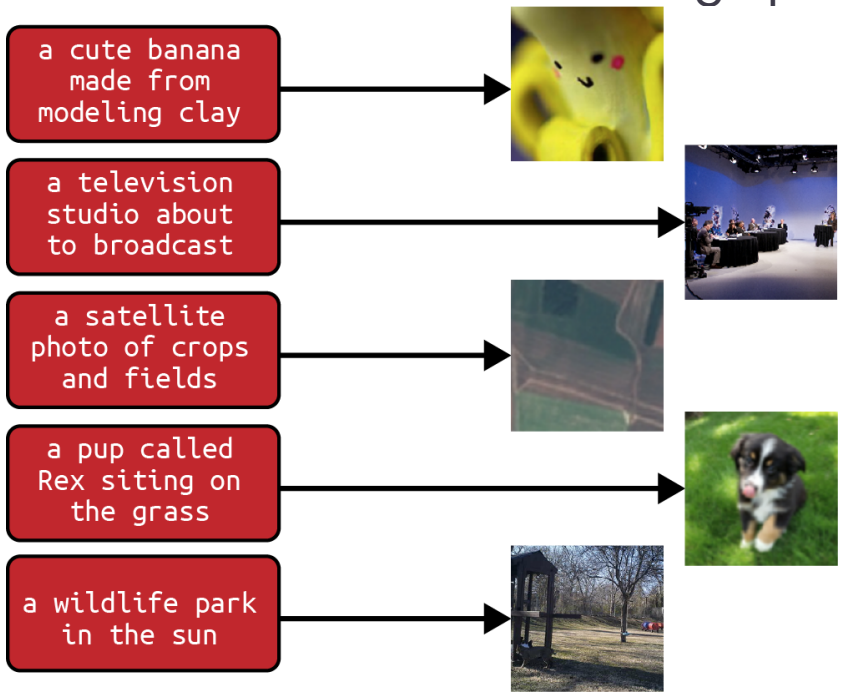
- *Multimodal learning involves training generative models to convert between two or more different kinds of data*
- *Text understanding and image generation are difficult to solve in their own right, as we have seen in previous chapters of this book. Multimodal modeling such as this presents an additional challenge, because the model must also learn how to cross the bridge between the two domains and learn a shared representation that allows it to accurately convert from a block of text to a high-fidelity image without loss of information.*



Contrastive Learning Image Pre-Training[CLIP]

bepec CT Programs

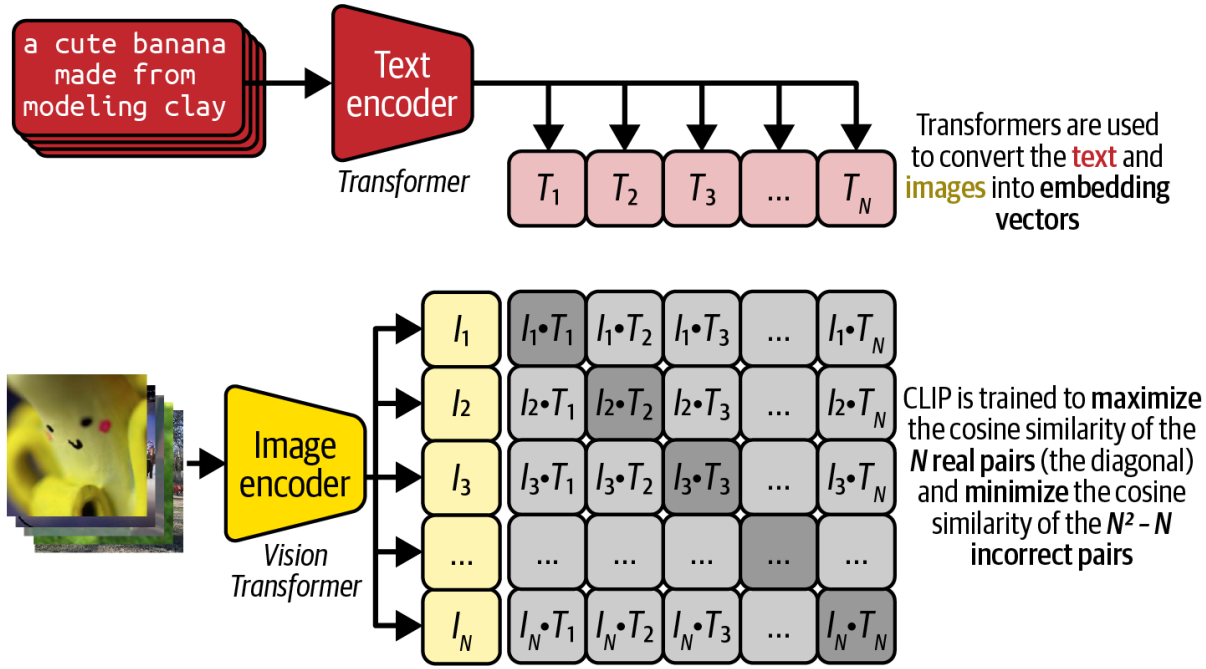
- CLIP was unveiled in a paper published by OpenAI in February 2021 (just a few days after the first DALL.E paper) that described it as “a neural network that efficiently learns visual concepts from natural language supervision.”
- It uses a technique called *contrastive learning* to match images with text descriptions. The model is trained on a dataset of 400 million text–image pairs scraped from the internet



Contrastive Learning Image Pre-Training[CLIP]

bepec CT Programs

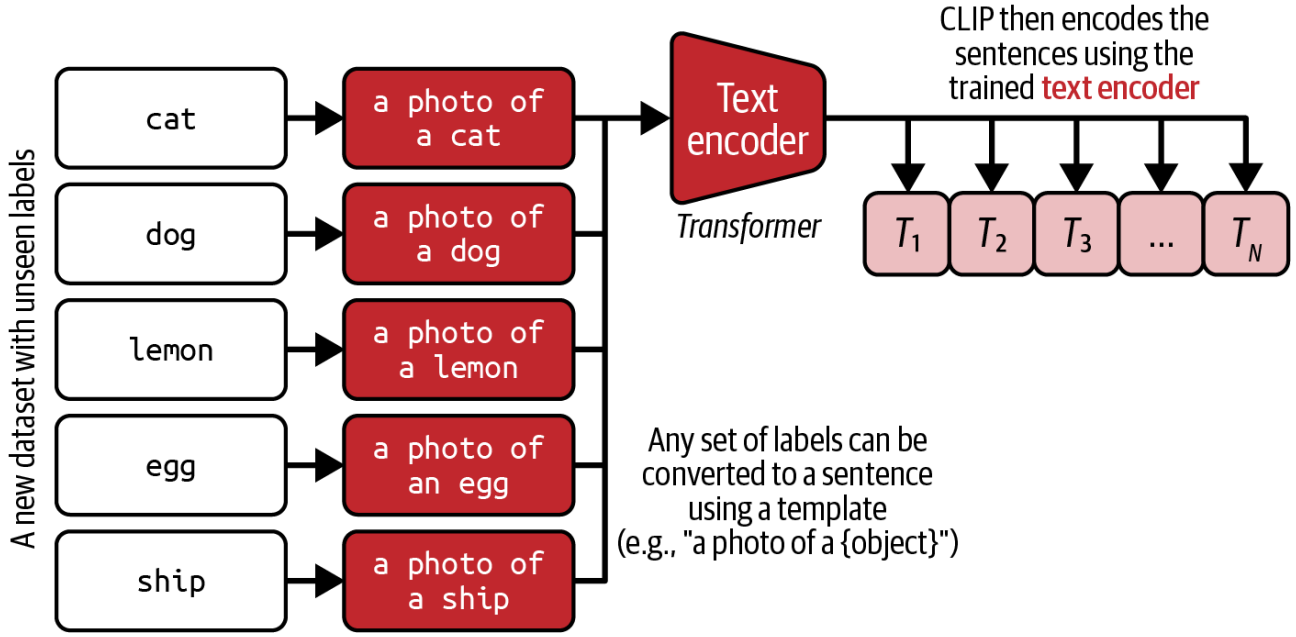
- The key idea behind contrastive learning is simple. We train two neural networks: a *text encoder* that converts text to a text embedding and an *image encoder* that converts an image to an image embedding.
- Then, given a batch of text–image pairs, we compare all text and image embedding combinations using *cosine similarity* and train the networks to maximize the score between matching text–image pairs and minimize the score between incorrect text–image pairs.



Contrastive Learning Image Pre-Training[CLIP]

bepec CT Programs

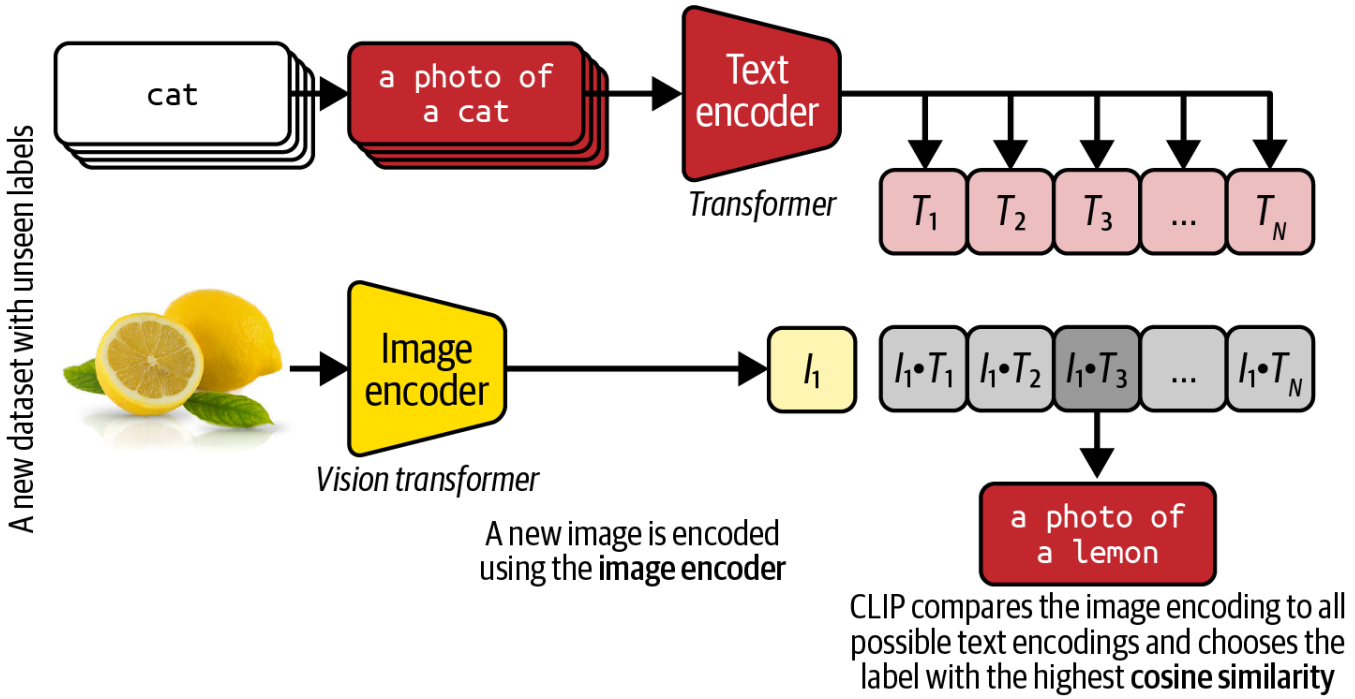
- Both the text encoder and the image encoder are Transformers—the image encoder is a Vision Transformer (ViT), introduced in "ViT VQ-GAN", which applies the same concept of attention to images.
- For example, suppose we want to use CLIP to predict the label of a given image in the ImageNet dataset. We can first convert the ImageNet labels into sentences by using a template (e.g., "a photo of a <label>")



Contrastive Learning Image Pre-Training[CLIP]

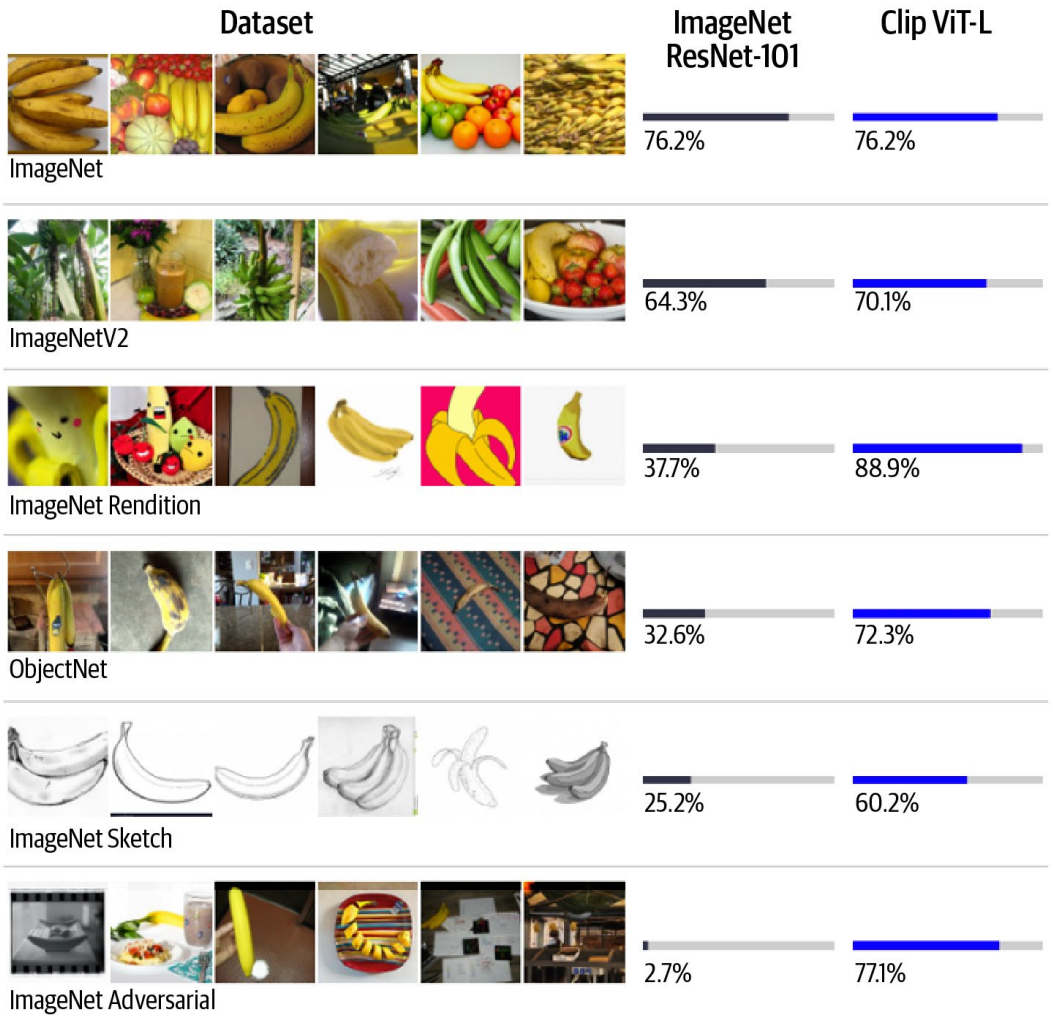
bepec CT Programs

- To predict the label of a given image, we can pass it through the CLIP image encoder and calculate the cosine similarity between the image embedding and all possible text embeddings in order to find the label with the maximum score



Contrastive Learning Image Pre-Training[CLIP]

bepec CT Programs



Contrastive Learning Image Pre-Training[CLIP]

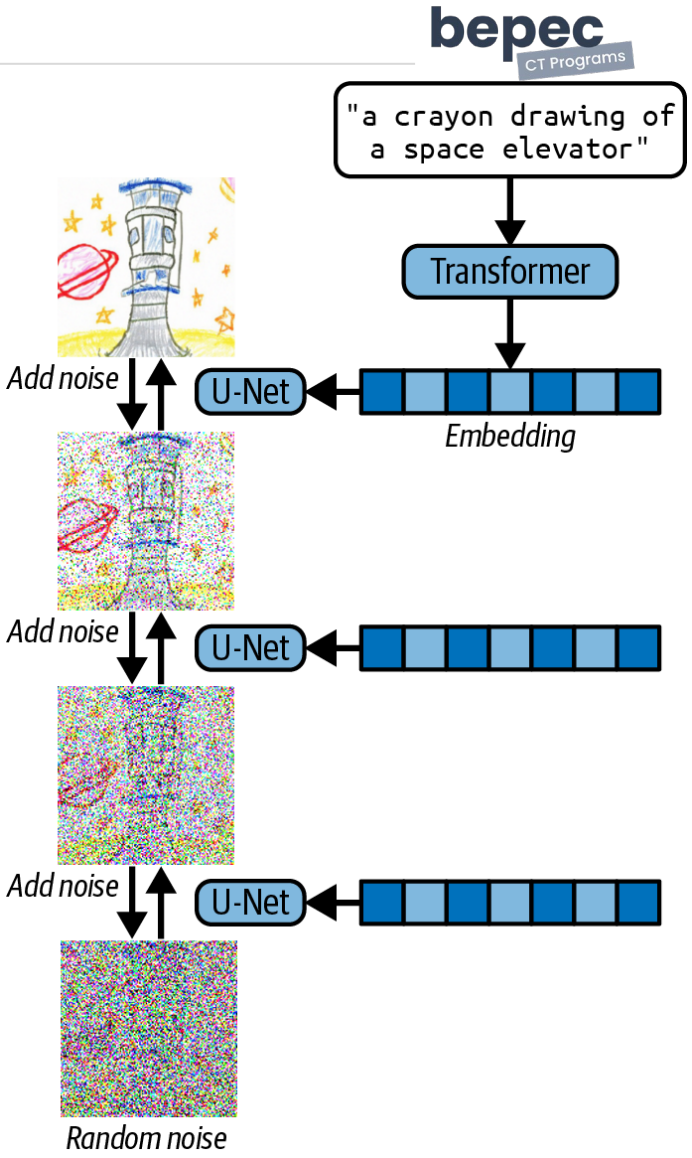
bepec CT Programs

The DALL.E 2 authors tried two different methods for training the prior model:

- An autoregressive model
- A diffusion model

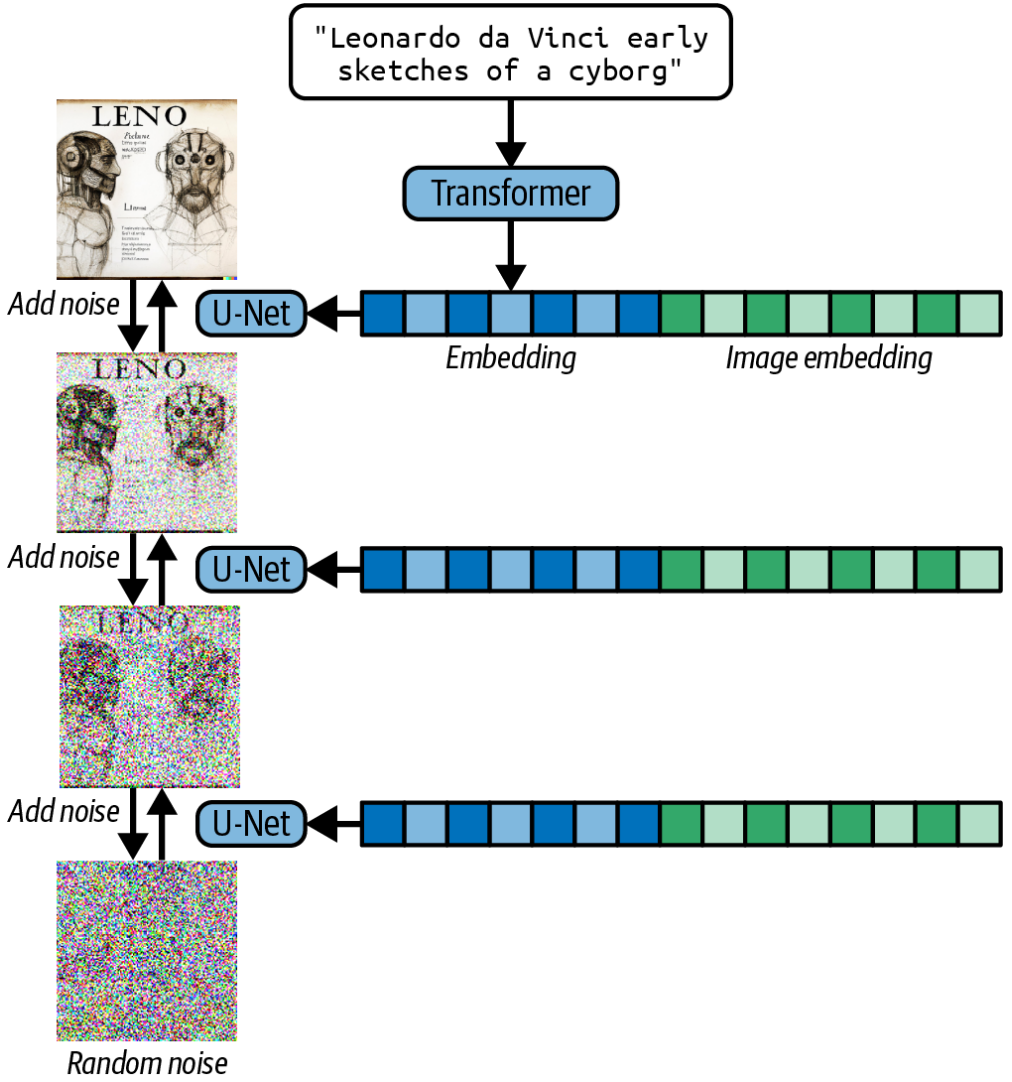
Guided Language to Image Diffusion for Generation and Editing

GLIDE is able to generate realistic images from text prompts, in much the same way that DALL.E 2 can. The difference is that GLIDE does not make use of CLIP embeddings, but instead works directly with the raw text prompt, training the entire model from scratch



DALL.E2 Architecture

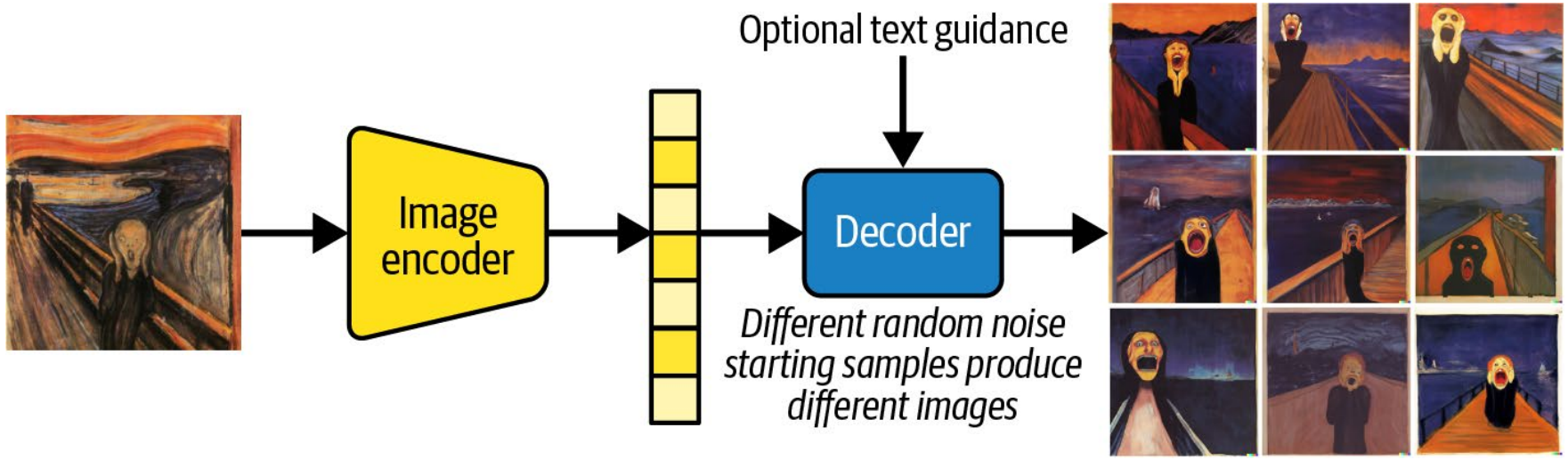
bepec CT Programs



Different Images

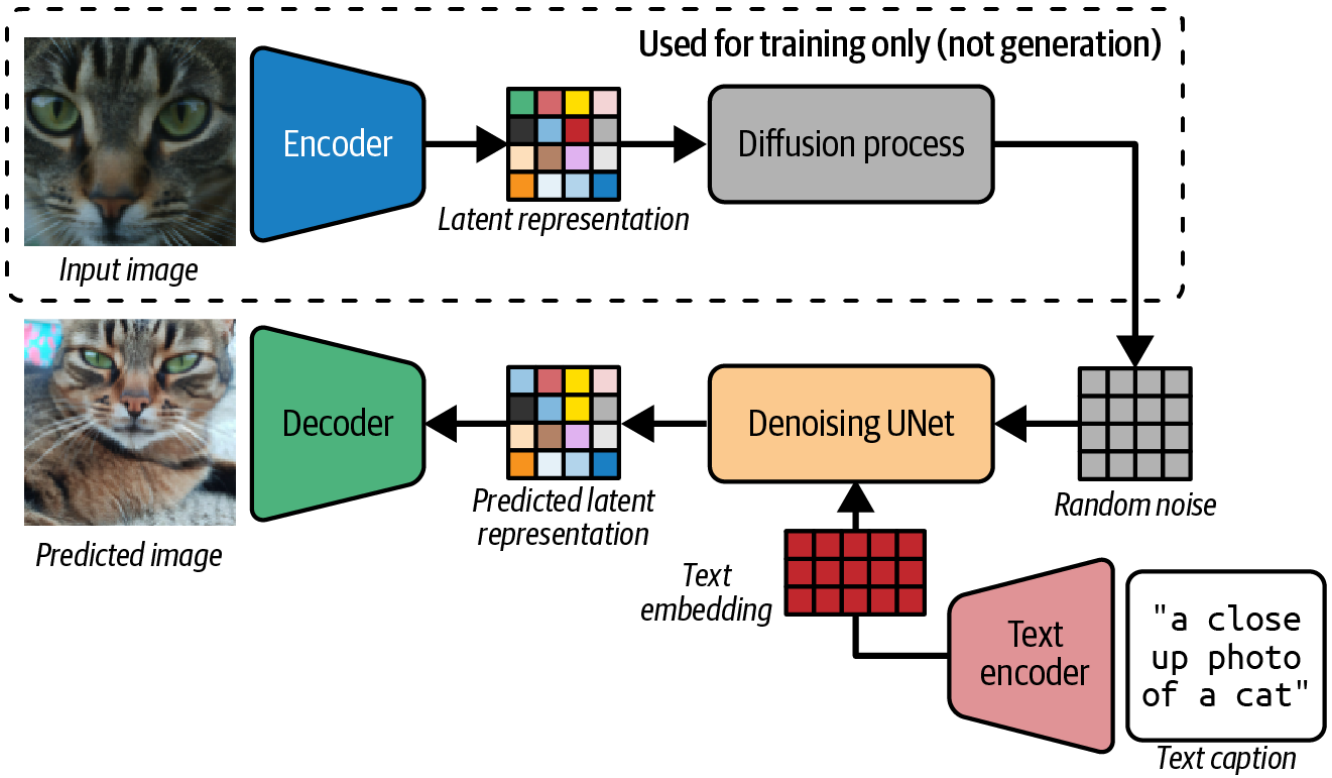
bepec CT Programs

Generate images using the DALL.E 2 decoder we sample an image consisting of pure random noise and then gradually reduce the amount of noise using the denoising diffusion model, conditioned on the provided image embedding. Selecting different initial random noise samples will result in different images.



Stable Diffusion

The main architectural difference between Stable Diffusion and the text-to-image models discussed previously is that it uses *latent diffusion* as its underlying generative model.



Machine-Generated Event Monitoring

bepec CT Programs

- Predictive maintenance is the best-practice strategy that identifies and rectifies possible equipment failures before they happen. According to Deloitte, it increases productivity by 25%, reduces breakdowns by 70%, and lowers maintenance costs by 25%.
- It helps manufacturers optimize operations by interpreting telemetry from equipment and machines to reduce unplanned downtime, gain operating efficiencies, and maximize utilization.
- If a problem is identified, gen AI can also recommend potential solutions and a service plan to help maintenance teams rectify the issue.
- Manufacturing engineers can interact with this technology using natural language and common inquiries, making it accessible to the current workforce and attractive to new employees.
- Video: Link

Customer-Service Automation

bepec CT Programs

- According to Salesforce, 80% of business buyers expect companies to respond and interact with them in real time, and 82% say personalized care influences their loyalty.
- Gen AI — which provides a helpful, value-added customer service experience that automates and accelerates time-to-resolution for common interactions like product troubleshooting, ordering replacement parts, scheduling service, product information, and product operation.
- Video- Link

US Steel

bepec CT Programs

- The first gen AI-driven application that U. S. Steel will launch is called MineMind™ which aims to simplify equipment maintenance by providing optimal solutions for mechanical problems, saving time and money, and ultimately improving productivity.
- When fully operational, MineMind™ will allow a U. S. Steel technician to reduce the amount of time to complete a work order by an estimated 20%.
- For example, the application will assist maintenance crews by guiding them through truck repairs, ordering parts and distilling complex information. The application also provides comprehensive references, providing detailed, verified source information to ensure accuracy.

GE Appliances

bepec CT Programs

- Flavorly AI uses generative AI to create a selection of unique recipes for consumers based on their food preferences and the ingredients they have available in their kitchen.
- Take for example, a familiar experience in homes across America: It's 6:30 p.m.; there is no dinner plan; and a selection of seemingly random ingredients are in the refrigerator. Consumers can simply select a recipe category and type of cuisine, then input available ingredients and include any dietary preferences. They will receive tailored recipes with ingredients, instructions, and even photos.

Nuance

- Generative AI Model: Link
- DaX Express: Link

Medical Imaging

bepec CT Programs

- Medical Imaging: Link

E-Commerce

bepec CT Programs

- Let’s imagine a company that wants to use a generative AI model to create product descriptions for its e-commerce site. Initially, they’d prepare the input data by collecting information on the products they sell and organizing it appropriately for the AI model's training.
- After preparing the input data, the company trains the AI model based on this data. They might use a pretrained model, such as Open AI's Davinci, and fine-tune it to produce product descriptions specific to their company.
- In this context, prompt engineering would involve developing prompts that provide information about the product’s features, benefits, and unique selling points. These prompts should be crafted in a way that allows the AI model to understand the context and purpose of the product description and generate a relevant and accurate output.
- After fine-tuning the model, the company uses it to create product descriptions based on the input text. They then deploy the AI model and make it available/accessible through an API, allowing the generated product descriptions to be integrated with their e-commerce website. To ensure that the generated product descriptions remain accurate and relevant over time, the company needs to monitor the AI model’s performance and make necessary updates and improvements, especially when new products come along.

E-Commerce

bepec
CT Programs





E-Commerce

bepec CT Programs

- **ASOS’s personalized styling service**, which leverages generative AI to create a virtual catwalk that provides customers with personalized styling offers. The AI analysis a customer's body shape and suggests outfits accordingly, generating a new outfit combination each time the user refreshes the page.

Responsible AI

bepec CT Programs

- Responsible AI" refers to the development and use of artificial intelligence in a manner that is ethical, transparent, accountable, and consistent with human values. It encompasses various principles and practices to ensure that AI systems are designed and deployed in a way that benefits society while minimizing potential harms
- **Ethical Design and Use**: Ensuring that AI systems are designed with ethical considerations in mind, such as respecting human rights and dignity.
- **Transparency**: Making the workings of AI systems understandable to users and other stakeholders. This involves explaining how AI makes decisions, which can be challenging with complex algorithms
- **Safety and Reliability**: Ensuring that AI systems are safe and function reliably under various conditions, which is critical especially in high-stakes domains like healthcare or transportation.
- **Privacy Protection**: Respecting and protecting the privacy of individuals whose data is used in AI systems. This includes secure data practices and compliance with data protection regulations.

Tools

bepec CT Programs

- **LangChain** is a framework for developing applications powered by language models. It enables applications that:
- **Are context-aware**: connect a language model to sources of context (prompt instructions, few shot examples, content to ground its response in, etc.)
- **Reason**: rely on a language model to reason (about how to answer based on provided context, what actions to take, etc.)
- This framework consists of several parts.
- **LangChain Libraries**: The Python and JavaScript libraries. Contains interfaces and integrations for a myriad of components, a basic run time for combining these components into chains and agents, and off-the-shelf implementations of chains and agents.
- **LangChain Templates**: A collection of easily deployable reference architectures for a wide variety of tasks.
- **LangServe**: A library for deploying LangChain chains as a REST API.
- **LangSmith**: A developer platform that lets you debug, test, evaluate, and monitor chains built on any LLM framework and seamlessly integrates with LangChain.

Tools

bepec CT Programs

- **Hugging Face** is a widely recognized company in the AI research and development community. They are known for their open-source libraries like Transformers, which provide thousands of pre-trained models for tasks like text classification, information extraction, question answering, and more. Hugging Face's platform and tools are extensively used for natural language processing (NLP) research and production applications, offering an easy-to-use interface for implementing and deploying AI models.
- **DeepSpeed** is an open-source deep learning optimization library developed by Microsoft. It's designed to improve the efficiency, scale, and speed of training large-scale deep learning models. DeepSpeed introduces several novel technologies that significantly reduce memory usage and improve the training speed of large models

Tools

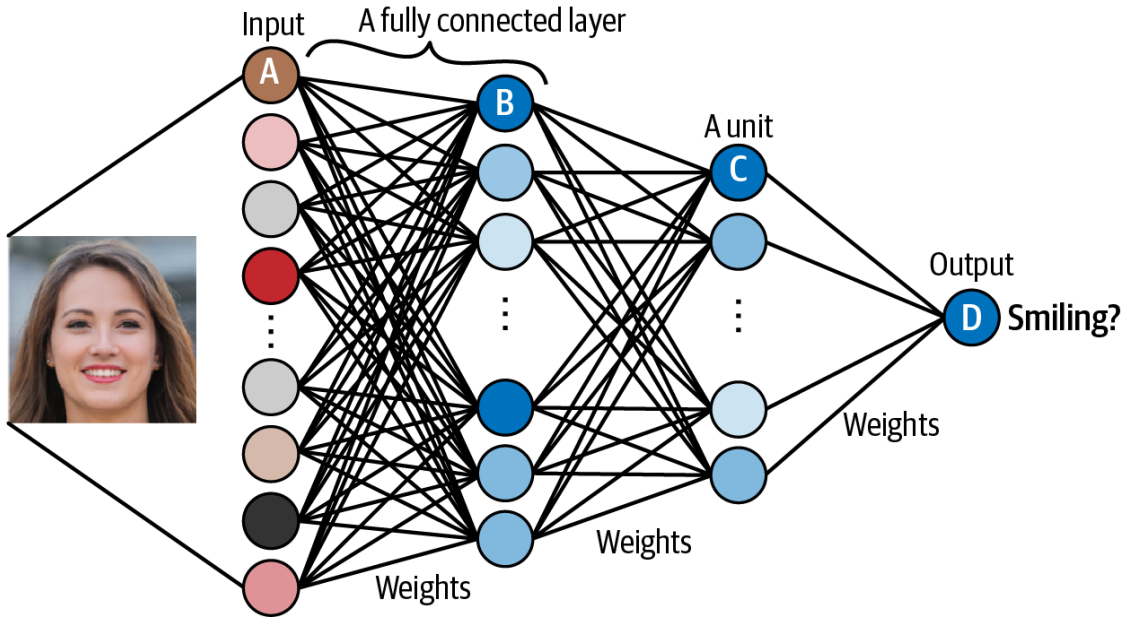
bepec CT Programs

- Azure OpenAI Services
- Amazon BedRock
- Amazon SageMaker
- Google Cloud Vertex AI

Neural Networks

bepec CT Programs

A neural network consists of a series of stacked *layers*. Each layer contains *units*that are connected to the previous layer's units through a set of *weights*. As we shall see, there are many different types of layers, but one of the most common is the *fully connected* (or *dense*) layer that connects all units in the layer directly to every unit in the previous layer.
Neural networks where all adjacent layers are fully connected are called *multilayer perceptrons* (MLPs).



Batch Normalization

bepec CT Programs

- Input is scaled, it's natural to expect the activations from all future layers to be relatively well scaled as well. Initially this may be true, but as the network trains and the weights move further away from their random initial values, this assumption can start to break down. This phenomenon is known as ***covariate shift*.**
- One common problem when training a deep neural network is ensuring that the weights of the network remain within a reasonable range of values—if they start to become too large, this is a sign that your network is suffering from what is known as the *exploding gradient* problem
- As errors are propagated backward through the network, the calculation of the gradient in the earlier layers can sometimes grow exponentially large, causing wild fluctuations in the weight values.
- If your loss function starts to return NAN, , chances are that your weights have grown large enough to cause an overflow error.

Batch Normalization

bepec CT Programs

- Batch Normalization standardizes the inputs to a layer for each mini-batch. This means that for each feature, the layer will have inputs that are zero mean and unit variance, effectively stabilizing the learning process

Steps:

- **Calculate Mean and Variance**: For each feature in the mini-batch, calculate the mean and variance.
- **Normalize**: Subtract the mean and divide by the square root of the variance plus a small constant (for numerical stability).
- **Scale and Shift**: Finally, multiply by a scale parameter (γ) and add a shift parameter (β). These parameters are learned during training and allow the network to undo the normalization if it determines that it's not beneficial.

Batch Normalization

bepec CT Programs

1.Improves Gradient Flow: By normalizing the inputs, gradients have a more predictable shape and scale, leading to better and faster convergence during training.
2.Allows Higher Learning Rates: Since the network is more stable, it's often possible to use higher learning rates, further speeding up training.
3.Reduces Overfitting: Batch Normalization has a slight regularizing effect, as the mini-batch statistics add some noise to the activations. This often reduces the need for other regularization techniques like Dropout.
4.Makes Weights Initialization Easier: Because the inputs to each layer are normalized, the network is less sensitive to the initial starting weights.
5.Applicable to Many Architectures: Batch Normalization has been successfully applied to fully connected networks, convolutional networks, and recurrent networks.

Convolution Transpose Layer

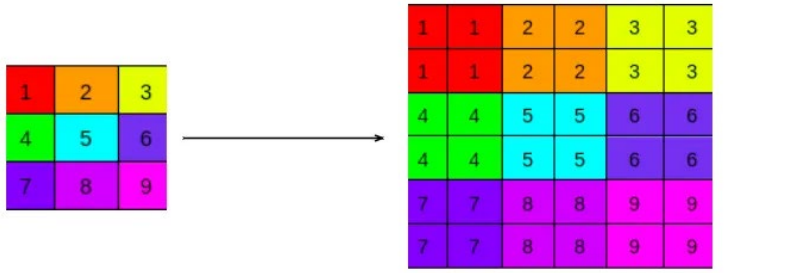
bepec CT Programs

- The convolutional transpose layer uses the same principle as a standard convolutional layer (passing a filter across the image), but is different in that setting strides = 2, *doubles* the size of the input tensor in both dimensions.
- The Convolution Transpose layer allows us to perform convolutional transpose operations on tensors. By stacking these layers, we can gradually expand the size of each layer, using strides of 2, until we get back to the original image dimension
- **Procedure:**
- **Padding the Input**: Instead of padding zeros around the input as in regular convolution, in the transpose version, zeros are inserted between input units. This is sometimes called "up sampling" the input.
- **Kernel Flipping**: The kernel is rotated 180 degrees (flipped horizontally and vertically). This is a property of the convolution operation itself.
- **Sliding the Kernel**: The flipped kernel is then slid over the padded input, similar to the regular convolution, but since the input has been upsampled, the output is larger.

UpSampling 2D

bepec CT Programs

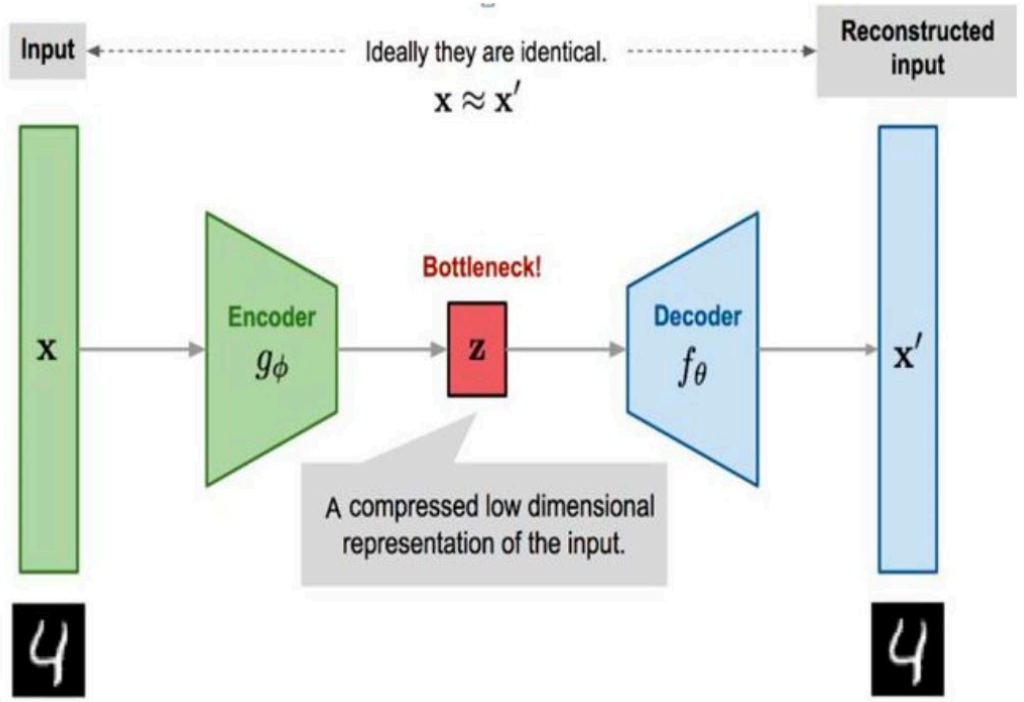
- The Upsampling 2D, layer simply repeats each row and column of its input in order to double the size. The Conv2D layer with stride 1 then performs the convolution operation. It is a similar idea to convolutional transpose, but instead of filling the gaps between pixels with zeros, upsampling just repeats the existing pixel values.
- It has been shown that the Conv2d Transpose method can lead to *artifacts*, or small checkerboard patterns in the output image.



Autoencoders

bepec CT Programs

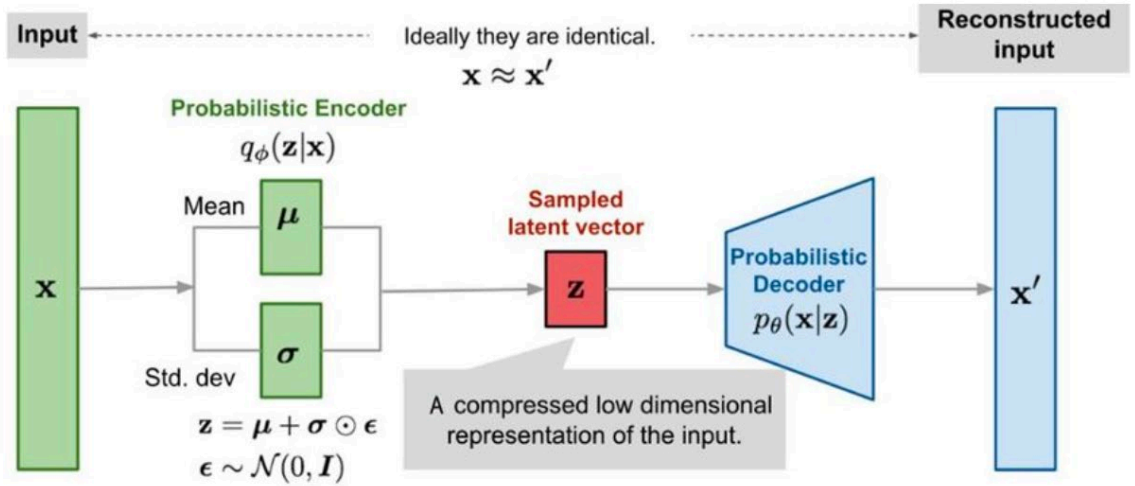
- Encoder network: Converts the high-dimensional input into the low-dimensional code wherein the
- input size is larger than the output
- Decoder network: Recovers the data from the code to larger output layers
- Decoder is symmetric to the encoder in terms of layer structure.
- Number of nodes per layer decreases with each subsequent layer of the encoder and increases with
- each subsequent layer in the decoder.



Variational Autoencoders

bepec CT Programs

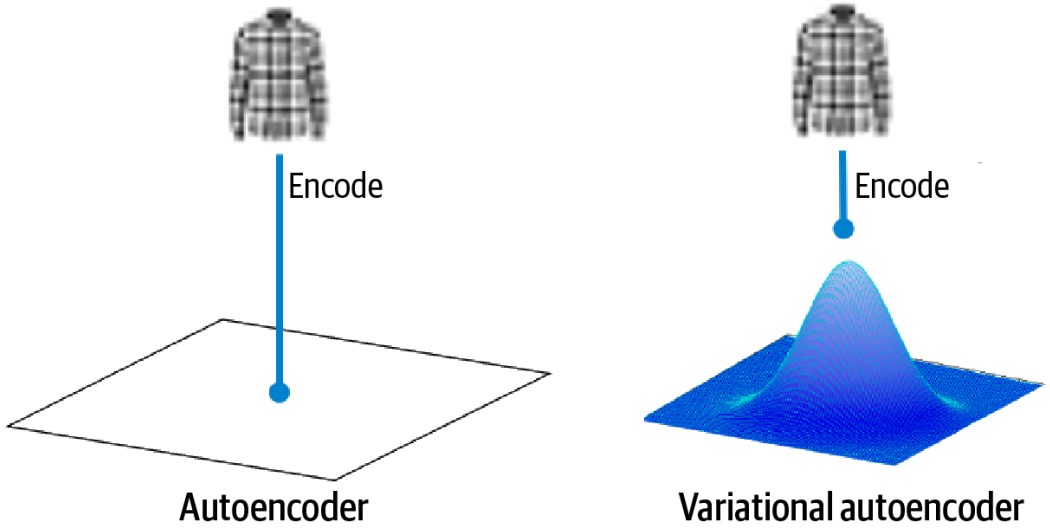
- Encoder network: Converts the high-dimensional input into the low-dimensional code wherein the
- input size is larger than the output
- Decoder network: Recovers the data from the code to larger output layers
- Decoder is symmetric to the encoder in terms of layer structure.
- Number of nodes per layer decreases with each subsequent layer of the encoder and increases with
- each subsequent layer in the decoder.



Variational Autoencoders

bepec CT Programs

- Encoder network: Converts the high-dimensional input into the low-dimensional code wherein the
- input size is larger than the output
- Decoder network: Recovers the data from the code to larger output layers
- Decoder is symmetric to the encoder in terms of layer structure.
- Number of nodes per layer decreases with each subsequent layer of the encoder and increases with
- each subsequent layer in the decoder.



Variational Autoencoders

bepec CT Programs

- The reparameterization trick is a technique used in variational inference to enable more efficient and stable optimization, particularly when training variational autoencoders (VAEs) and other models involving stochastic variables. It's a method for transforming random variables in a way that allows the gradient of an objective function to be more easily computed.
- In variational inference, you often deal with expectations under distributions that involve random variables. Directly optimizing these can be difficult due to the randomness.
- The reparameterization trick involves expressing the random variable in a way that separates the deterministic and stochastic parts. This allows you to move the randomness to a parameter-free noise variable, making it easier to take derivatives and optimize.

Variational Autoencoders

bepec CT Programs

For instance, in a Gaussian VAE, you might have:

$q(z|x)=\mathrm{N}(z;\mu(x),\sigma 2(x))$

Using the reparameterization trick, you can express:

$z=\mu(x)+\sigma(x)\cdot\epsilon$

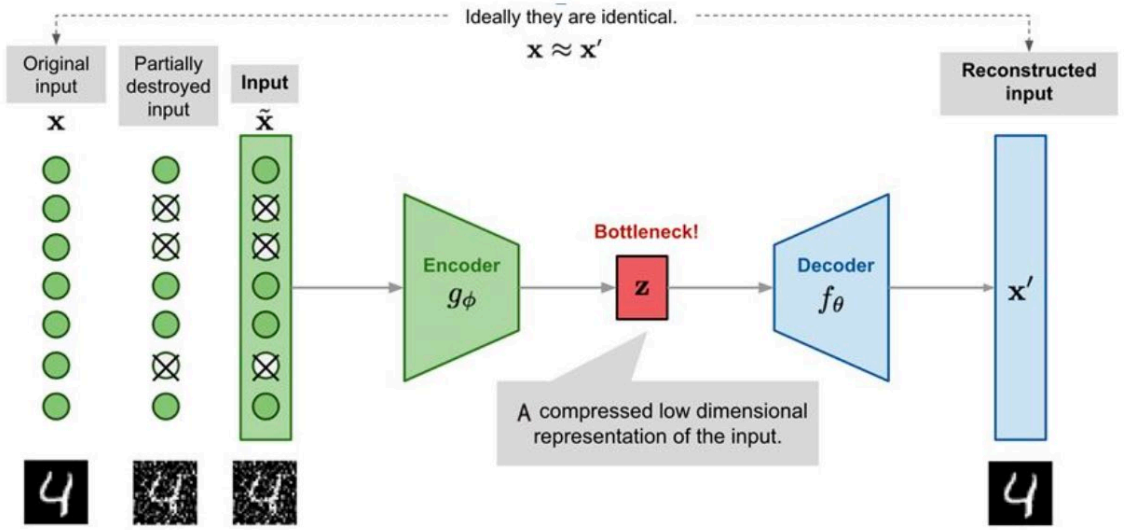
Where $\epsilon\sim\mathrm{N}(0,1)$. This allows you to compute gradients with respect to μ and σ despite the randomness in z.

- z is the latent variable you originally wanted to sample.
- ϵ is a noise term sampled from a simple distribution (like $\mathrm{N}(0,1)$).

DeNoising Autoencoders

bepec CT Programs

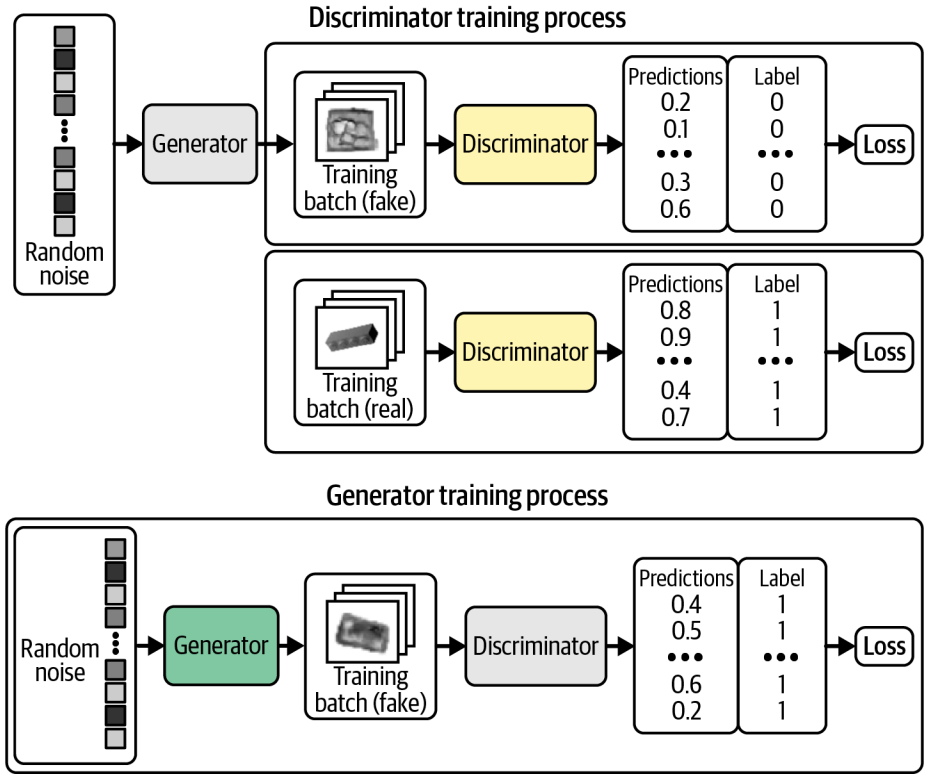
- Encoder network: Converts the high-dimensional input into the low-dimensional code wherein the
- input size is larger than the output
- Decoder network: Recovers the data from the code to larger output layers
- Decoder is symmetric to the encoder in terms of layer structure.
- Number of nodes per layer decreases with each subsequent layer of the encoder and increases with
- each subsequent layer in the decoder.



GANS Architecture

bepec CT Programs

- A GAN is a deep neural network architecture that consists of two networks, a generator network and
- a discriminator network.
- Through multiple cycles of generation and discrimination, both networks train each other, while
- simultaneously trying to defeat each other



Diffusion Models

bepec CT Programs

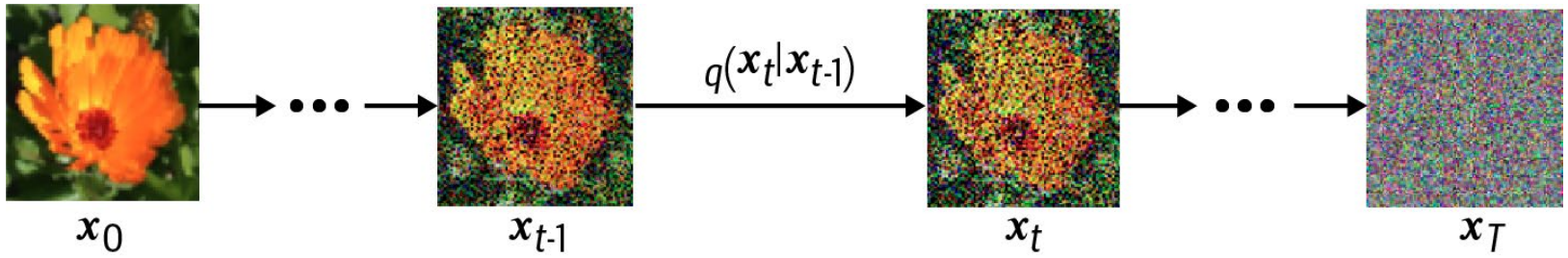
- The core idea behind a denoising diffusion model is simple—we train a deep learning model to denoise an image over a series of very small steps. If we start from pure random noise, in theory we should be able to keep applying the model until we obtain an image that looks as if it were drawn from the training set.

Diffusion Models

bepec CT Programs

Suppose we have an image $\mathbf{x}0$ that we want to corrupt gradually over a large number of steps (say, T=1,000), so that eventually it is indistinguishable from standard Gaussian noise (i.e., $\mathbf{x}T$ should have zero mean and unit variance).

We can define a function q that adds a small amount of Gaussian noise with variance βt to an image Xt-1 to generate a new image $\mathbf{x}t$. If we keep applying this function, we will generate a sequence of progressively noisier images $(\mathbf{x}0,\ldots,\mathbf{x}t)$



Diffusion Models

bepec CT Programs

Forward noising process Equation:

$$q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right)$$

The forward diffusion process equation after Reparameterization Trick

$$q\left(\mathbf{x}_t|\mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I}\right)$$

Diffusion Schedule

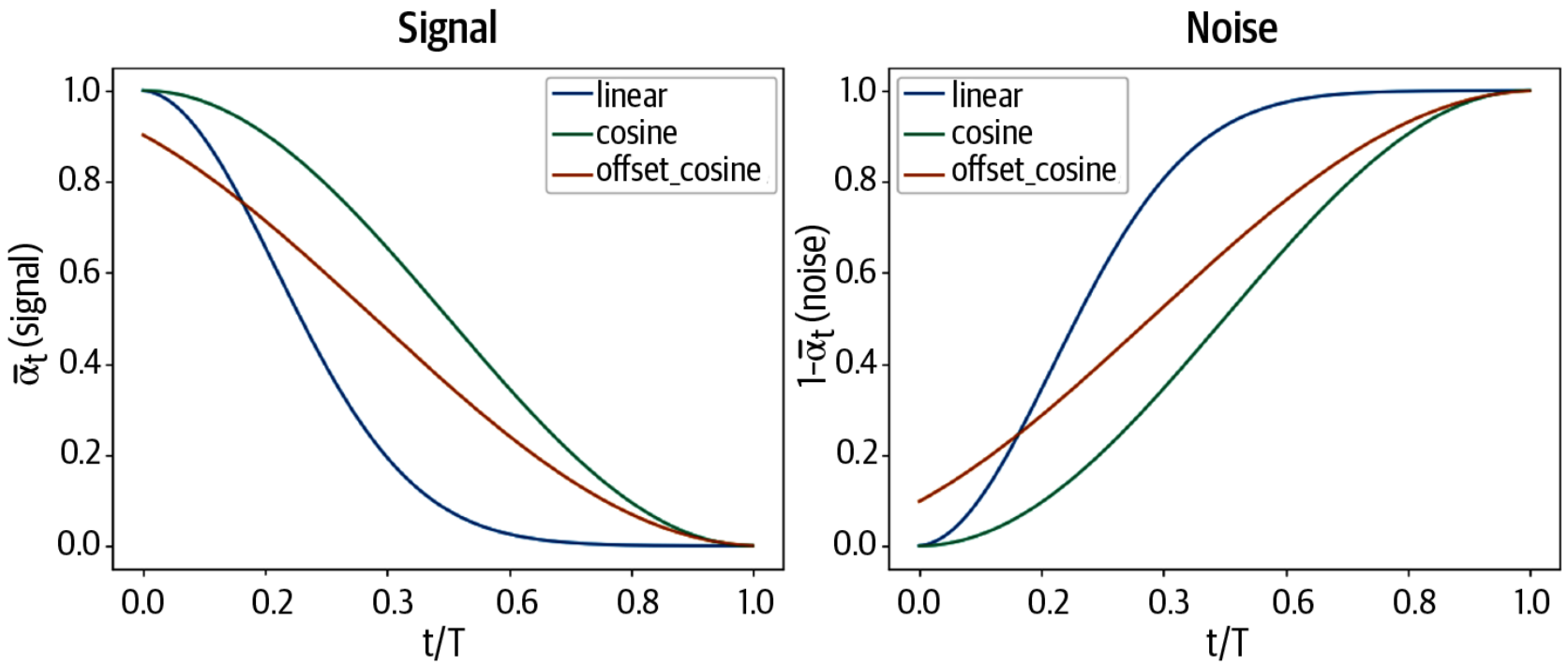
bepec CT Programs

Notice that we are also free to choose a different βt at each timestep—they don't all have be the same. How the βt (or α) values change with T is called the *diffusion schedule*.

1.Linear Diffusion Scheduler: In this method, the noise is added or removed linearly over the steps of the diffusion process. The idea is to incrementally add noise to an image (or data point) in a linear fashion across several steps, gradually transforming it into a random noise image. Similarly, the reverse process involves linearly removing the noise to reconstruct the original image from the noise.
2.Cosine Diffusion Scheduler: This approach uses a cosine function to manage the noise levels instead of a linear function. The cosine function can provide a smoother transition between noise levels compared to the linear method. This can lead to improved quality in the generated images or data, as the noise schedule is better adapted to the natural properties of the data.
3.Offset Cosine Diffusion Scheduler: This is a variation of the cosine scheduler where an offset is introduced in the cosine function. The offset can allow for more control over the noise schedule, potentially leading to better performance in certain applications. It offers a different way to handle the noise transitions, which might be advantageous in specific scenarios or datasets.

Diffusion Schedule

bepec CT Programs



Diffusion Models

bepec CT Programs

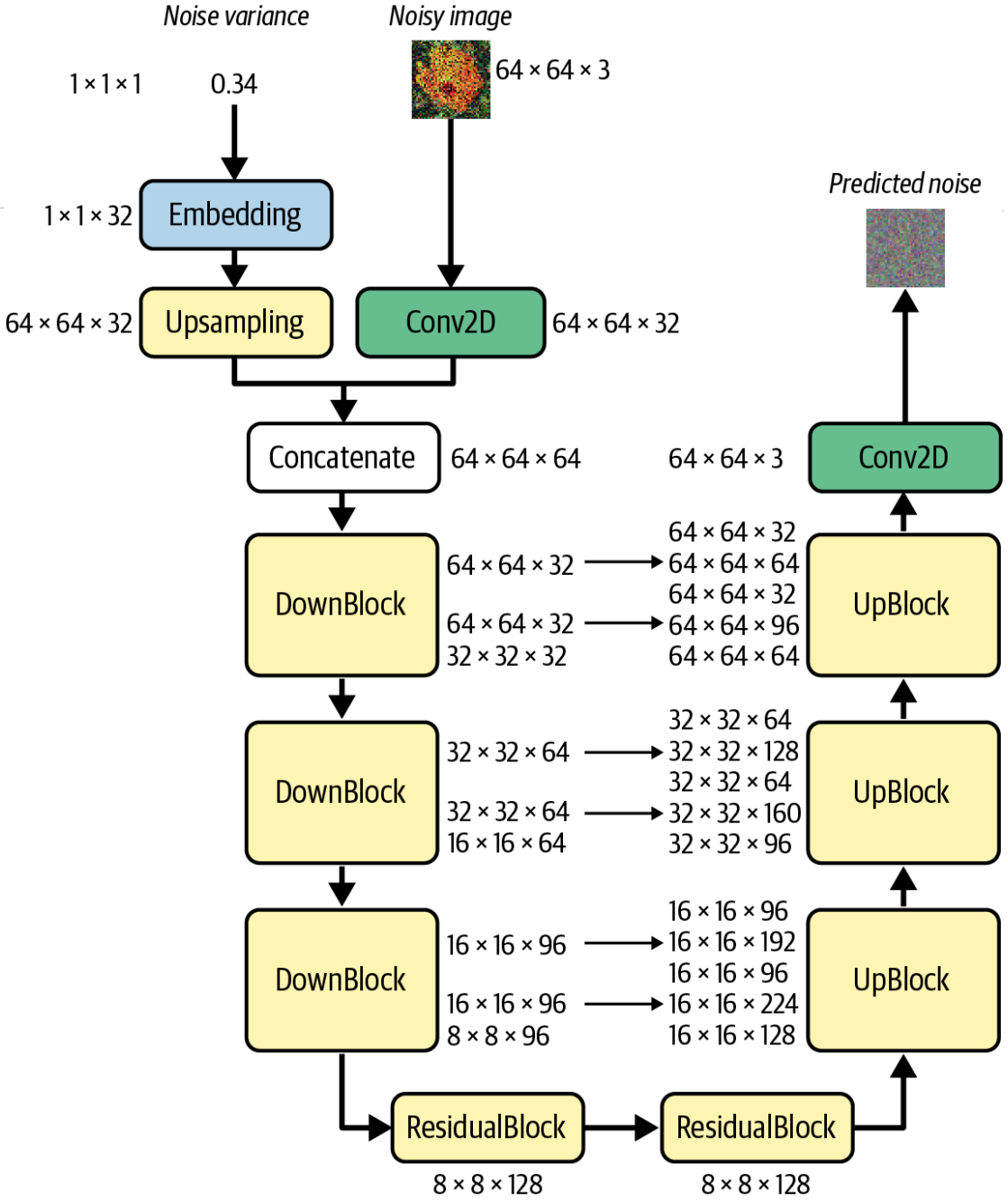
- We first normalize the batch of images to have zero mean and unit variance.
- Next, we sample noise to match the shape of the input images.
- We also sample random diffusion times...
- And use these to generate the noise and signal rates according to the cosine diffusion schedule.
- Then we apply the signal and noise weightings to the input images to generate the noisy images.
- Next, we denoise the noisy images by asking the network to predict the noise and then undoing the noising operation, using the provided noise rate and signal rates
- We can then calculate the loss (mean absolute error) between the predicted noise and the true noise...
- And take a gradient step against this loss function.
- The EMA network weights are updated to a weighted average of the existing EMA weights and the trained network weights after the gradient step.

EMA Network

bepec CT Programs

- In diffusion models, particularly those used for tasks like image generation, the neural network learns to reverse the diffusion process, effectively denoising the data at each step.
- The EMA network is a secondary network whose parameters are the exponential moving average of the primary network's parameters. If the primary network's parameters are changing rapidly due to the stochastic nature of the training process, the EMA network's parameters change more smoothly.
- During training, the primary network's parameters are updated regularly via backpropagation. In parallel, the EMA network's parameters are updated using the EMA formula.
- The use of an EMA network can act as a form of implicit regularization, helping to prevent overfitting to the training data.
- While both the primary network and the EMA network are trained simultaneously, during inference (when generating new samples or reconstructing data), it's common to use the EMA network. This is because the EMA network's parameters, being a smoothed version, often yield better and more consistent results.

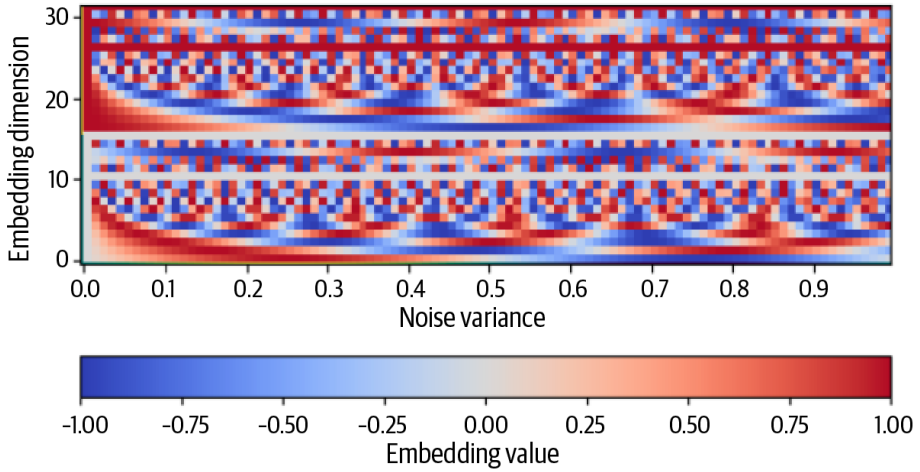
U-Net


Noise variance
Noisy image
1 × 1 × 1
0.34
64 × 64 × 3
1 × 1 × 32
Embedding
Predicted noise
64 × 64 × 32
Upsampling
Conv2D
64 × 64 × 32
Concatenate
64 × 64 × 64
64 × 64 × 3
Conv2D
DownBlock
64 × 64 × 32
64 × 64 × 32
32 × 32 × 32
64 × 64 × 32
64 × 64 × 64
64 × 64 × 32
64 × 64 × 96
64 × 64 × 64
UpBlock
DownBlock
32 × 32 × 64
32 × 32 × 64
16 × 16 × 64
32 × 32 × 64
32 × 32 × 128
32 × 32 × 64
32 × 32 × 160
32 × 32 × 96
UpBlock
DownBlock
16 × 16 × 96
16 × 16 × 96
8 × 8 × 96
16 × 16 × 96
16 × 16 × 192
16 × 16 × 96
16 × 16 × 224
16 × 16 × 128
UpBlock
ResidualBlock
ResidualBlock
8 × 8 × 128
8 × 8 × 128

Sinusoidal Encoding

bepec CT Programs

- The purpose of this is to create a continuous representation of positions, often used in models that process sequential data, like Transformers.
- The idea is that we want to be able to convert a scalar value (the noise variance) into a distinct higher-dimensional vector that is able to provide a more complex representation, for use downstream in the network.



Residual Block

bepec CT Programs

- A *residual block* is a group of layers that contains a skip connection that adds the input to the output. Residual blocks help us to build deeper networks that can learn more complex patterns without suffering as greatly from vanishing gradient and degradation problems.
- The degradation problem is the fact that as neural networks become deeper, they are not necessarily as accurate as their shallower counterparts—accuracy seems to become saturated at a certain depth and then degrade rapidly.

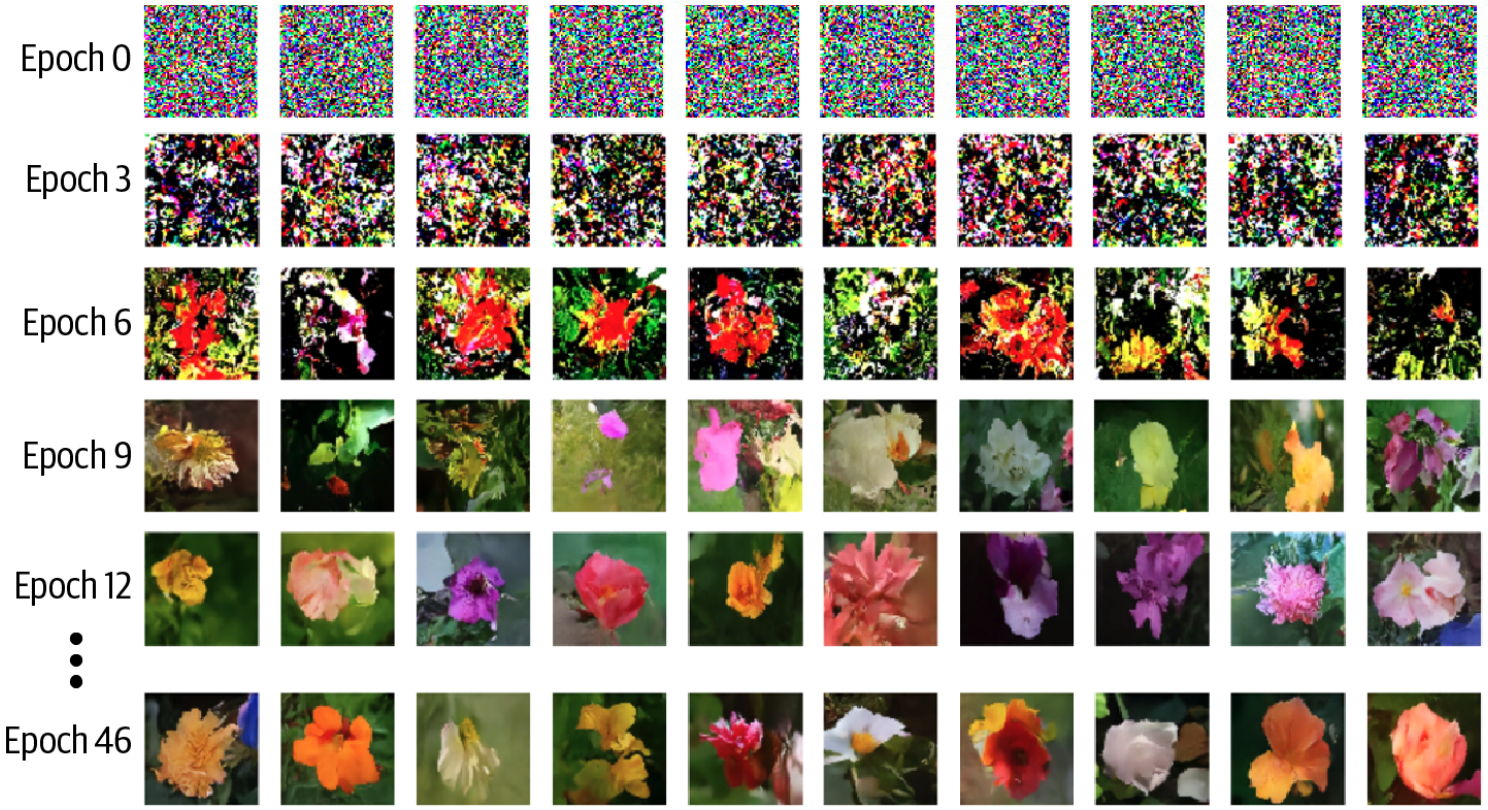
Denoising Steps:

bepec CT Programs

- Look over a fixed number of steps (e.g., 30).
- The diffusion times are all set to 1 (i.e., at the start of the reverse diffusion process).
- The noise and signal rates are calculated according to the diffusion schedule.
- The U-Net is used to predict the noise, allowing us to calculate the denoised image estimate.
- The diffusion times are reduced by one step.
- The new noise and signal rates are calculated.
- The t-1 images are calculated by reapplying the predicted noise to the predicted image, according to the t-1 diffusion schedule rates.
- After 30 steps, the final $\mathbf{x}0$ predicted images are returned.

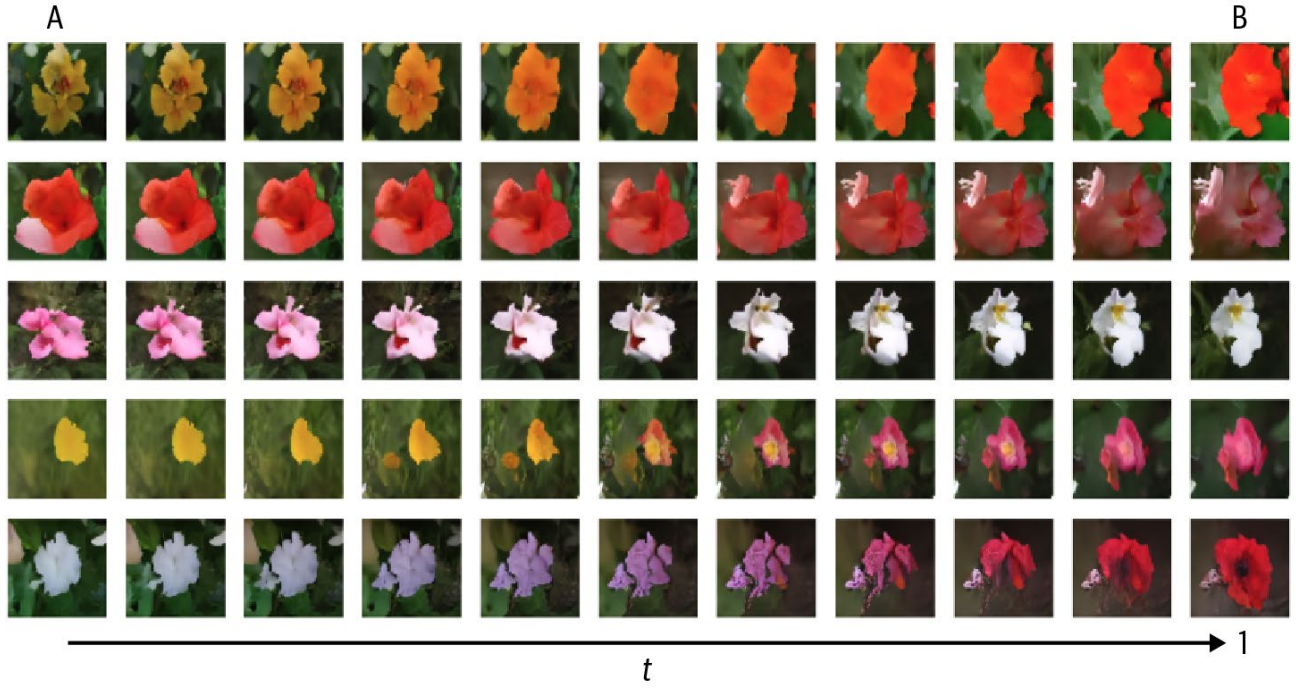
Denoising Steps:

bepec CT Programs



Interpolation:

bepec CT Programs



$$a \sin\left(\frac{\pi}{2}t\right) + b \cos\left(\frac{\pi}{2}t\right)$$