Stable Diffusion For Image Generation

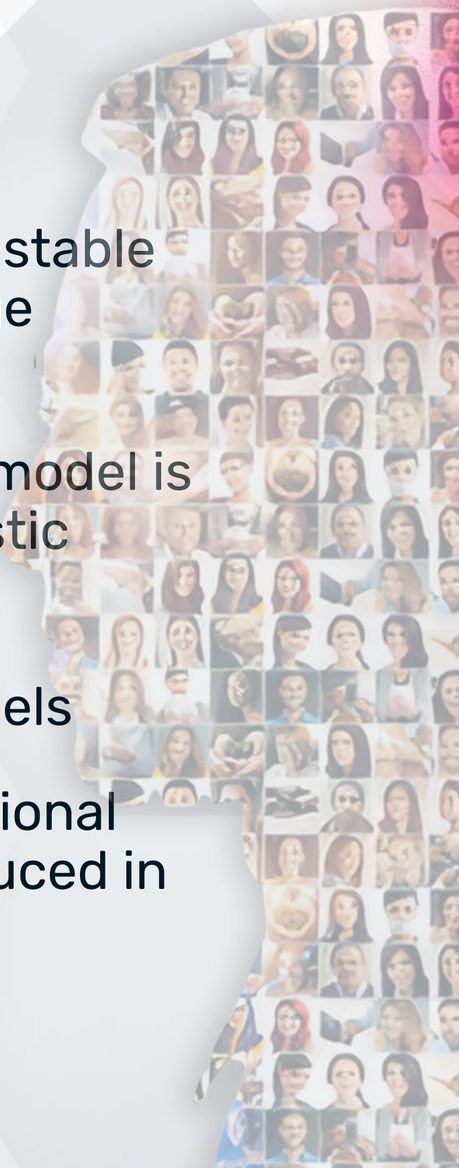
Prepared by

Emad Abd Al Fatah	**2K21/CO/165**
Raad Ghazi	**2K21/CO/360**
Yaseen Mohammed	**2K21/CO/538**

Course coordinator: **KHUSHBU GUPTA**

Introduction

- In this presentation, we are going to explore the innovative approach of stable diffusion for image generation, a technique that combines text and image generation using deep learning methods
- The math that we are going to use for the development of the diffusion model is based on the mathematical principles outlined in the Diffusion Probabilistic Models (DDPM) paper .
- We will also explore the Forward and Reverse Processes in Diffusion Models
- The classifier free guidance ,U-Net Architecture, CLIP Integration, Variational Autoencoder (VAE) and the image generation process will be also introduced in this presentation.



Latent Diffusion Models

A latent diffusion model (LDM) is a type of deep learning model that can create and alter high-resolution images. It is nothing but a generative model. LDM is a probabilistic model that begins with random noise and gradually transforms it into realistic images. LDM has several advantages, including:

Flexible conditioning

- Text, images, segmentation maps, and other inputs can be encoded into the latent space.

Detail preservation

- The encoder-decoder structure allows for manipulating images while still preserving intricate details from the original inputs.

They are probabilistic models that can generate high-quality images by starting with random noise and gradually transforming them into realistic images through a diffusion process.
The key innovation of latent diffusion models is that they apply this diffusion process not to the raw pixel values of an image but instead to an encoded latent representation of the image.

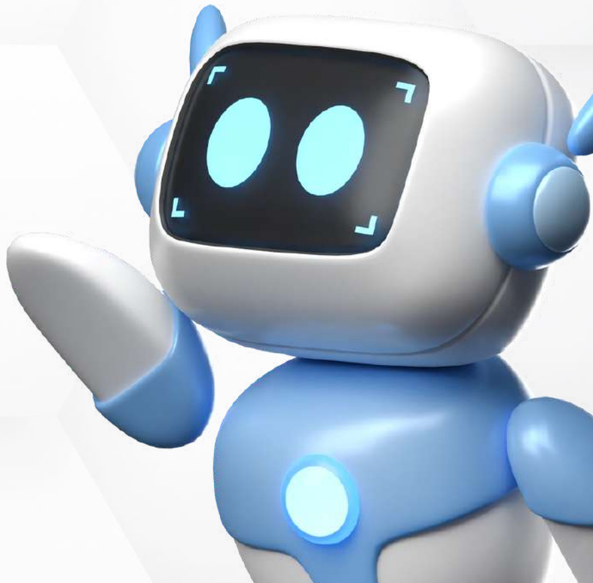
What is a generative model

- Generative modeling is the use of artificial intelligence (AI), statistics and probability in applications to produce a representation or abstraction of observed phenomena or target variables that can be calculated from observations.

- Generative models are pivotal in learning the probability distributions of data, enabling us to sample from these distributions to generate new images

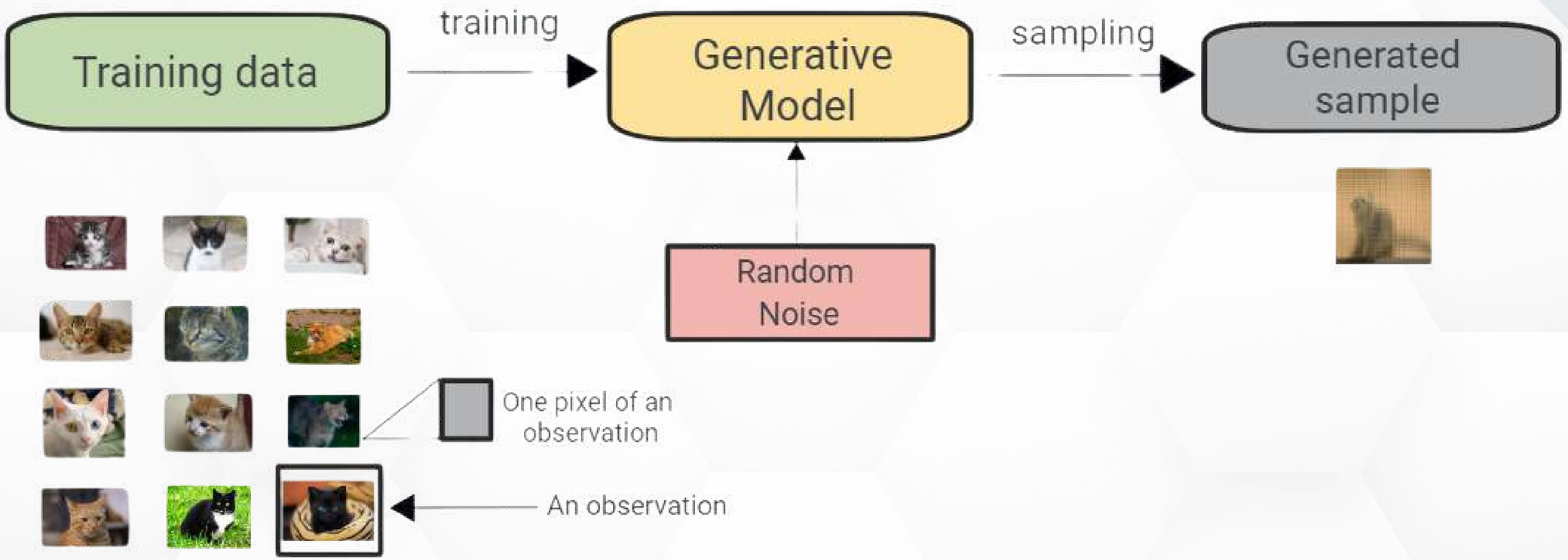
SO how does it work ????

- Generative models are generally run on neural networks. To create a generative model, a large dataset is typically required. The model is trained by feeding it various examples from the data set and adjusting its parameters to better match the distribution of the data.
- Once the model is trained, it can be used to generate new data by sampling from the learned distribution. The generated data can be similar to the original data set, but with some variations or noise.



For Example:

- a data set containing images of cats could be used to build a model that can generate a new image of a cat that has never existed but still looks almost realistic. This is possible because the model has learned the general rules that govern the appearance of a cat.

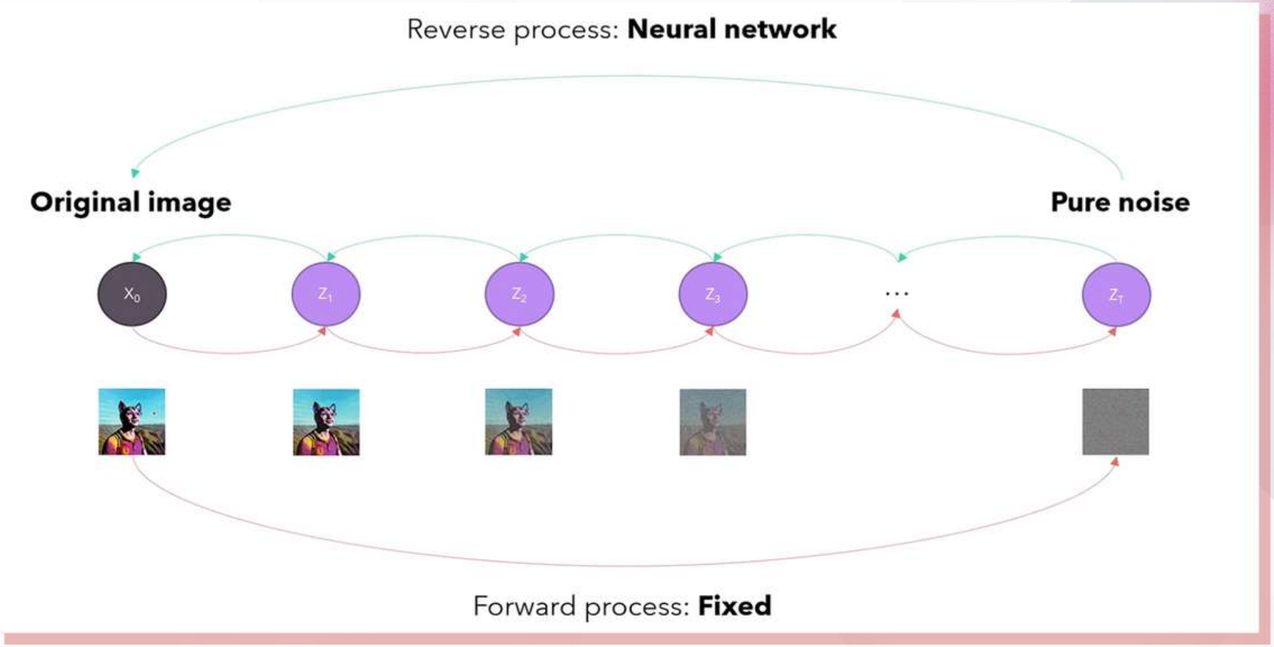


Joint Distribution and Conditional Probability in the generative model

- a generative model is a class of algorithms that use joint probability to predict the distribution of individual classes in a dataset. The joint probability distribution, or P(y, x), models the distribution of each label, and is equivalent to a model of the distribution of label values and the distribution of observations given a label.
- Joint probability distribution encodes the marginal distributions, which are the distributions of each individual random variable. the conditional probability distributions, which describe how the outputs of one random variable are distributed when given information on the outputs of the other random variable.
- generative models learn a probability distribution of the data and we can sample from the data to generate new images. If we have many distributions and we need to sample from a combination of both we will need a joint distribution and we can evaluate the probabilities using conditional probability and by marginalizing a variable . Then we learn the parameters of this distribution and sample from it to generate new data.

Forward and Reverse Processes

- The forward and reverse diffusion processes are the core components of diffusion models, which define how data is transformed into noise and then how the noise is transformed back into data. The forward diffusion process gradually turns an image into noise, while the reverse diffusion process turns that noise back into the image.



- In the forward diffusion process, Gaussian noise is added step by step to the original data until it becomes completely random at the final step. This creates a Markov chain starting from the original data and ending with a fully noised sample. In contrast, the reverse diffusion process trains the model to progressively remove noise from fully noised data, learning to reconstruct the original information. These trained models can then generate new data by inputting random noise and applying the learned denoising process to produce outputs resembling the original data distribution.

Training and Loss Functions

- So in diffusion model process in simple ,We take the input image x0x0 and gradually add Gaussian noise to it through a series of T steps. We will call this the forward process.
- Afterward, a neural network is trained to recover the original data by reversing the noising process. By being able to model the reverse process, we can generate new data. This is the so-called reverse diffusion process or, in general, the sampling process of a generative model.
- In diffusion models, after completing both the forward and reverse processes, we obtain the Evidence Lower Bound (ELBO) and define the loss function

- The **ELBO** is derived from the probability distribution of the original data and the noised data after the reverse process. It represents a lower bound on the log-likelihood of the data given the model parameters. The ELBO is calculated using probabilistic inference techniques such as variational inference. therefore if we maximize the lower bound we will also maximize the likelihood.

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] =: L$$

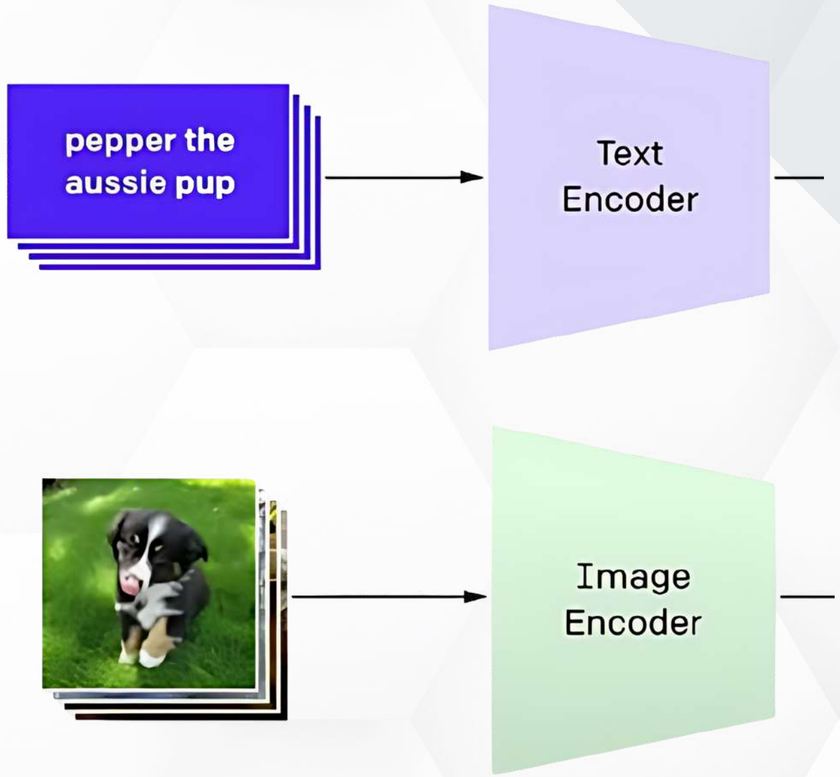
- The **loss function** in diffusion models quantifies the discrepancy between the original data and the reconstructed data after the reverse process. It typically includes terms to penalize differences in pixel values and encourage the model to learn accurate denoising and reconstruction processes. The loss function is optimized during training using techniques like stochastic gradient descent (SGD) to minimize the reconstruction error and improve model performance.

Classifier-Free Guidance

- Classifier-free guidance refers to a technique where the model generates data without relying on explicit classifiers but rather uses contextual cues or prompts to guide the generation process. (relies on prompt or context signal).

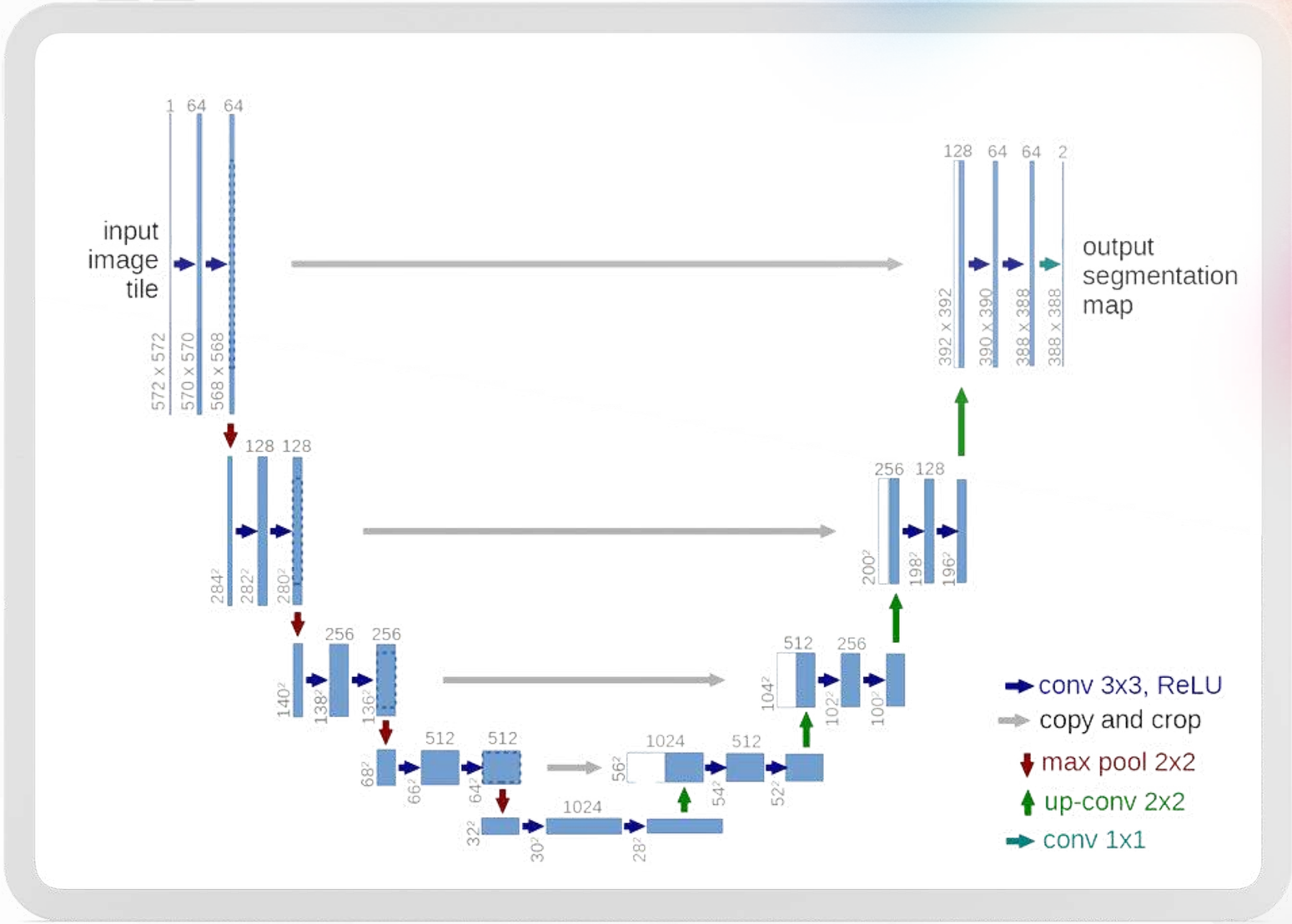
- This approach helps the model generate more diverse and contextually relevant outputs by leveraging the information provided in the prompts

- therefore We will Utilize classifier-free guidance by incorporating prompts or contextual signals to influence the generation process, enhancing the quality and relevance of generated outputs.



U-Net Architecture

The U-Net architecture is a popular convolutional neural network (CNN) architecture commonly used in image processing tasks such as image generation and segmentation. It's particularly well-known for its effectiveness in biomedical image segmentation, but it's also widely applicable in various other image-related tasks.



U-Net components

- Encoder: Reduces image size, increases feature channels for feature extraction.
- Decoder: Upsamples feature maps, reconstructs spatial information.
- Skip Connections: Connects encoder and decoder, preserves spatial information for precise segmentation.
- Prompts: Instructions for image generation, specifying desired features.
- Noise Levels: Control random variation, enhance diversity and realism.
- Outputs: Synthetic images or segmentation maps, depending on the task.

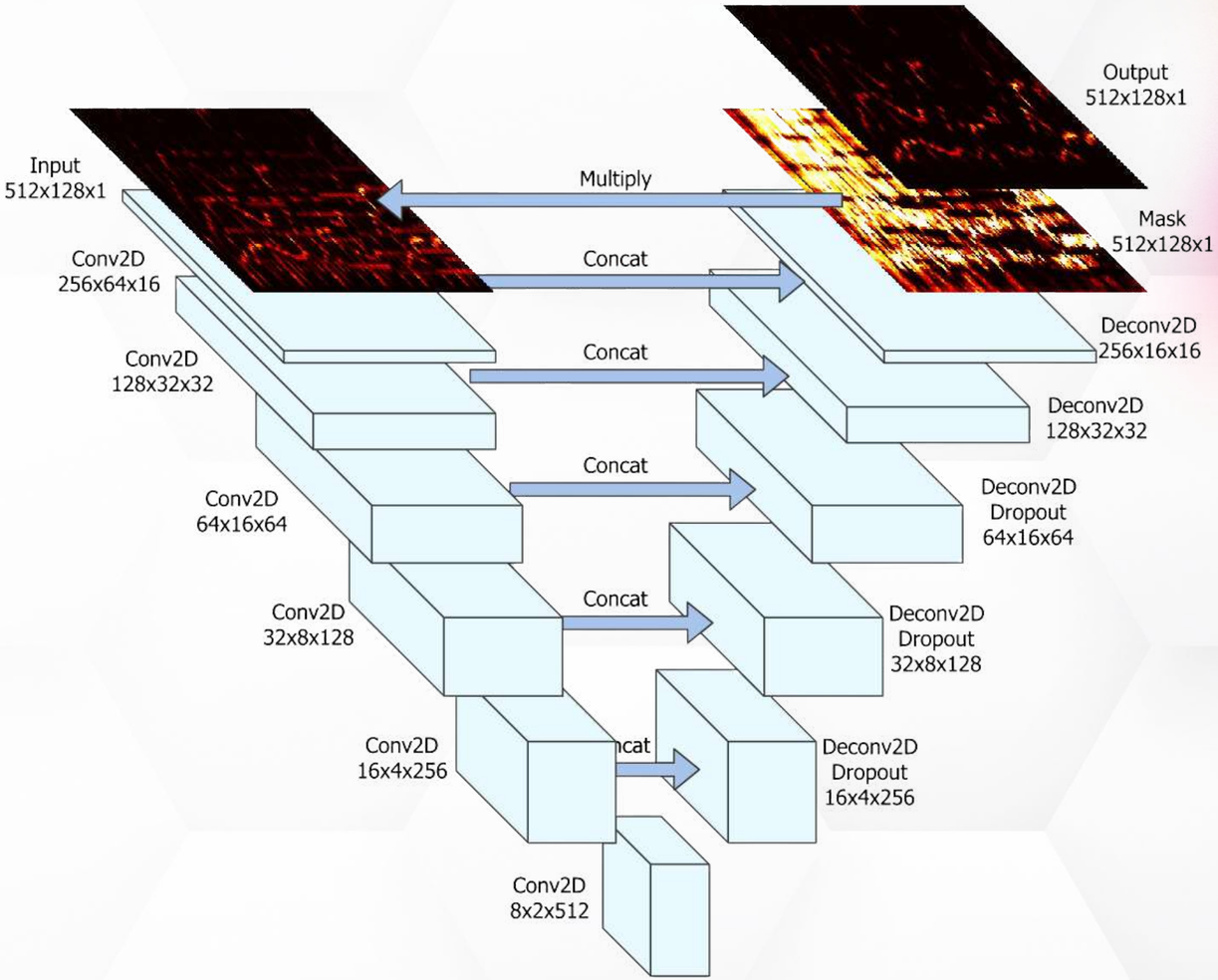
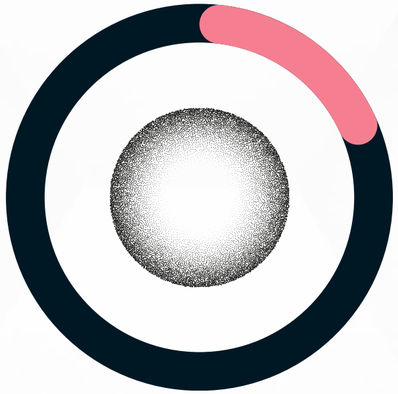
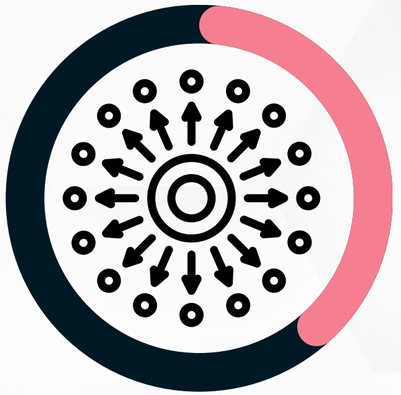


Image Generation Process Steps

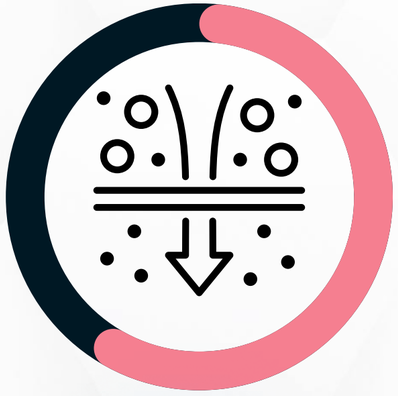
Initialize Data



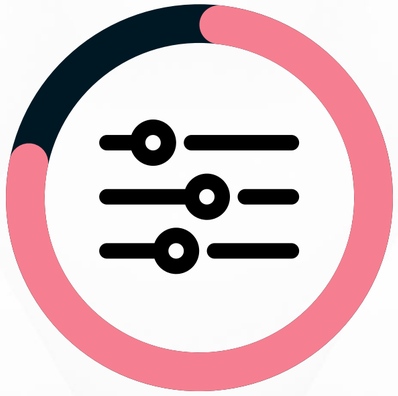
Add Noise



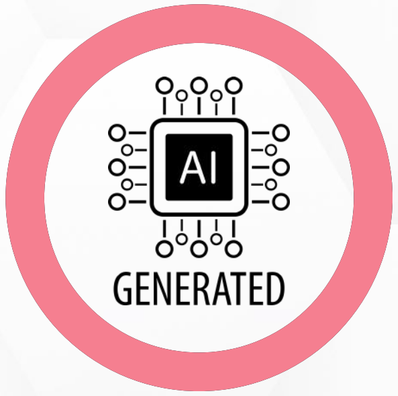
Forward Diffusion Process



Reverse Diffusion Process



Adjust Parameters



Evaluate Generated Data

Image Generation Process Steps

- Initialize Data: Start with a dataset containing the original clean data you want to generate variations of.
- Add Noise: Introduce controlled noise to the clean data. This noise can be in various forms, such as Gaussian noise or structured noise patterns. The level of noise added can be adjusted based on the desired diversity in the generated data.
- Forward Diffusion Process: In the forward diffusion process, iteratively degrade the quality of the data by diffusing it through multiple steps. This process involves gradually increasing the level of noise added to the data, simulating a diffusion or degradation process over time.
- Reverse Diffusion Process: After reaching a certain level of degradation or diffusion, perform a reverse diffusion process. This involves iteratively removing the noise added during the forward diffusion process, gradually restoring the data back to its original or a desired state.
- Adjust Parameters: Throughout the diffusion and reverse diffusion processes, you can adjust parameters such as the diffusion step size, the intensity of added noise, and the number of diffusion steps to control the characteristics of the generated data.
- Evaluate Generated Data: Finally, evaluate the generated data to ensure that it meets the desired criteria in terms of diversity, realism, and fidelity to the original clean data. This evaluation can involve qualitative assessment by visual inspection as well as quantitative measures to compare the generated data with the original dataset.

How to condition the reverse process?

- The latest and most successful approach is called classifier-free guidance, in which, instead of training two networks, one conditional network and an unconditional network, we train a single network and during training, with some probability, we set the conditional signal to zero, this way the network becomes a mix of conditioned and unconditioned network, and we can take the conditioned and unconditioned output and combine them with a weight that indicates how much we want the network to pay attention to the conditioning signal.

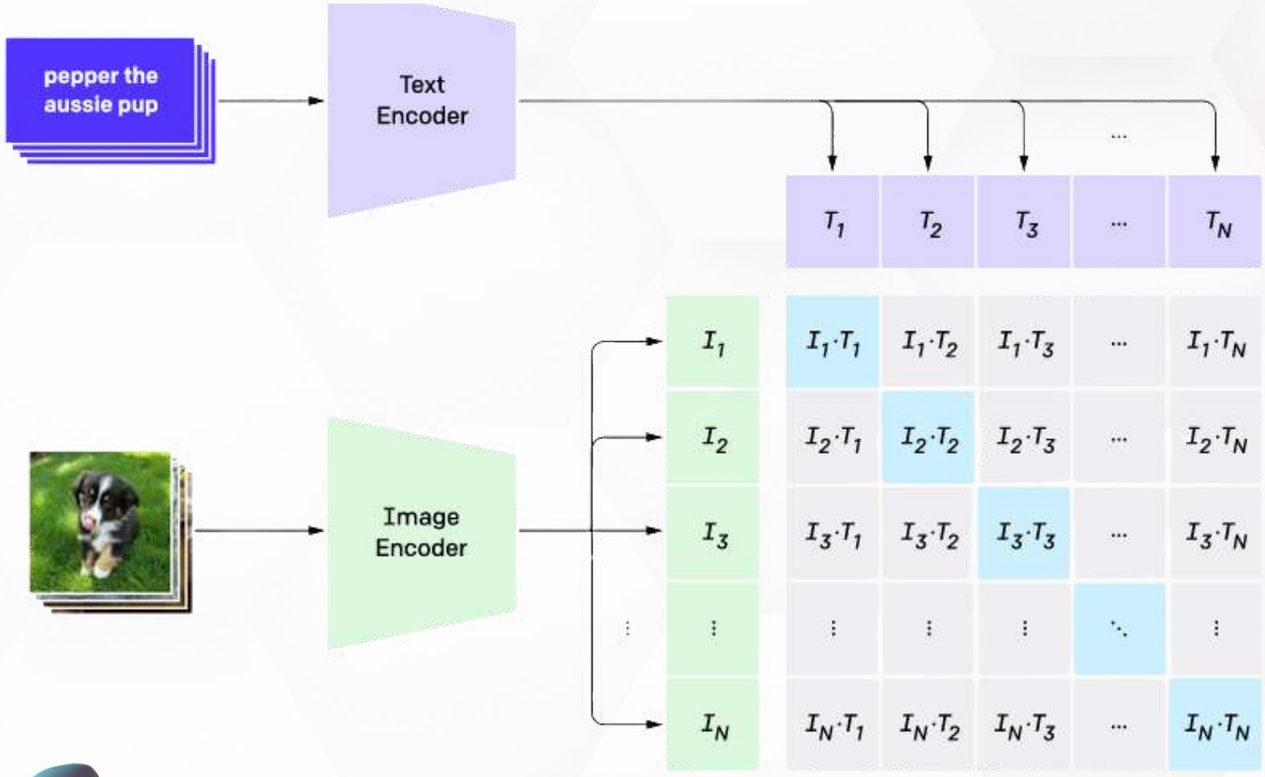
Classifier Free Guidance (Combine output)

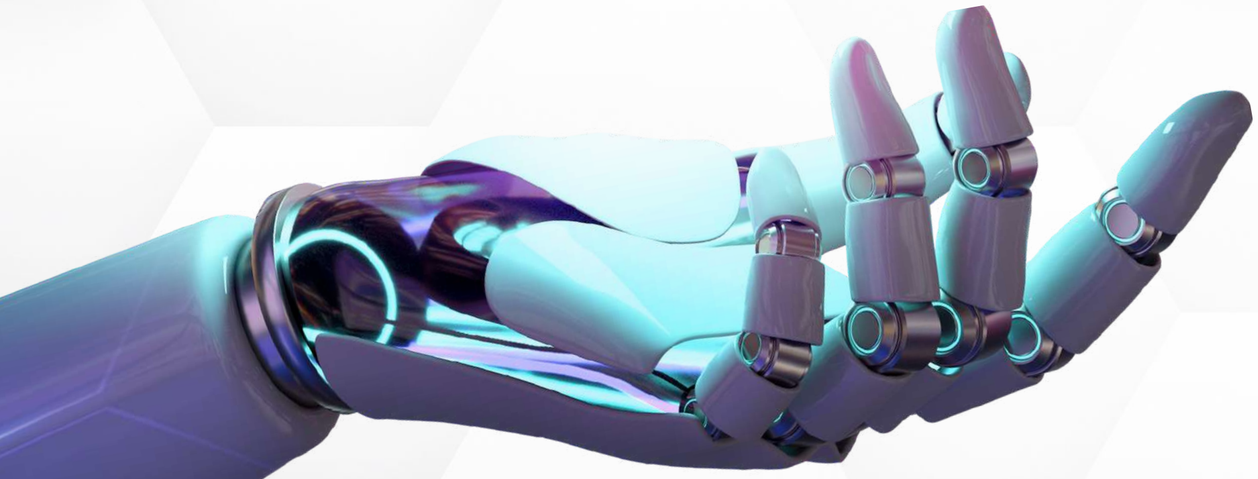
$$output = w * (output_{conditioned} - output_{unconditioned}) + output_{unconditioned}$$

A weight that indicates how much we want the model to pay attention to the conditioning signal (prompt).

Integration of CLIP for Image Generation

CLIP (Contrastive Language for Image Pretraining) integration enhances our model's capabilities. By leveraging prompts and images during training and generation, we achieve a deeper understanding of contextual cues for image creation.



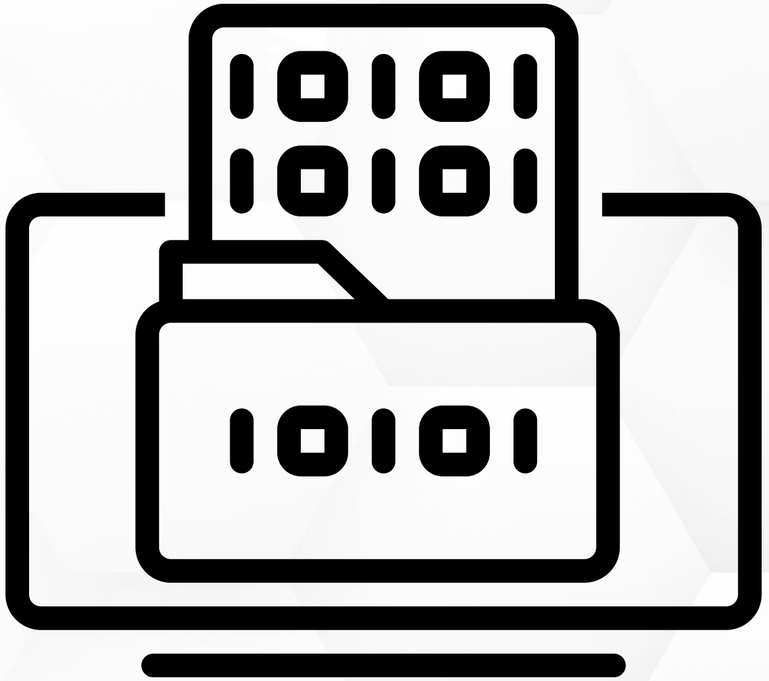


Variational Autoencoder (VAE)

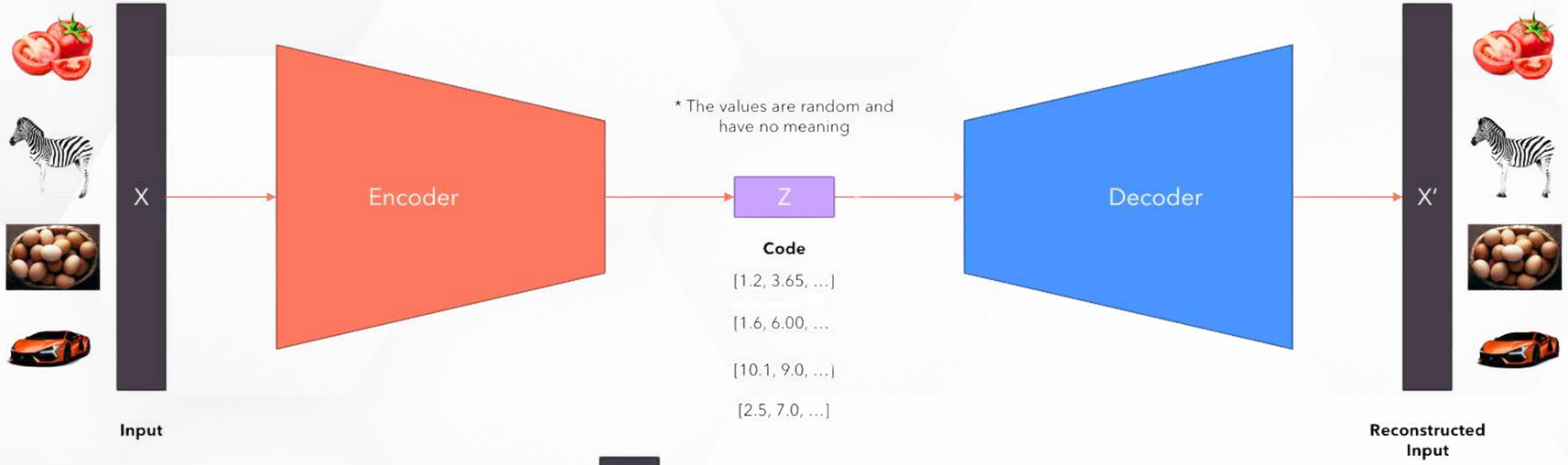
- ## Enhancing Semantic Relationships with VAE:

Variational Autoencoder (VAE) improves semantic relationships compared to traditional autoencoders.

- Stable Diffusion is a latent diffusion model, in which we don’t learn the distribution p(x) of our data set of images, but rather, the distribution of a latent representation of our data by using a Variational Autoencoder.
- This allows us to reduce the computation we need to perform the steps needed to generate a sample, because each data will not be represented by a 512x512 image, but its latent representation, which is 64x64.



Variational Autoencoder (VAE)



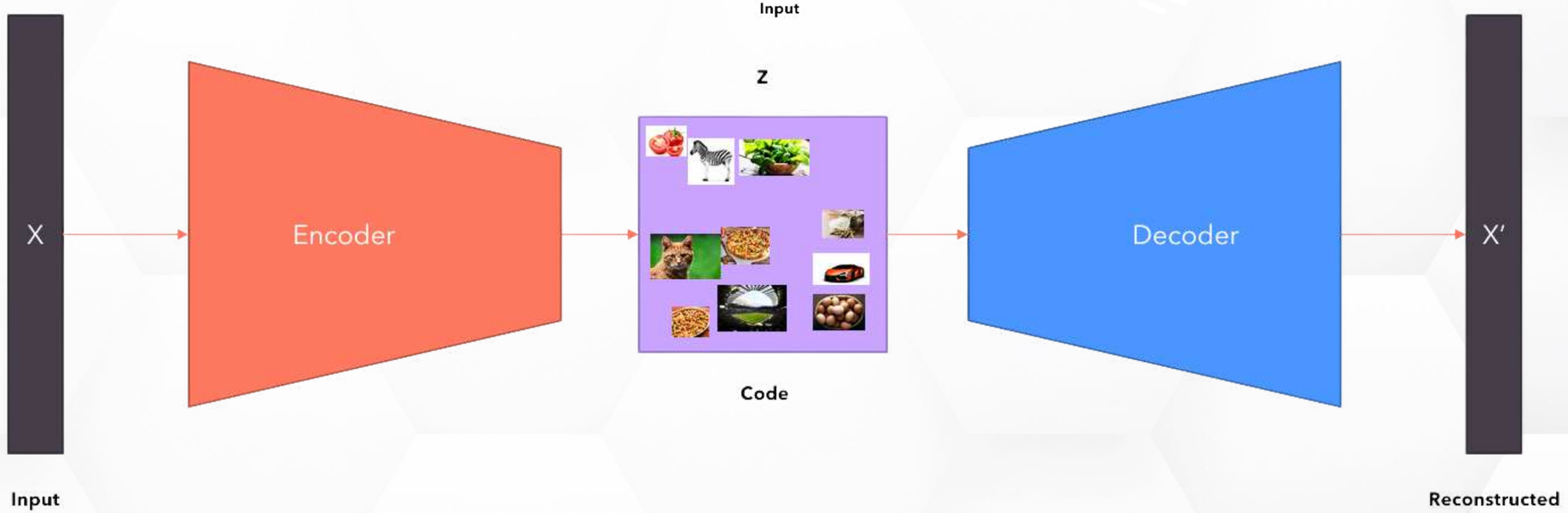
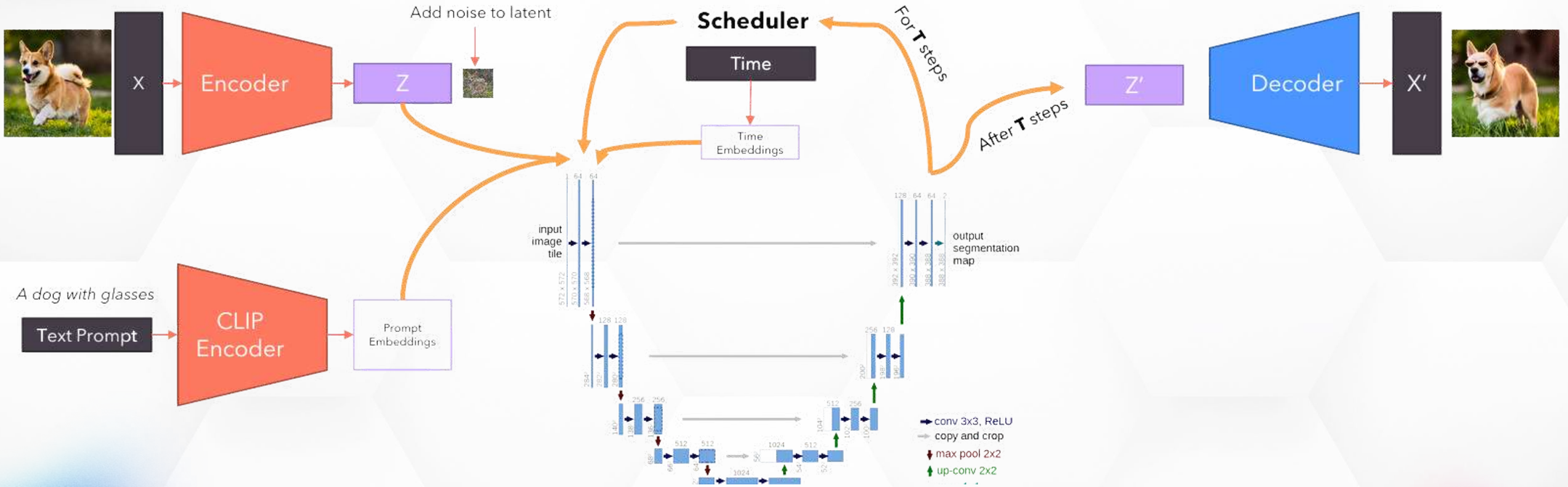


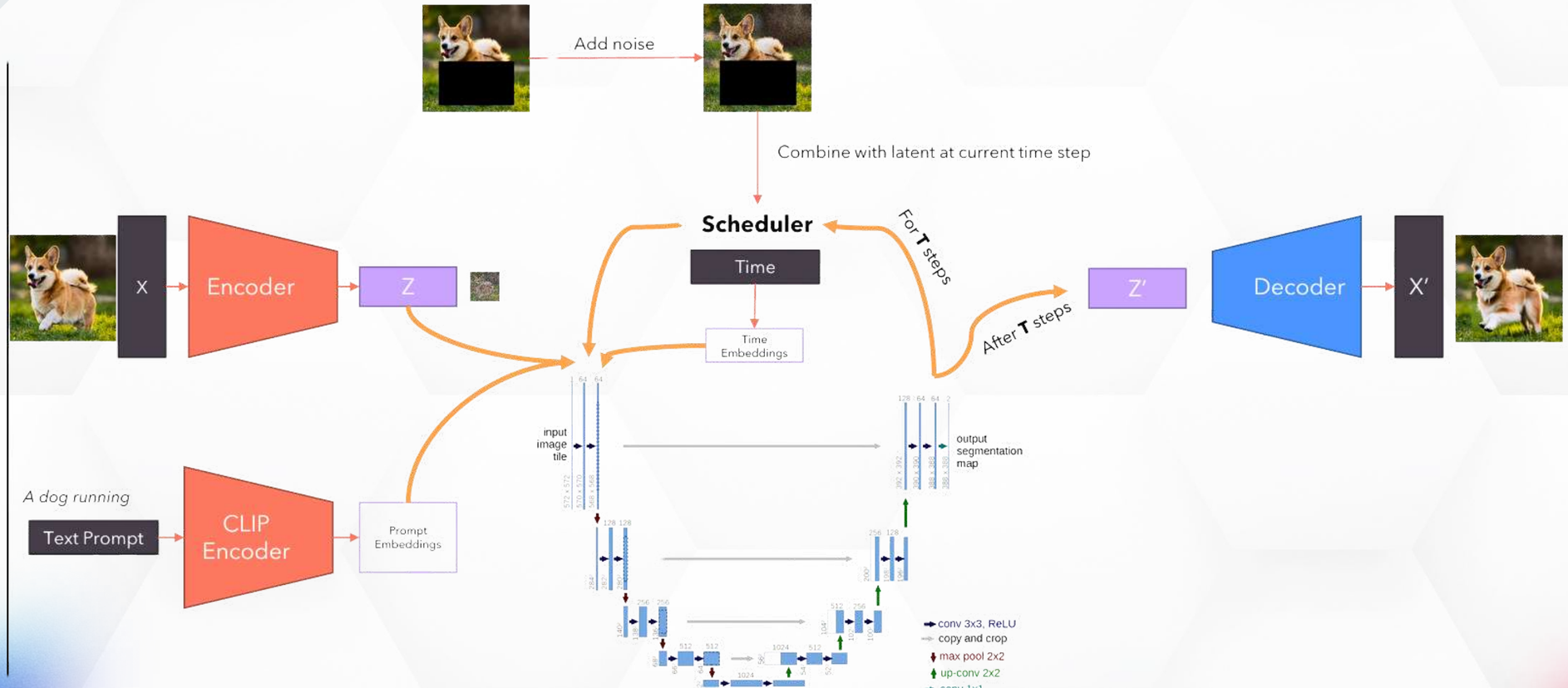
Image to Image Generation

Diffusion models excel in diverse image generation tasks, transitioning seamlessly from text-to-image to image-to-image architectures. Their versatility lies in their ability to capture intricate spatial relationships and generate high-fidelity outputs across multiple domains.



Inpainting for Image Completion

Inpainting, a key aspect of our model, focuses on completing missing parts in images. By replacing selected parts based on prompts and original data, we achieve comprehensive and visually appealing image completion.



Implementation and Code

Our implementation leverages PyTorch and CLIP. Key components include decoder, encoder, diffusion, attention model, and CLIP model. **The pipeline:** dictates prompt scale importance, with additional features like DDPM scheduler and weights loader enhancing performance.

Attention Model File: This file implements the attention mechanism, specifically utilizing softmax for weighting. This mechanism assigns importance to different parts of the input, aiding in capturing dependencies and relationships crucial for the model's performance.

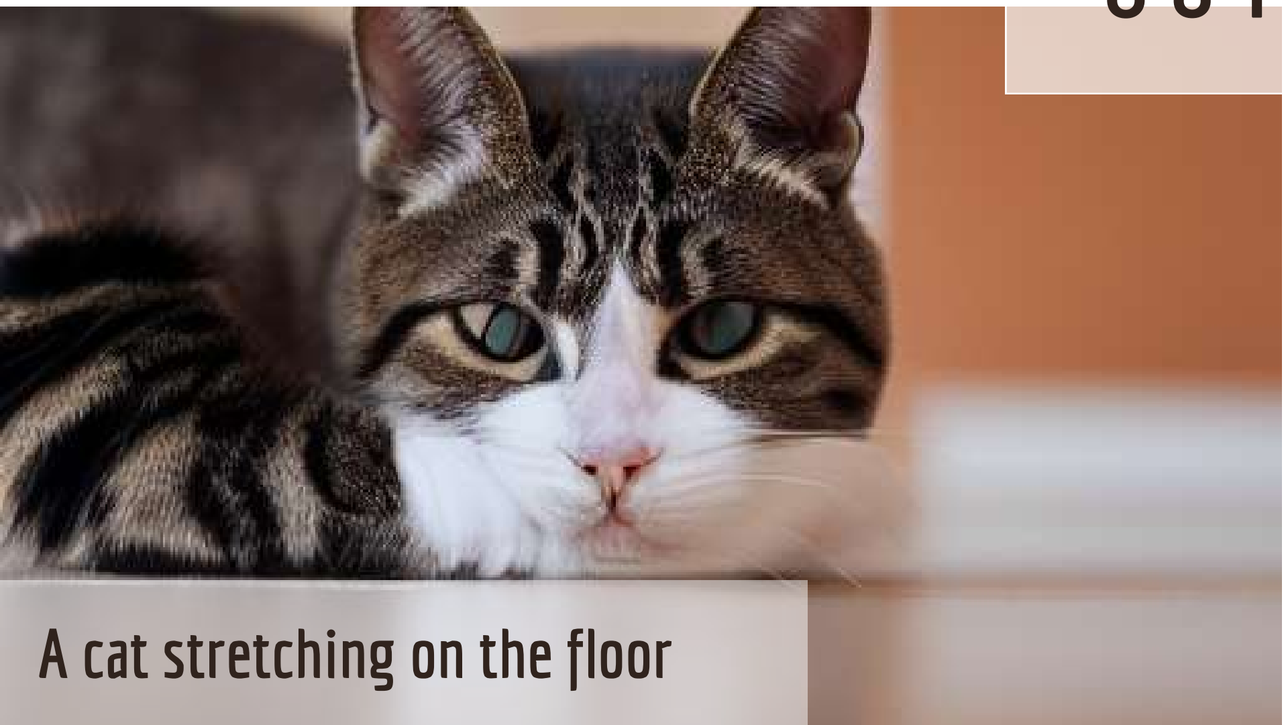
CLIP Model File: Within this file lies the implementation of the Contrastive Language-Image Pre-training (CLIP) model. CLIP is a neural network model that learns to associate images and text by maximizing agreement between image and text representations across a large dataset. It encapsulates a powerful fusion of vision and language understanding, enabling the model to understand and generate images based on textual prompts.

Demo Visualization File:
This file showcases image generation from text prompts, utilizing encoder, decoder, diffusion, attention, and CLIP models.

A dog wearing glasses

a black camera

OUTPUT



A cat stretching on the floor

A boy playing football

Thankyou