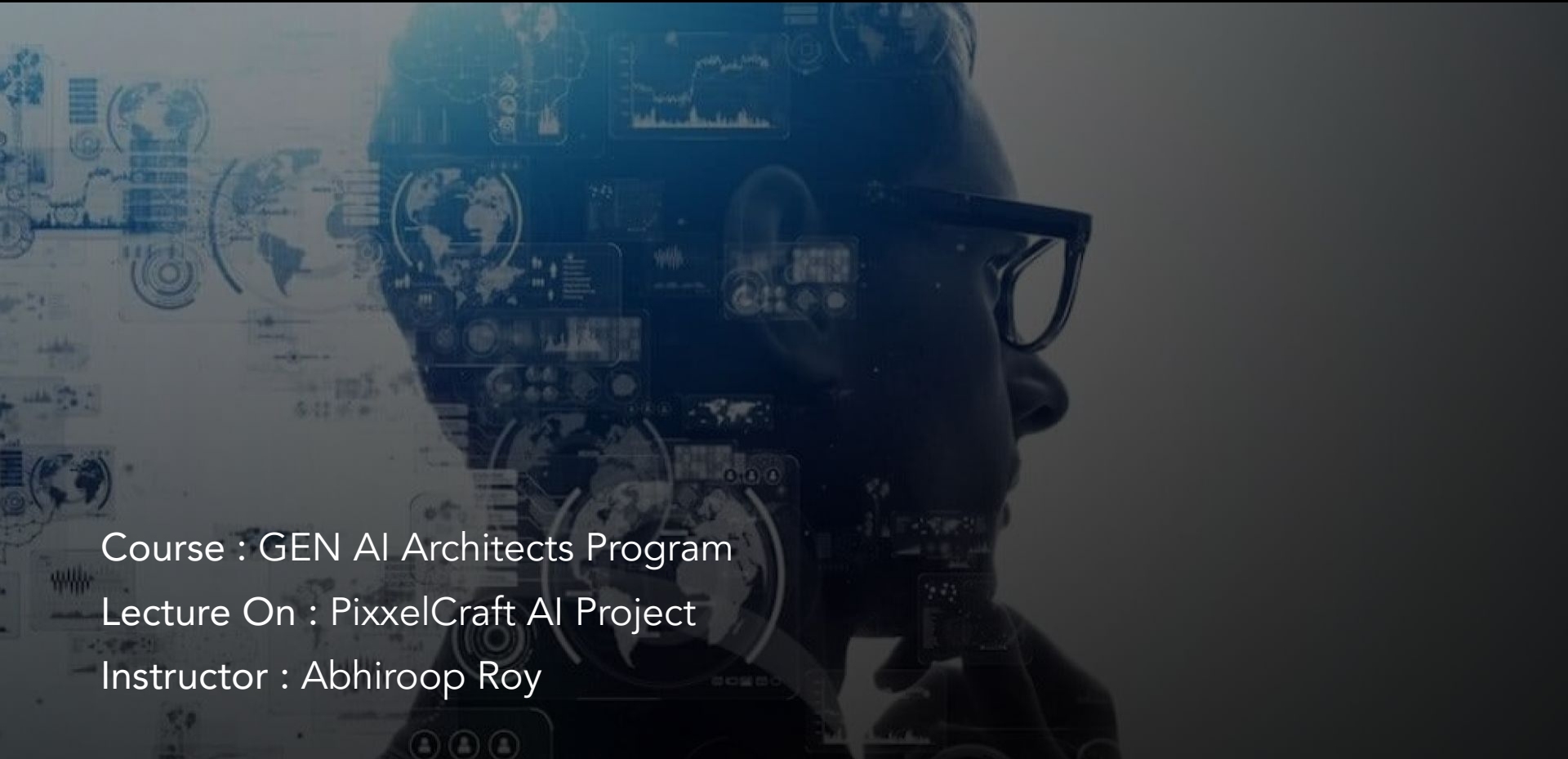
upGrad ENTERPRISE

GEN AI Architects Program - Hexaware

4th October, 2024

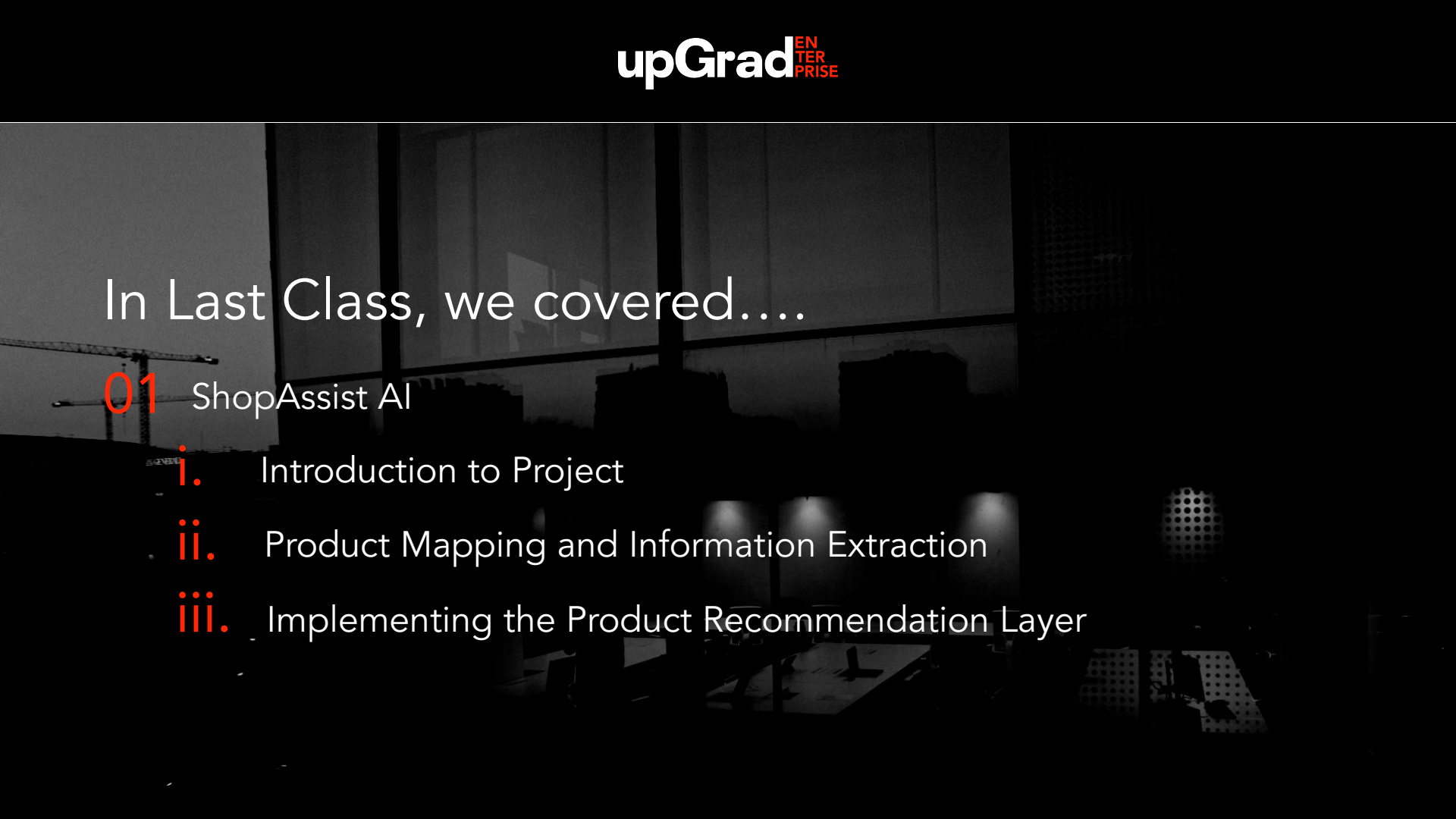
upGrad ENTERPRISE



Course : GEN AI Architects Program

Lecture On : PixxelCraft AI Project

Instructor : Abhiroop Roy



In Last Class, we covered….

01 ShopAssist AI

i. Introduction to Project

ii. Product Mapping and Information Extraction

iii. Implementing the Product Recommendation Layer

Today's Agenda

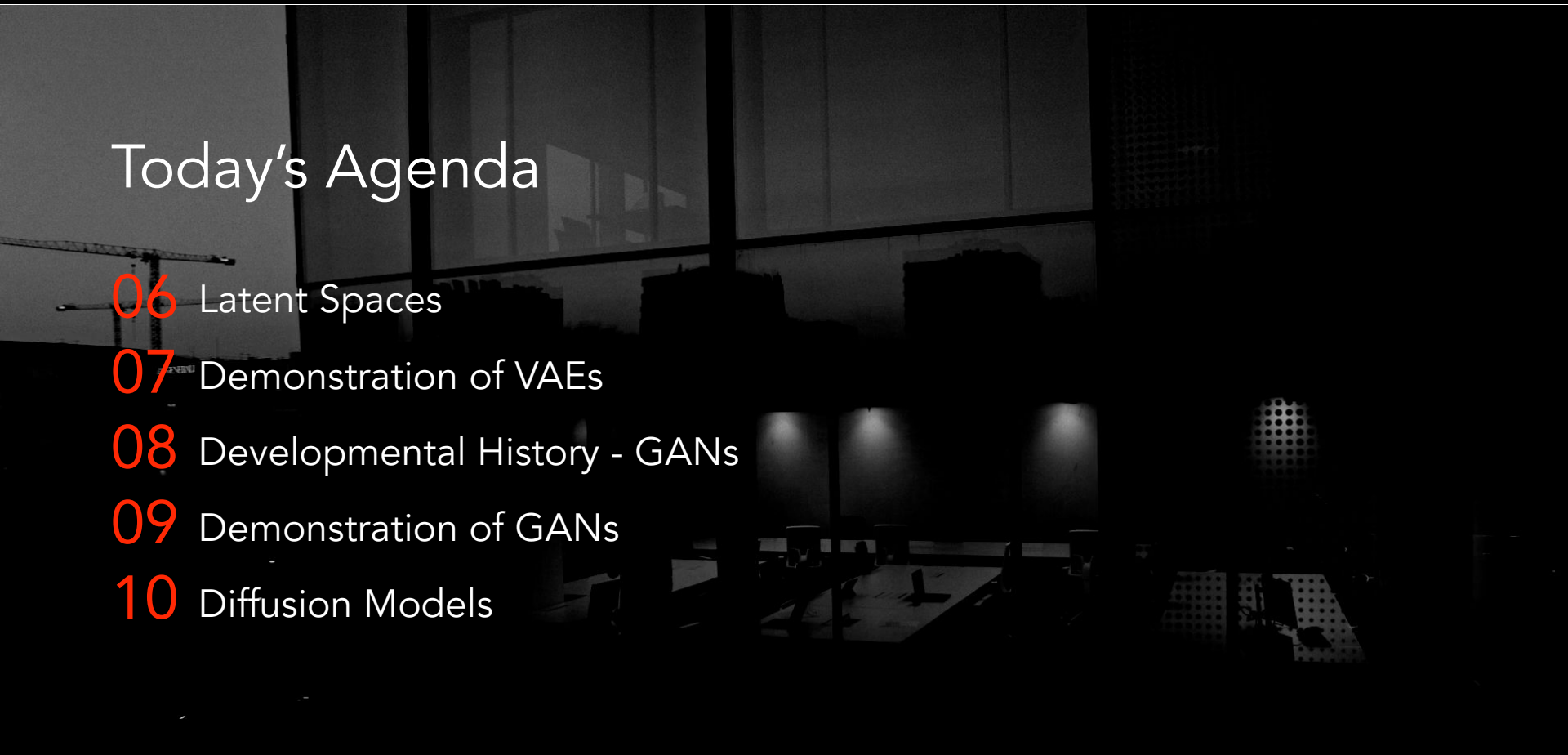
01 An Introduction to Diffusion Models

02 Contemporary Image Generation Models

03 Images and Their Representation

04 Developmental History

05 Developmental History - VAEs



Today's Agenda

06 Latent Spaces

07 Demonstration of VAEs

08 Developmental History - GANs

09 Demonstration of GANs

10 Diffusion Models

Today's Agenda

11 Forward and Reverse Diffusion

12 Noise Predictor

13 Training Diffusion Models

14 Image Generation Conditioned by Text

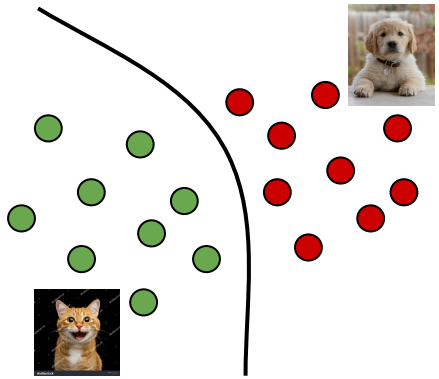
upGrad ENTERPRISE

Introduction

"Portrait of Indian village woman at a gathering in the forests of Himachal Pradesh , Cinematic, Photoshoot, Shot on 25mm lens, Depth of Field, Tilt Blur, Shutter Speed 1/1000, F/22, White Balance, 32k, Super-Resolution, Pro Photo RGB, Half rear Lighting"

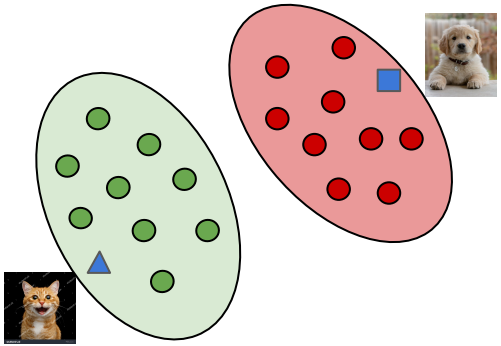
DISCRIMINATIVE VS GENERATIVE MODELS

Discriminative



Classify the new image as a cat or a dog

Generative



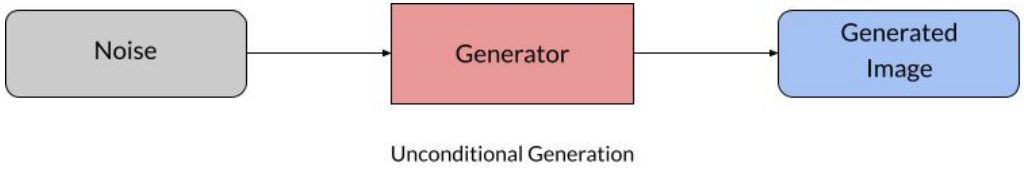
Generate a new image that looks similar to the cats in this data set

WHAT ARE IMAGE GENERATION MODELS?

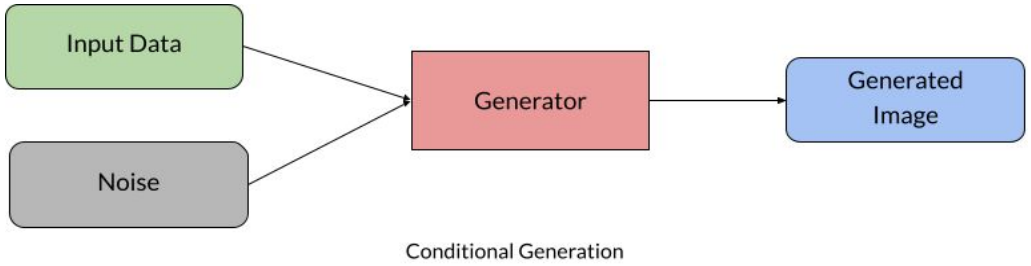
Image generation models take in an input (text, noise, images, and so on) and use that to sample an image from the appropriate distribution

UNCONDITIONAL VS CONDITIONAL GENERATION

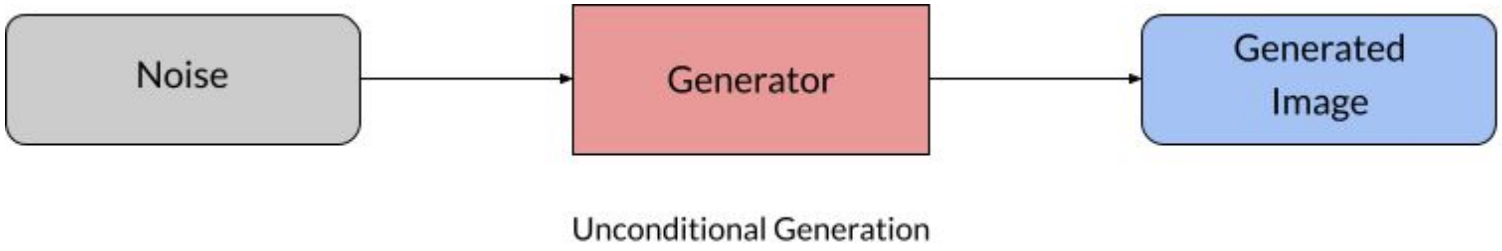
- Unconditional image generation: Generate samples unconditionally from the dataset



- Conditional image generation: Generate samples conditionally from the dataset, based on an input (class, text, image, and so on)



UNCONDITIONAL GENERATIVE MODELS

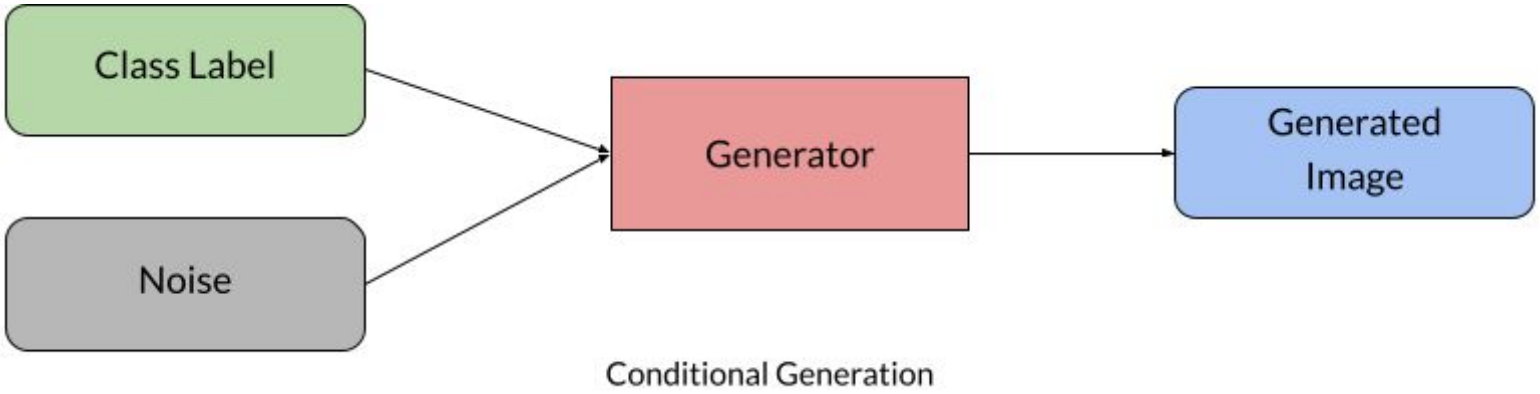


Unconditional Generation

Only input is noise. For example, GANs, DDPMs, …

GAN: Generative Adversarial Network
DDPM: Denoising Diffusion Probabilistic Model

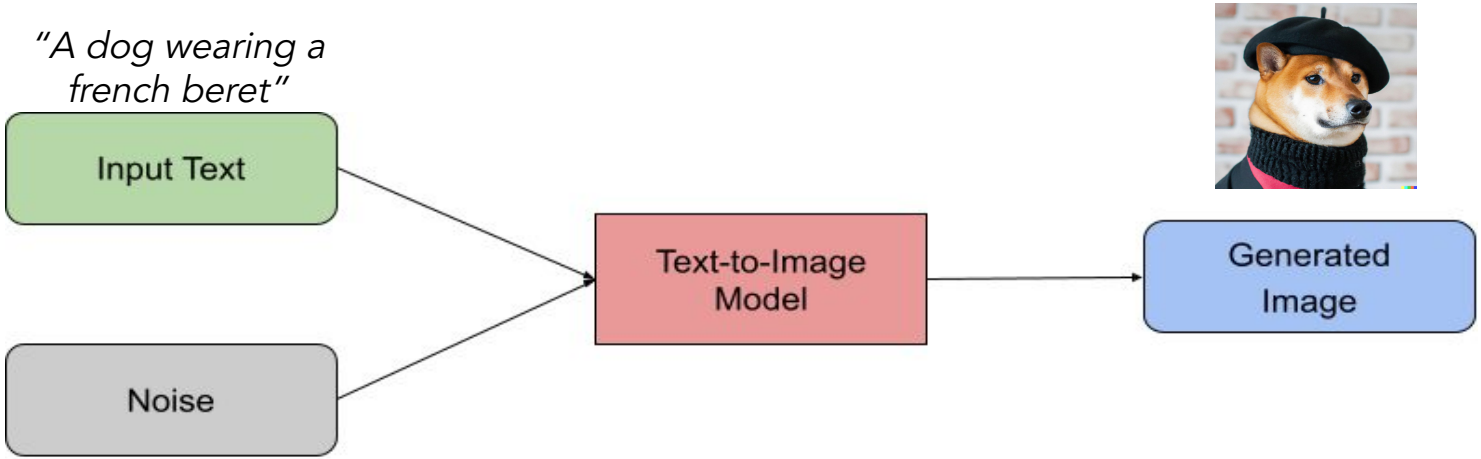
CONDITIONAL GENERATIVE MODELS



Conditional Generation

Class label is used as an input along with noise. For example, Conditional GANs, …

CONDITIONAL GENERATIVE MODELS | TEXT-TO-IMAGE



Text is used as an input along with noise. For example, Stable Diffusion, DALL-E, …

CONDITIONAL GENERATIVE MODELS | IMAGE-TO-IMAGE

Image-to-image Generation

Taking image as an input and giving another image as an output. Use cases can be image colorization, style transfer, inpainting, superresolution, …

MULTIMODAL GENERATIVE MODELS

"Describe this picture"

Text + Image → Multi-Modal Generative Model → Text

This is a horse

"Provide a picture of a horse"

Text → Multi-Modal Generative Model → Image

"Write a song about a horse"

Text → Multi-Modal Generative Model → Audio

Multi-modal models can accept prompts from different modalities and produce results in multiple modalities

upGrad ENTERPRISE

Contemporary Image Generation Models

EVOLUTION OF IMAGE GENERATION MODELS

- Shift from older models (e.g., VAEs and GANs) to newer diffusion models
- Driven by the pursuit of:
  - Higher quality and realistic image generation
  - Improved scalability for complex datasets
  - Addressing uncertainty and training stability challenges
- Notable diffusion models: DALL·E, Stable Diffusion, Midjourney, Imagen

CONTEMPORARY DIFFUSION MODELS | DALL-E

- Text-to-image model
- Developed by OpenAI
- Initial release January 5, 2021
- Transformer language model

DALL·E

DALL·E: Creating images from text

"panda mad scientist mixing sparkling chemicals, digital art"

CONTEMPORARY DIFFUSION MODELS | STABLE DIFFUSION

- Text-to-image model
- Original authors Runway, CompVis, and Stability AI
- Developed by Stability AI
- Initial release August 22, 2022

Stable Diffusion

High-Resolution Image Synthesis with Latent Diffusion Models

"a photograph of an astronaut riding a horse"

CONTEMPORARY DIFFUSION MODELS | MIDJOURNEY

- Founded by David Holz
- Developer Midjourney, Inc.
- Initial release July 12, 2022

Midjourney

"mechanical dove"

CONTEMPORARY DIFFUSION MODELS | IMAGEN

- A text-to-image diffusion model
- Developed by Google Inc.
- Creates photo realistic images

Google Research

Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

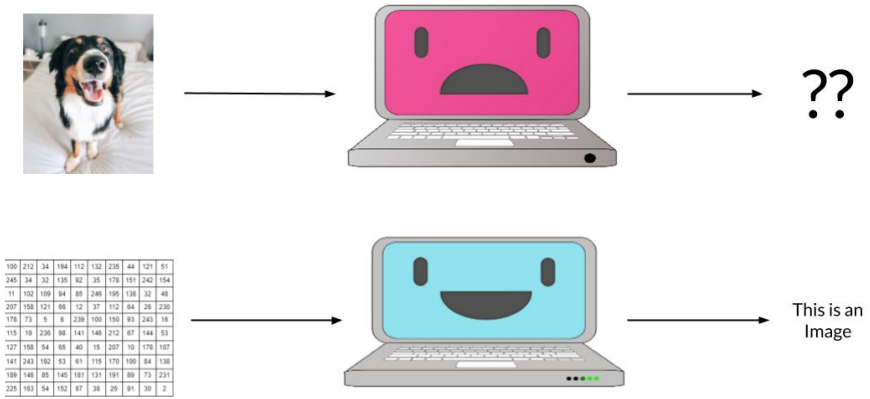
"A transparent sculpture of a duck made out of glass."

upGrad ENTERPRISE

Images and Their Representation

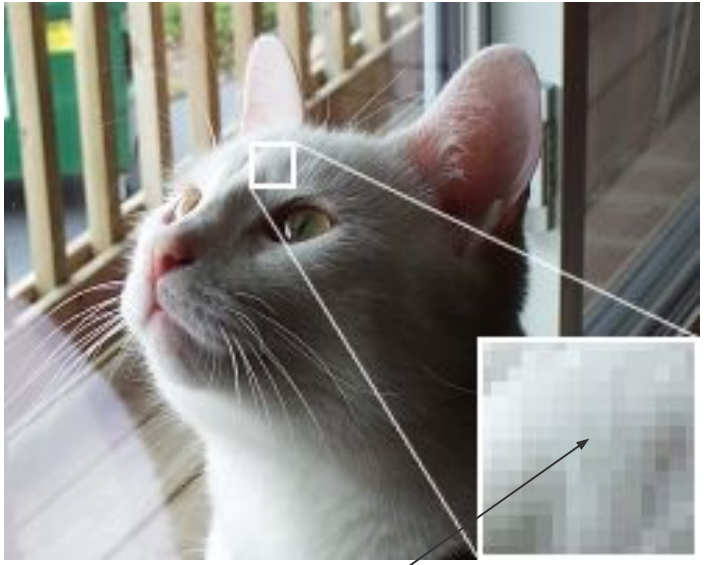
WHY STUDY IMAGES?

- Images are:
  - Spatial data structured in pixel grids
  - Represented as arrays of pixel values
- Understanding of image structure and storage is required for image generation



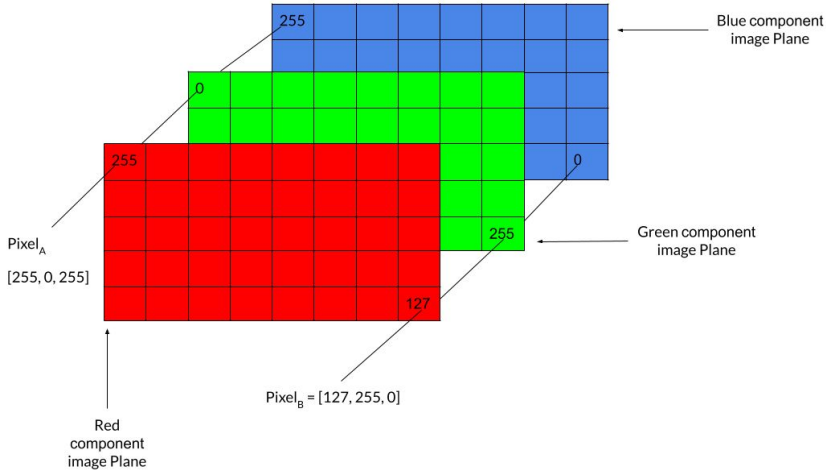
PIXELS

- The term **pixel** is a portmanteau of "picture" and "element"
- Images are represented by a grid of pixels that store color information
- The number of pixels define the resolution of the image



RGB

- Images are typically divided into three channels corresponding to the colours red, green, and blue
- By varying the brightness (also called gray levels) of different channels, we can create a vast array of colors
- RGB representation is used by JPEG, PNG, GIF, …



CHANNELS

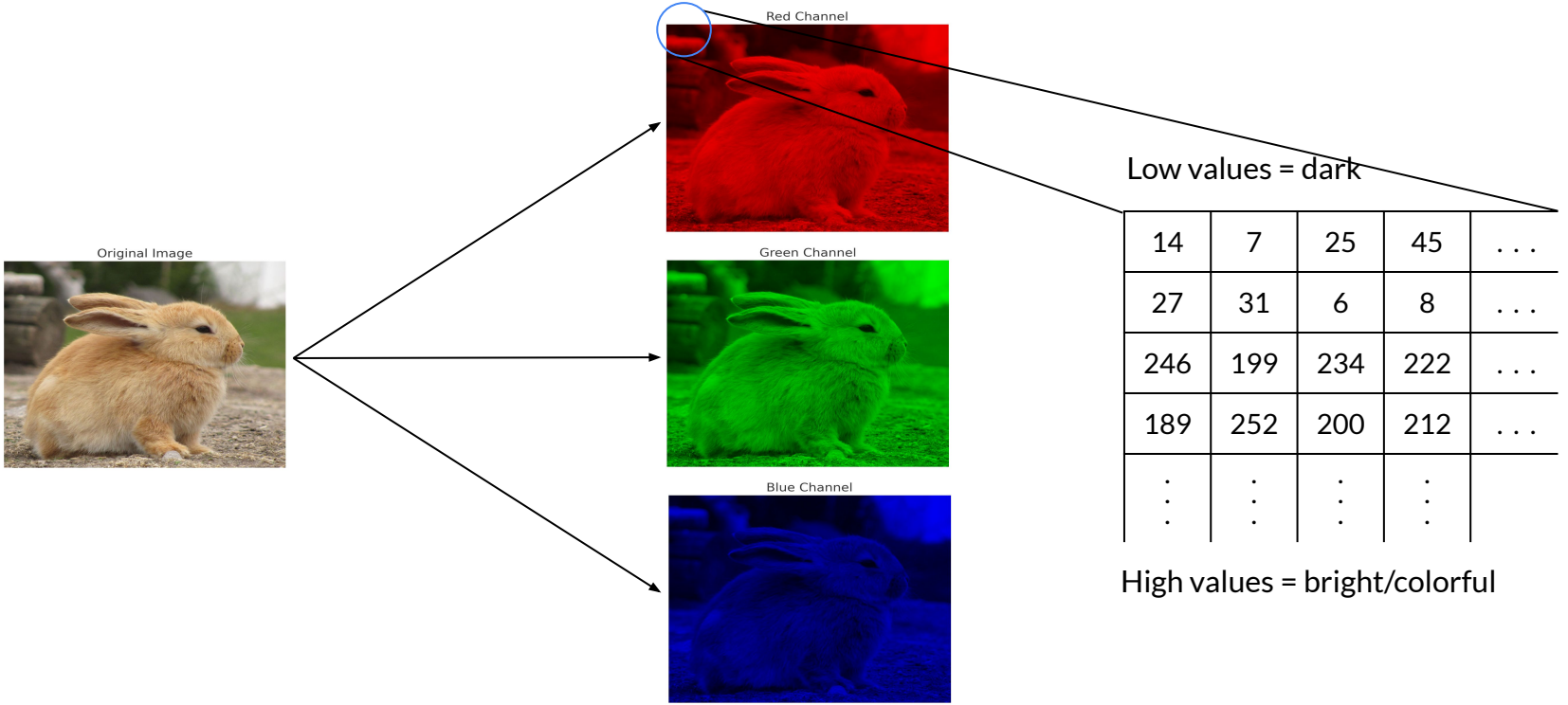

Red Channel
Original Image
Green Channel
Blue Channel
Low values = dark
14 7 25 45 . . .
27 31 6 8 . . .
246 199 234 222 . . .
189 252 200 212 . . .
High values = bright/colorful

IMAGE REPRESENTATION IN PYTHON: *PILLOW*

- A Python library for opening, manipulating, and saving various image file formats
- Easy-to-use functions for opening images, resizing, cropping, adding text, and more

IMAGE REPRESENTATION IN PYTHON: *PILLOW*

- Opening and displaying images

```
from PIL import Image
image = Image.open('image.jpg')
image.show()
```

- Resizing images

```
new_size = (new_width, new_height)
resized_image = image.resize(new_size)
```

- Saving images

```
resized_image.save('resized_image.jpg')
```

IMAGE REPRESENTATION IN PYTHON: numpy, cv2, matplotlib

- Another pipeline used for manipulating images consists of the libraries *cv2*, *matplotlib*, and *numpy*
- *cv2* is the Python implementation of the OpenCV library
- It is used to load images in the form of a NumPy array
- Matplotlib can then be used to display the images

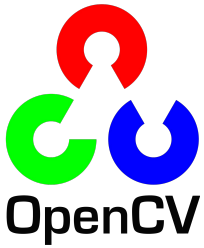


IMAGE REPRESENTATION IN PYTHON: numpy, cv2, matplotlib

- Importing library

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- Loading images

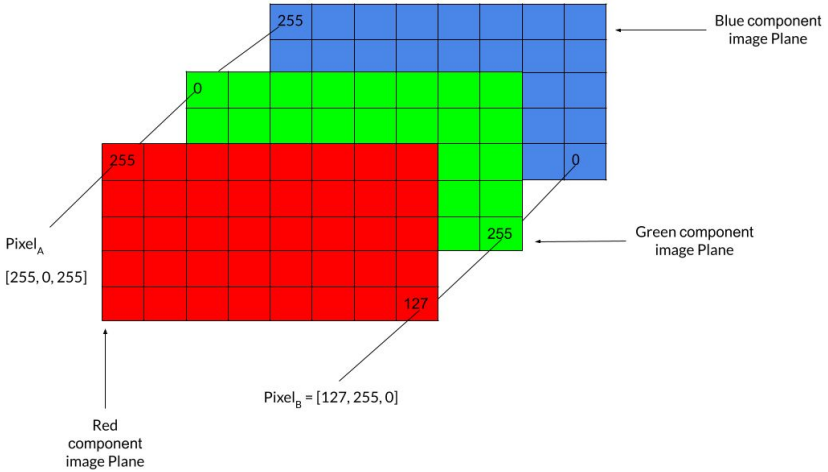
```
image = cv2.imread('image.jpg')
```

- Displaying images

```
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image_rgb)
plt.axis('off')  # Optional: Turn off axis labels
plt.show()
```

WORKING WITH IMAGE DATA IN PYTHON

- Image data is represented as multidimensional arrays (tensors) for compatibility with TensorFlow/Keras
- These arrays can be easily fed into TensorFlow/Keras models for training, validation, or inference
- *pillow*, *matplotlib*, and *scikit-image* offer methods to load, manipulate, and preprocess images
- Objective is to convert images into numerical data arrays regardless of the library used



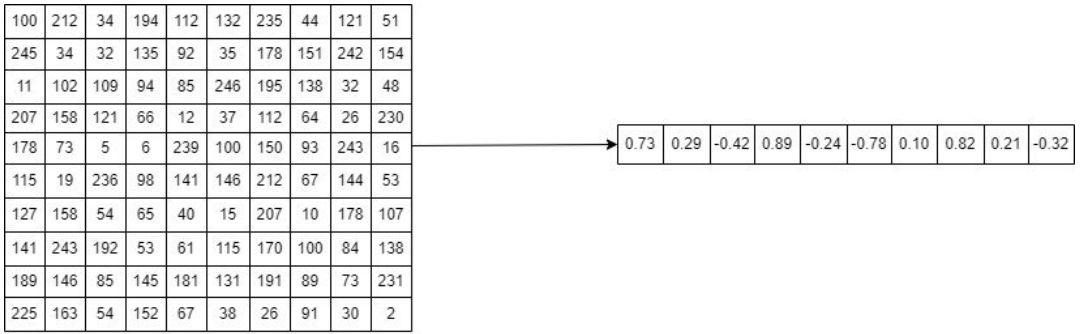
Developmental History

STARTING OFF WITH VAES AND GANS

- Both VAEs and GANs are vital in the field of image generation AI
- They contribute to advancements in image processing, natural language generation, and many other areas
- Studying GANs and VAEs equips you with **fundamental generative modeling knowledge**, probabilistic modeling concepts, and insights into **noise manipulation** and generation
- These fundamentals can then be applied to develop **new** models such as **diffusion** models

PREREQUISITE FOR IMAGE GENERATION: DIMENSIONALITY REDUCTION

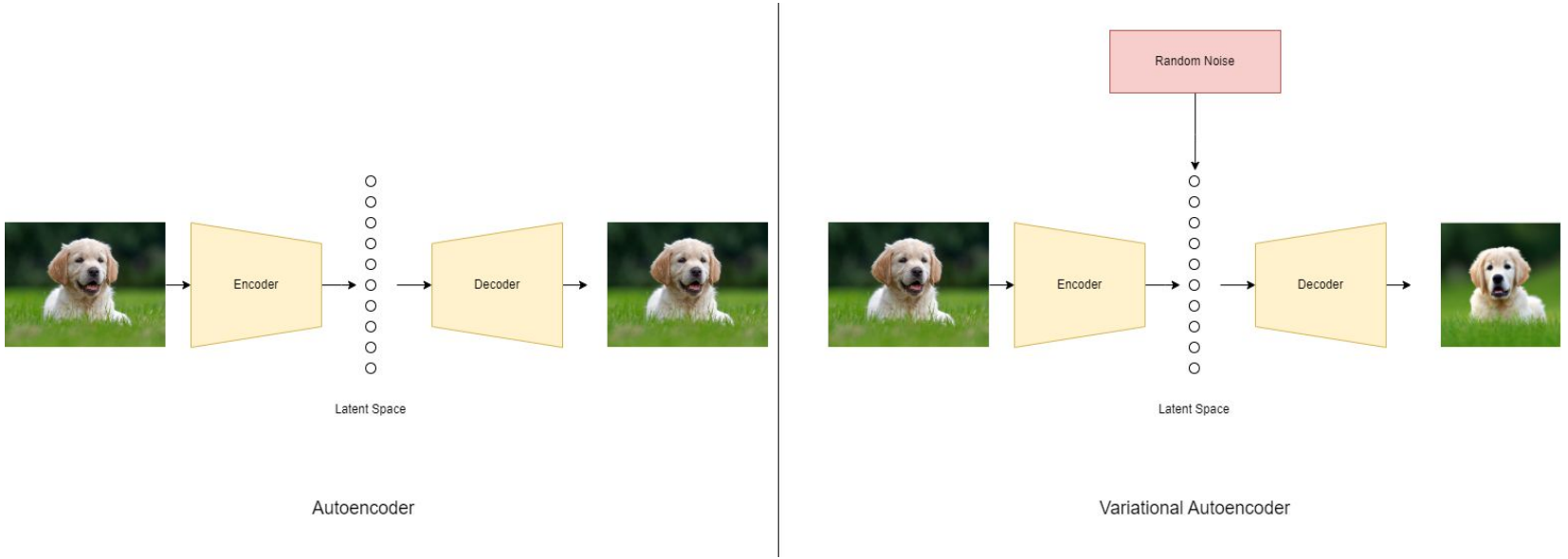
- Dimensionality Reduction: Capture a more compact and meaningful representation of the data
- Controlled Generation: Lower-dimensional spaces can make it easier to control and manipulate the generation process
- Efficiency: Computationally, it is more efficient to work with lower-dimensional spaces



upGrad ENTERPRISE

Developmental History - VAEs

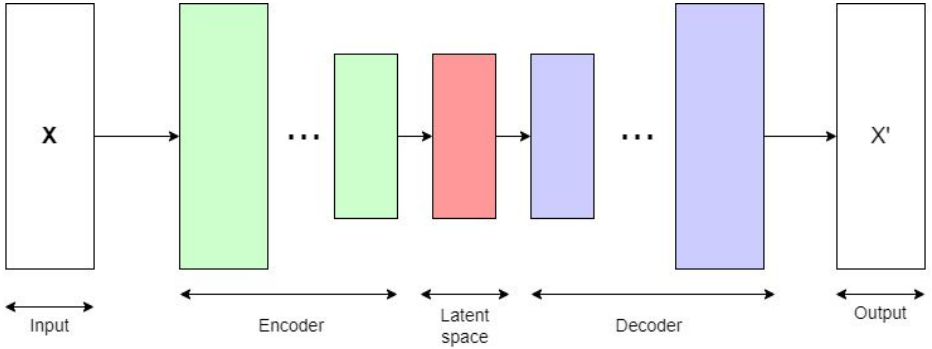
VARIATIONAL AUTOENCODERS



Autoencoders and VAEs are neural networks that use encoder-decoder architecture to create images

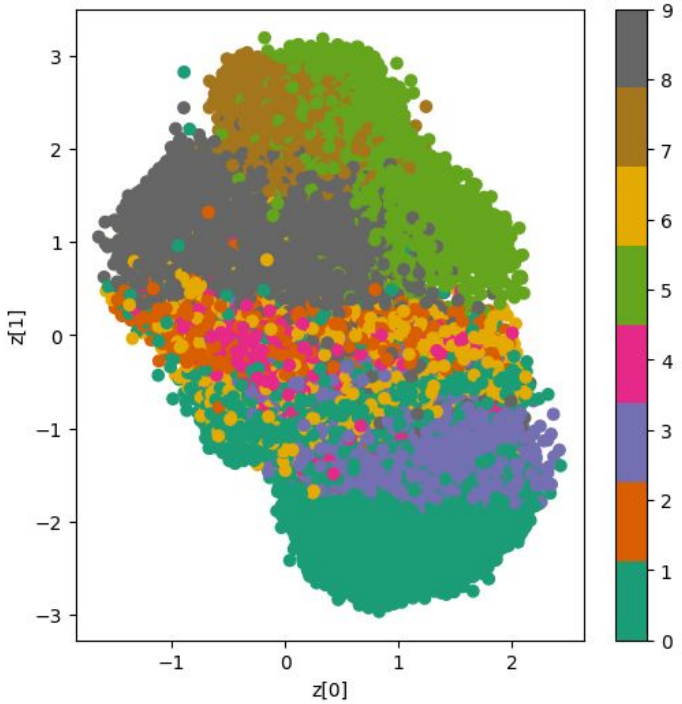
ENCODER-DECODER ARCHITECTURE

- Encoder: One or more layers of neurons that convert the input into an encoded representation
- Latent Space: Lower dimensional representation of the input that captures the most salient features
- Decoder: One or more layers of neurons that recreate the image from the latent space
- Noise: VAEs introduce perturbations in the latent space by adding noise



LATENT SPACE

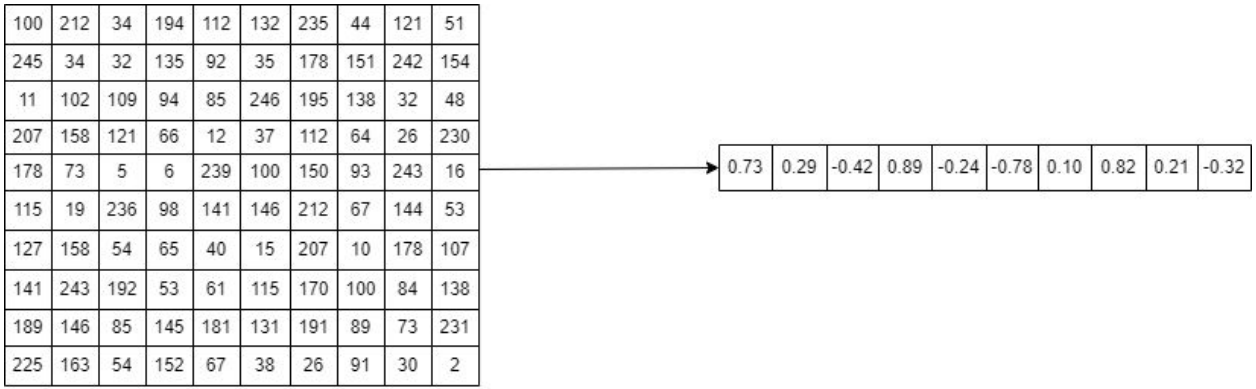
- Lower dimensional representation of the data
- Learns **salient features** of the images in the data set
- Images that are **similar** to each other tend to **cluster** together
- By adding **noise**, we create small perturbations that may create **new images** similar to the input image



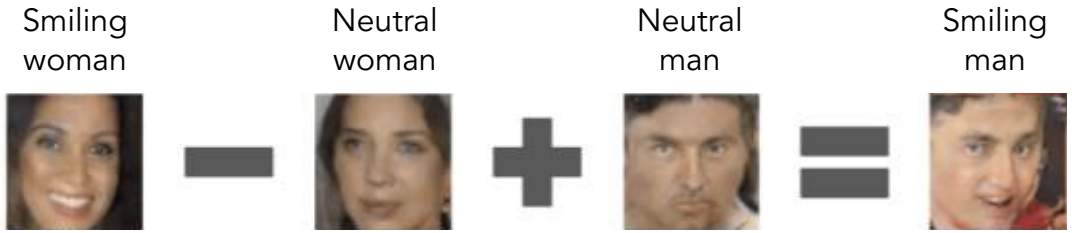
Latent Spaces

NATURE OF LATENT SPACE

- Typically a lower-dimensional representation of data that captures **essential features** or patterns of the higher-dimensional data such as images
- By manipulating these latent space vectors, we can generate new images that share similarities with the original dataset



LATENT VECTOR ARITHMETIC



Suppose [*x*, *y*] represents values of smiling/not and man/woman, such that [1, 0] means smiling woman and [0, 1] means neutral man, the above latent vector arithmetic can be represented as follows:

[1, 0] - [0, 0] + [0, 1] = [1 - 0 + 0, 0 - 0 + 1] = [1,1]

HOW TO USE LATENT SPACES?

Latent space representation

0.43	-0.78	0.12	-0.56	0.98	-0.24	0.65	-0.34	0.87	-0.012

Slight manipulation

0.33	-0.78	0.22	-0.56	0.90	-0.24	0.65	-0.30	0.86	-0.010

upGrad ENTERPRISE

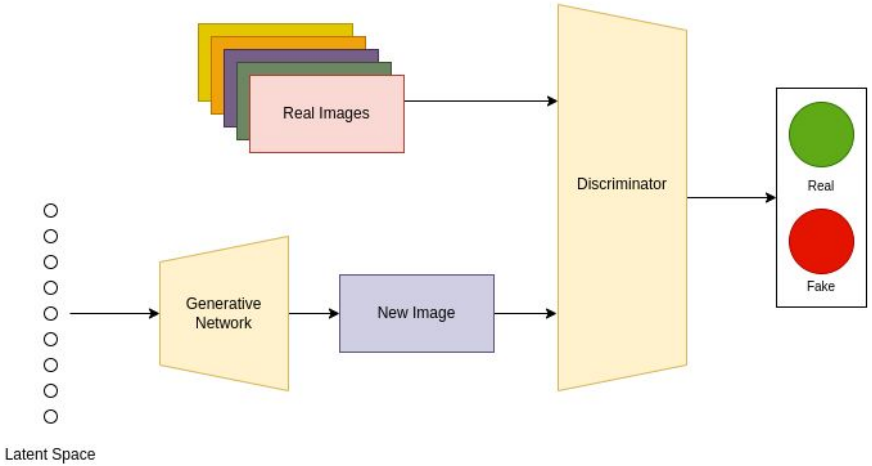
Demonstration - VAEs

upGrad ENTERPRISE

Developmental History - GANs

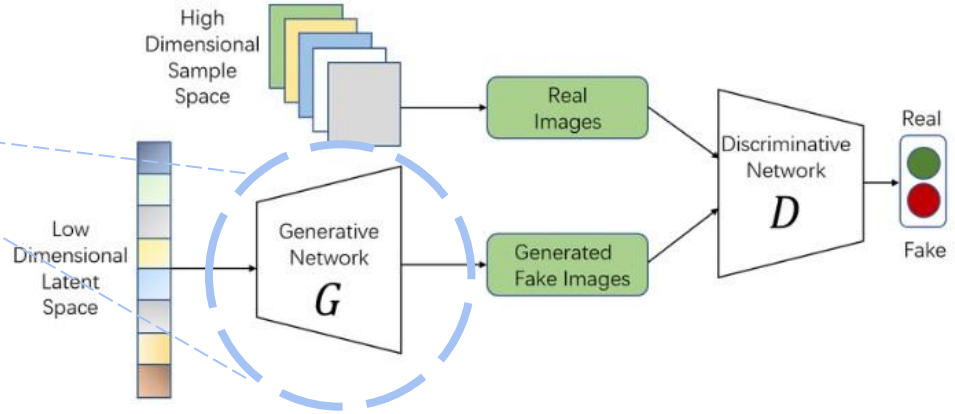
GENERATIVE ADVERSARIAL NETWORKS

- Consists of two networks: Generator and discriminator
- Generator creates new images
- Discriminator tries to determine whether the new images are real or fake
- GANs are trained till the discriminator is unable to classify incoming images as real or fake accurately (accuracy of 50%)

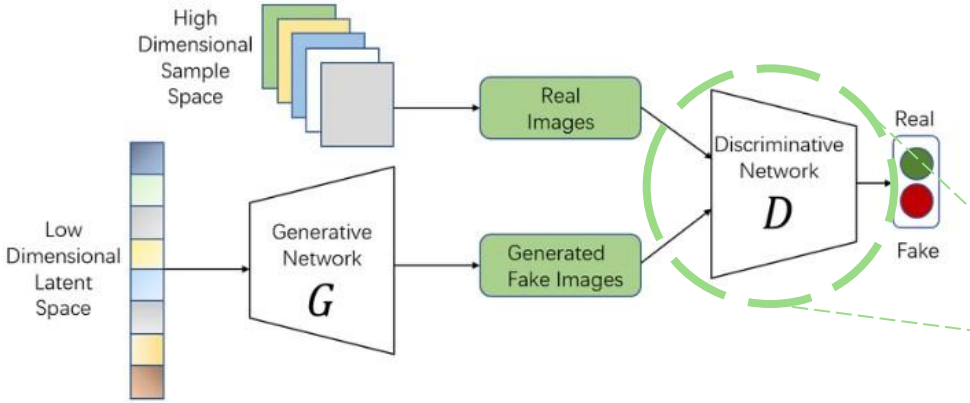


GANS | GENERATOR

Generator: Creates fake data that looks as realistic as possible



GANS | DISCRIMINATOR



Discriminator: Tells the difference between real and fake data

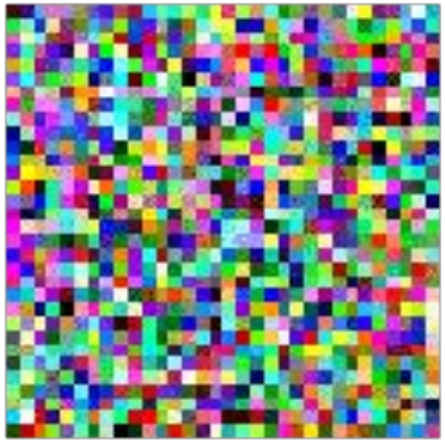
ADVERSARIAL?

- The two networks (Generator and discriminator) try to one up each other
- The generator tries to generate images that are closer and closer to the real images in the data set
- The discriminator tries to determine whether or not the generated images lie in the distribution described by the data set
- The complete network is trained as a whole
- After training, the discriminator is discarded

Demonstration - GANs

Diffusion Models

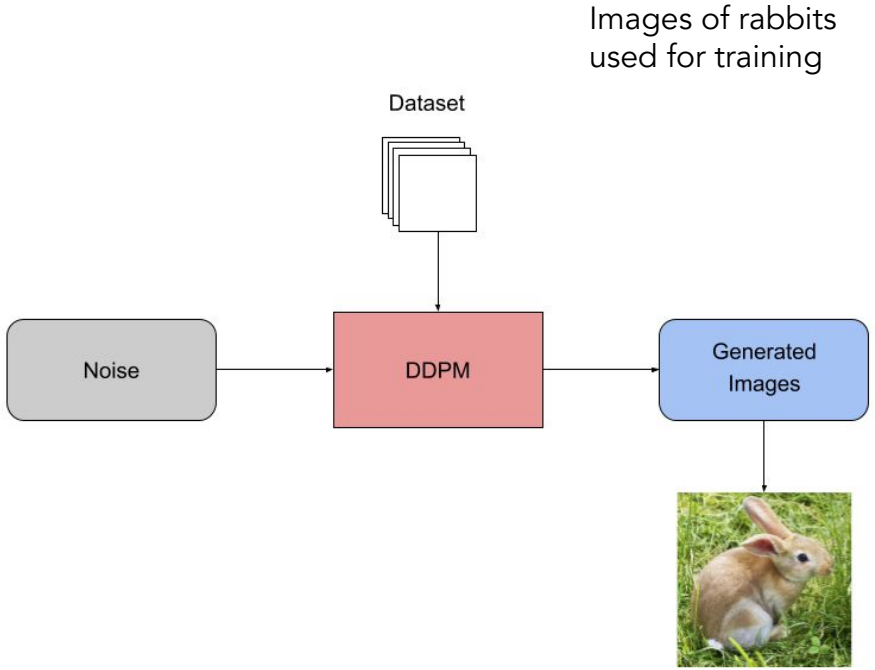
DIFFUSION MODELS



Diffusion models generate images by moving from noise to image

UNCONDITIONAL GENERATION USING DIFFUSION

- Described in *Denoising Diffusion Probabilistic Model* by *Ho et. al.* 2020
- Noise is the only input
- Generated images belong to the distribution represented by the data set

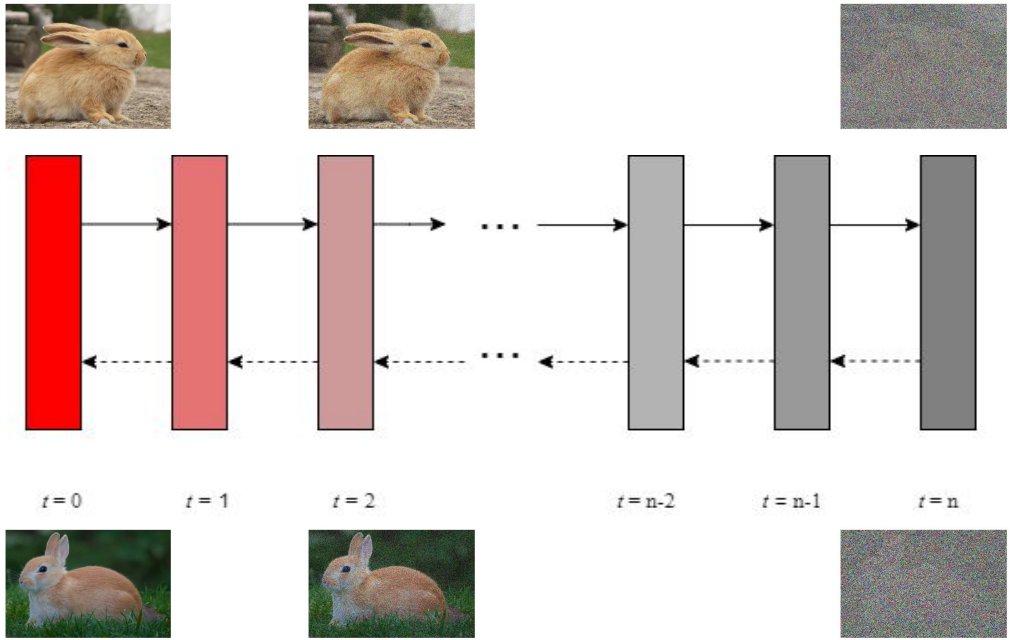


upGrad ENTERPRISE

Forward and Reverse Diffusion

FORWARD AND REVERSE DIFFUSION

Forward diffusion

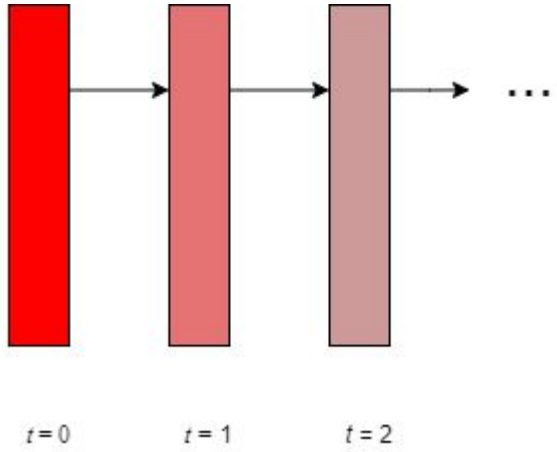


Reverse diffusion

FORWARD DIFFUSION

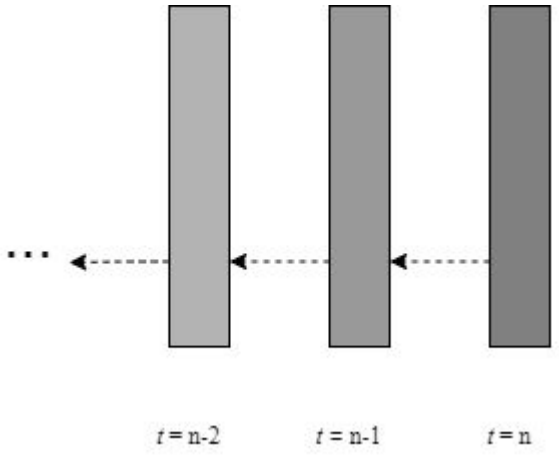
- Happens during **training**
- Add Gaussian noise to the data at each step
- The rate at which noise is added is determined by a **scheduler**
- As Gaussians add up nicely, we can directly get x_t from x_0 using reparametrization
- The point is to enable the network to predict the noise added at any step

$$x_t = x_0 + N(\mu_t, \sigma_t)$$



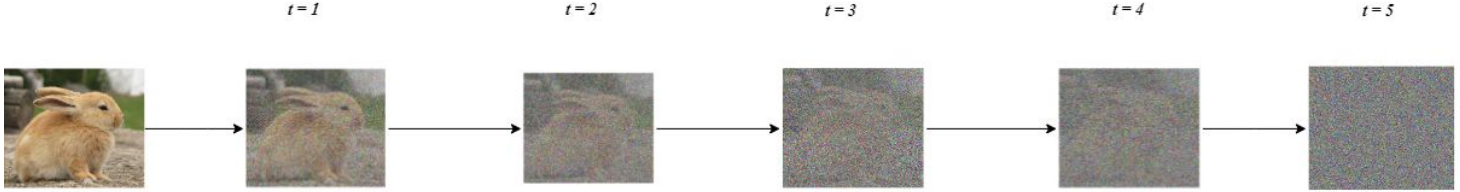
REVERSE DIFFUSION

- Happens during **prediction**
- Similar to forward diffusion, we can directly get x_0 from any x_t
- This gives us an estimate of the final image at the timestep t
- Since the model is trained to predict noise added, it is now able to remove that noise and produce a new image

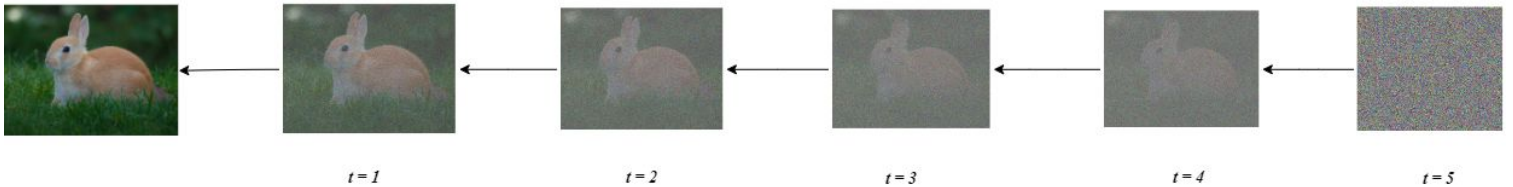


SUMMARIZING THE DIFFUSION PROCESS

Images are used as training data to train the model to predict noise in the forward diffusion process



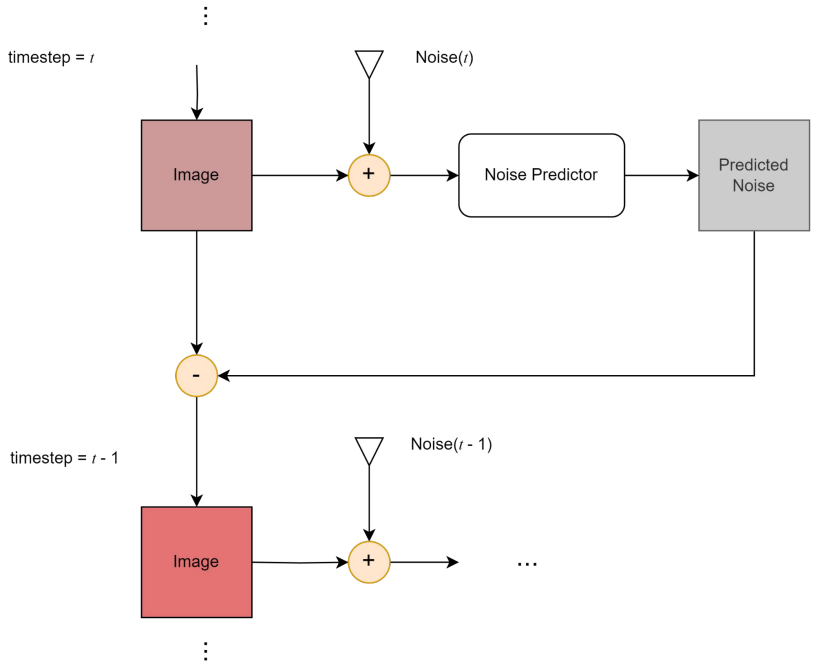
A random noisy patch is then denoised in the reverse diffusion process to produce a new image



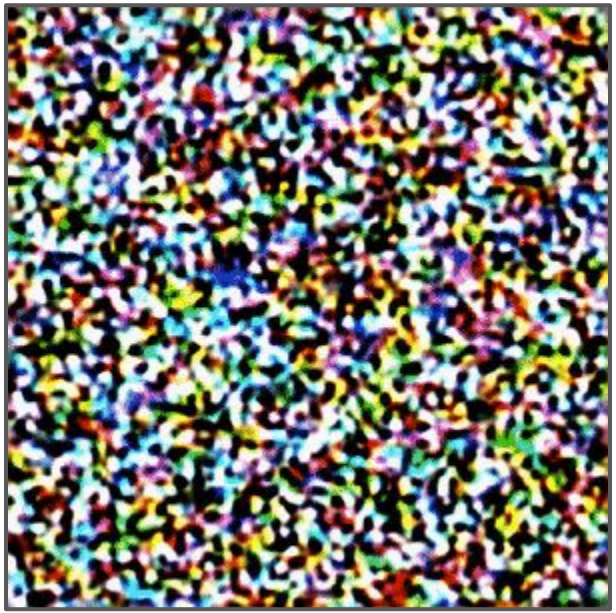
The model has basically learned to remove noise from random noise to produce images similar to the training data

GENERALIZING NUMBER OF NOISING/DENOISING STEPS

Images at each stage are denoised using the predicted noise by using the trained model



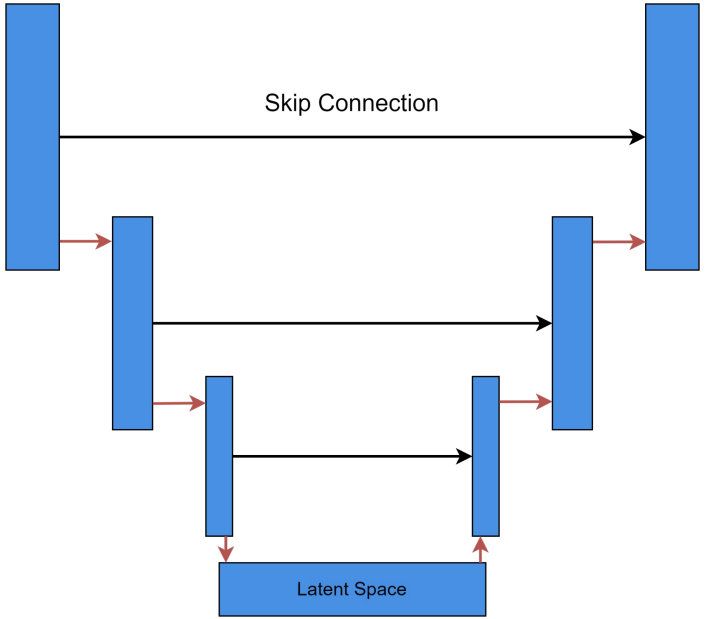
EXAMPLE OF DENOISING A NOISY PATCH



Noise Predictor

NOISE PREDICTION: U-NET

- A U-net is used to predict noise in the diffusion model
- It has an encoder-decoder architecture
- Encoder (left): Converts the input image into a lower-dimensional latent space and extracts meaningful features at multiple stages
- Decoder (right): Gradually increasing dimensions to produce a final output with the same size as the input image
- Skip connections are use to pass information from the encoder to the decoder

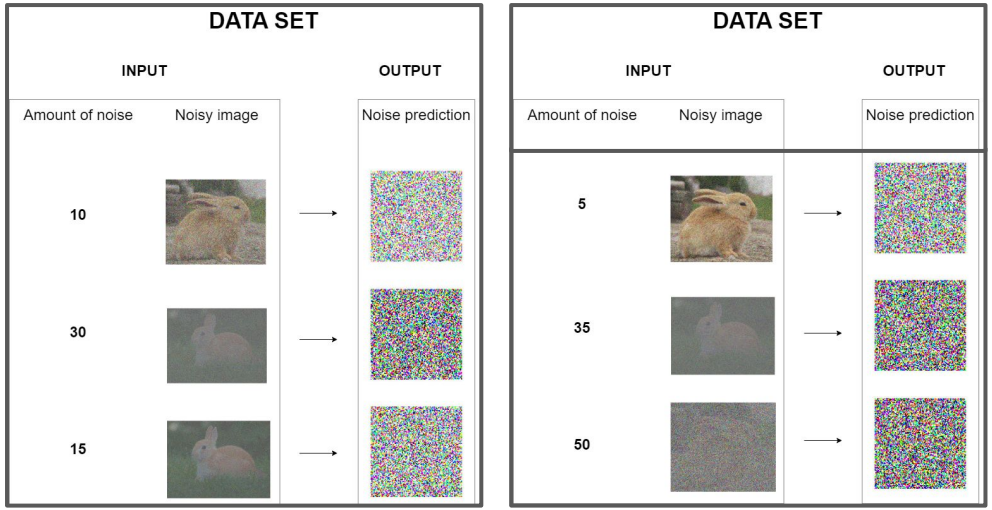


upGrad ENTERPRISE

Training Diffusion Models

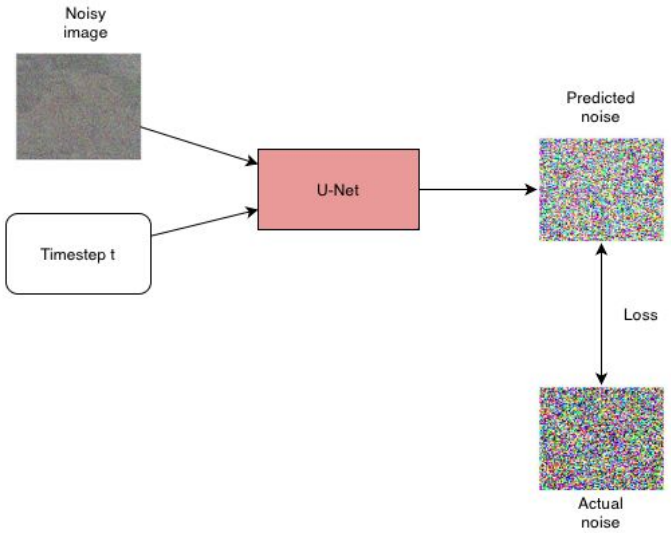
TRAINING A DIFFUSION MODEL

- As you saw in the reverse process, the neural network **predicts** the amount of **noise** in an image
- We **train** the network to predict the noise given an image and a timestep
- A sample data set for this model is given here



TRAINING A DIFFUSION MODEL | LOSS

Loss is defined by the difference between the actual and the predicted noise

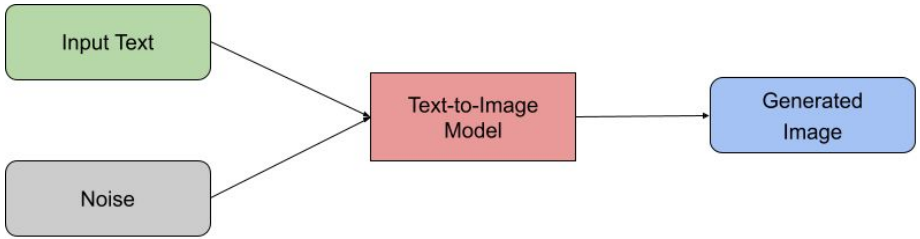


The loss is then used to **train** the model using **backpropagation**

Image Generation Conditioned by Text

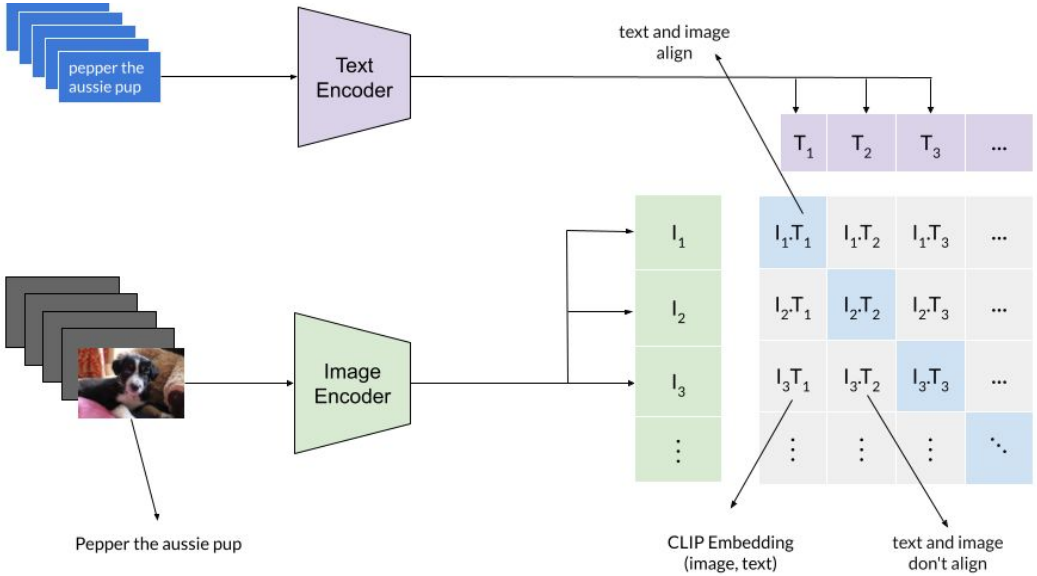
CONDITIONING USING TEXT

- So far, we have seen a diffusion model that can generate images similar to a given data set
- We want to move towards a model that converts from text to image
- Stable Diffusion achieves this, along with a few other optimizations



High-Resolution Image Synthesis with Latent Diffusion Models by Rombach et. al.
Code provided by StabilityAI

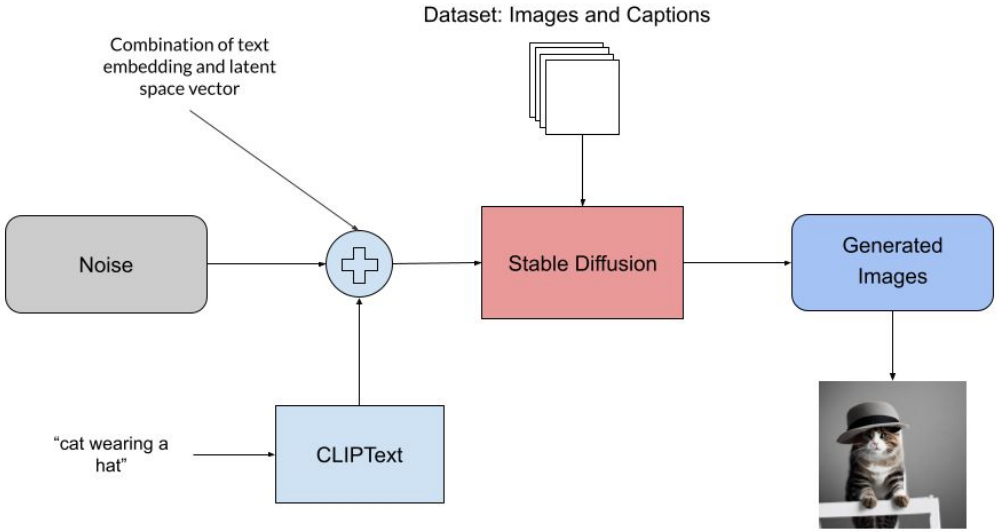
CLIP EMBEDDINGS



We transform text and images into their respective embedding vectors using the CLIP model. These embeddings guide the denoising process

PUTTING IT ALL TOGETHER

- Stable Diffusion is trained on a data set that contains images and their respective captions
- A noise patch is taken as an input which is denoised iteratively
- The denoising process is guided by the text prompt



APPROACHING IMAGE GENERATION MODELS

Here are some considerations to keep in mind:

- We will not go too much into the math behind diffusion models
- Rather, we will understand diffusion models in detail through code
- Thus, we will learn to set up Stable Diffusion pipeline environments in Python
- And we will practice prompting for image models