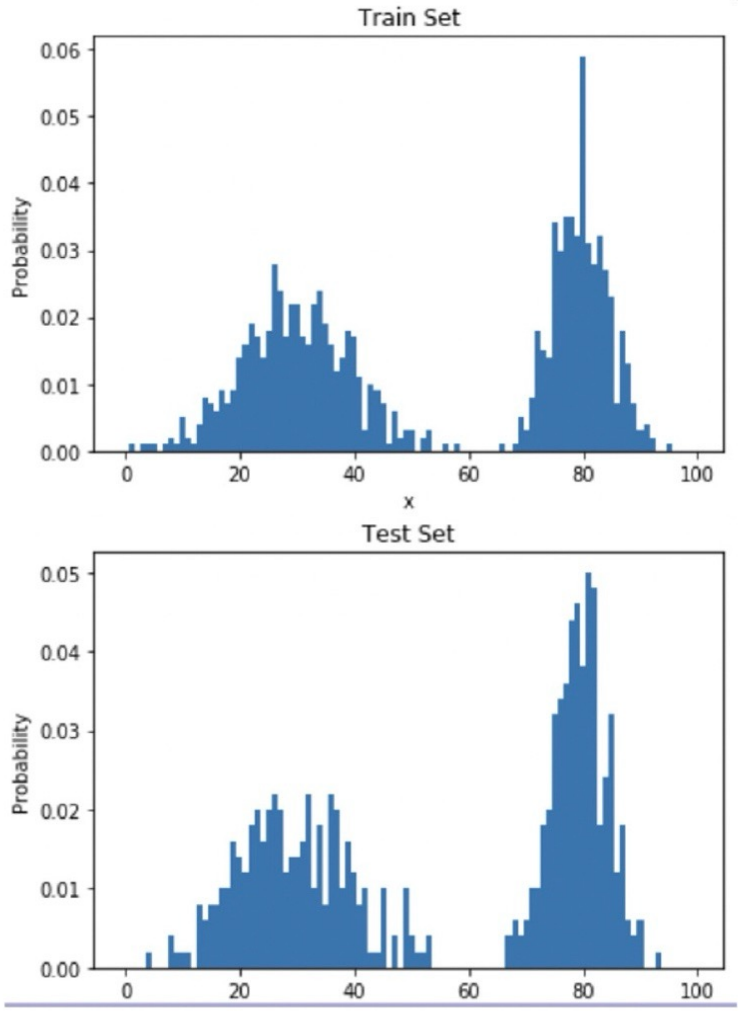
PyTorch Project

Muhammad Yousif

Recall: Data Synthesis

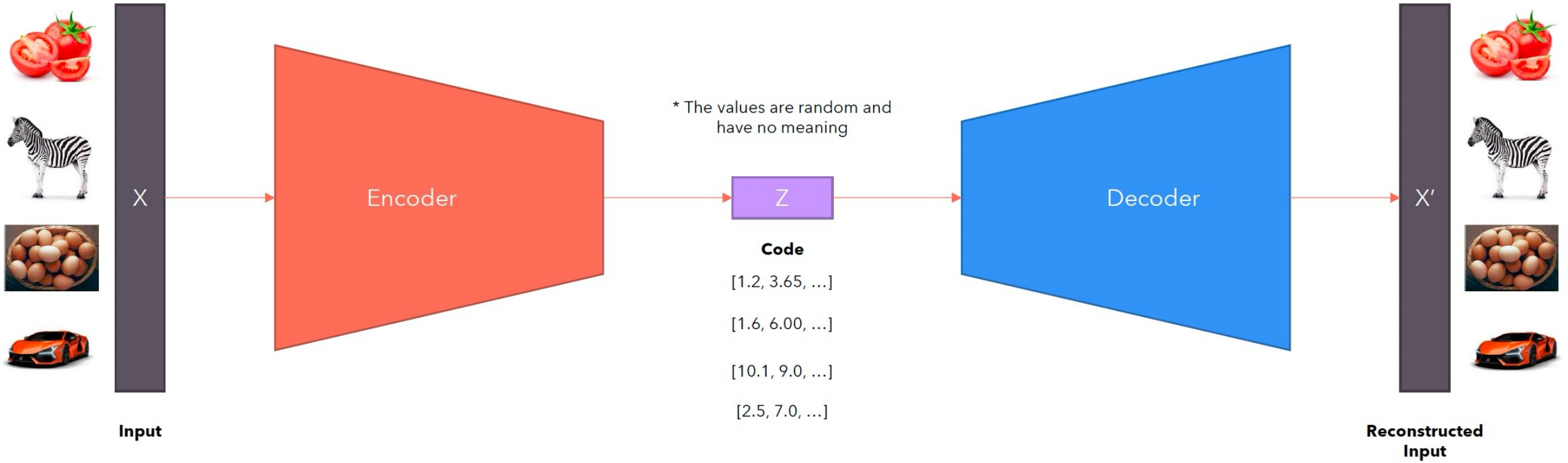
Lets say we have some training set of data following some distribution $p_{data}(x)$:



The goal of generative machine learning models is to *learn* this distribution to the best of their ability

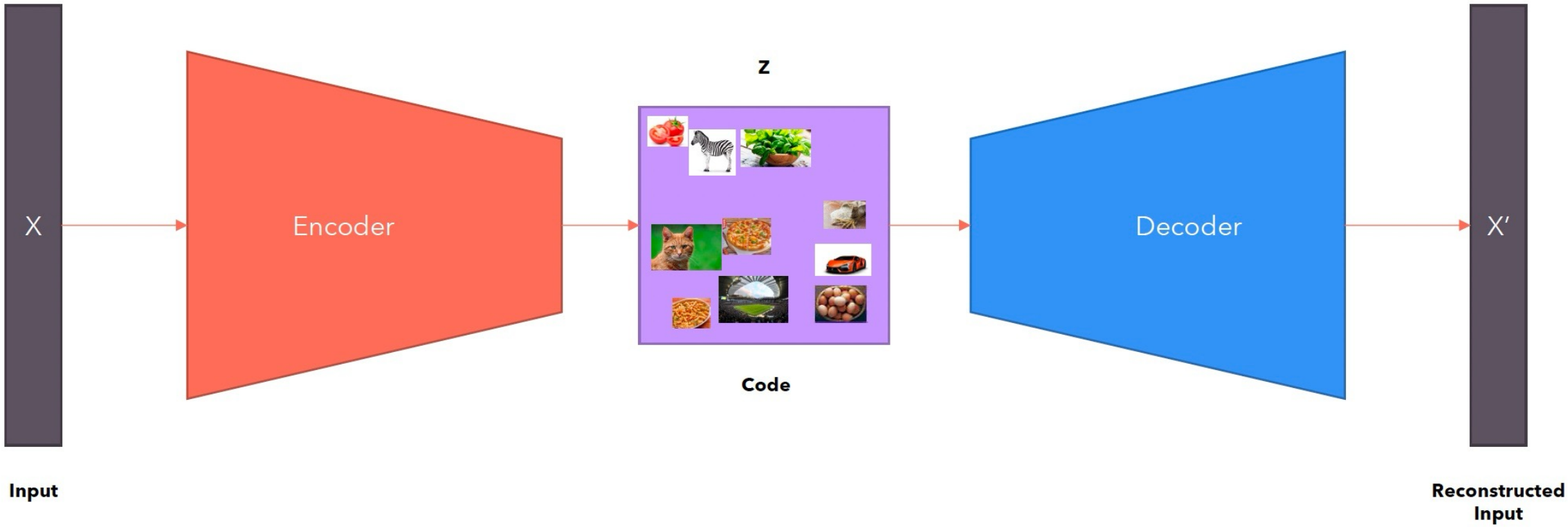
We generate new data by *sampling* from the learned distribution

What is an Autoencoder?



What's the problem with Autoencoders?

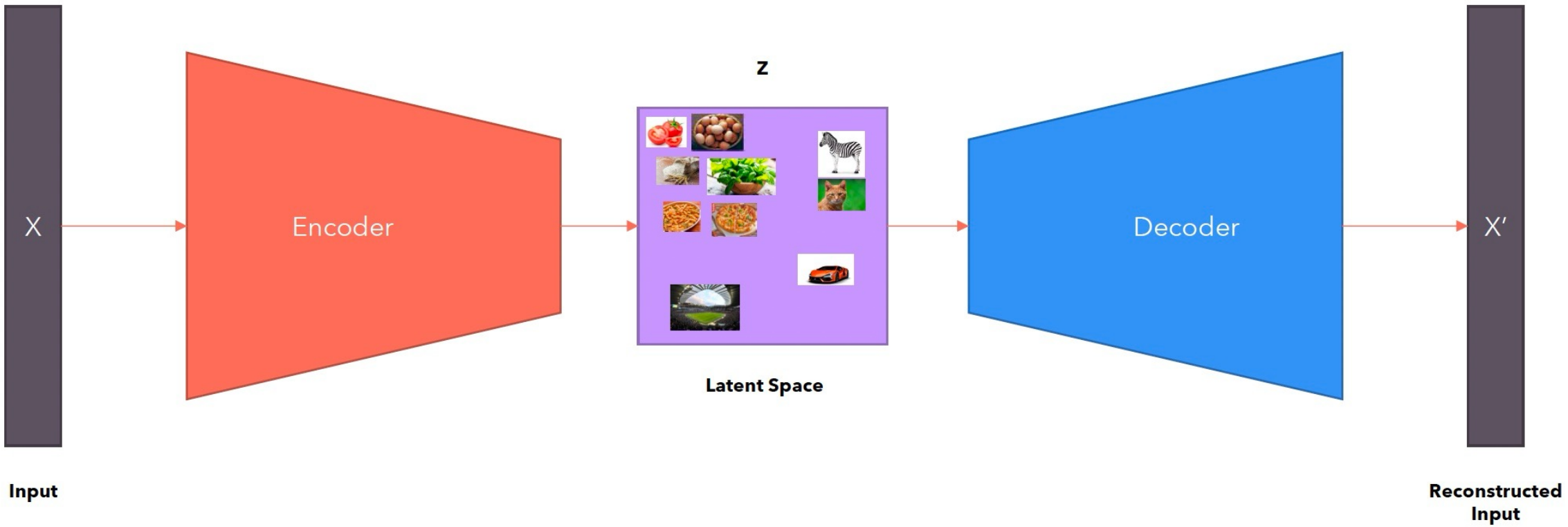
The code learned by the model **makes no sense**. That is, the model can just assign any vector to the inputs without the numbers in the vector representing any pattern. The model doesn't capture any **semantic relationship** between the data.



Umar Jamil - https://github.com/hkproj/pytorch-ddpm

Introducing the Variational Autoencoder

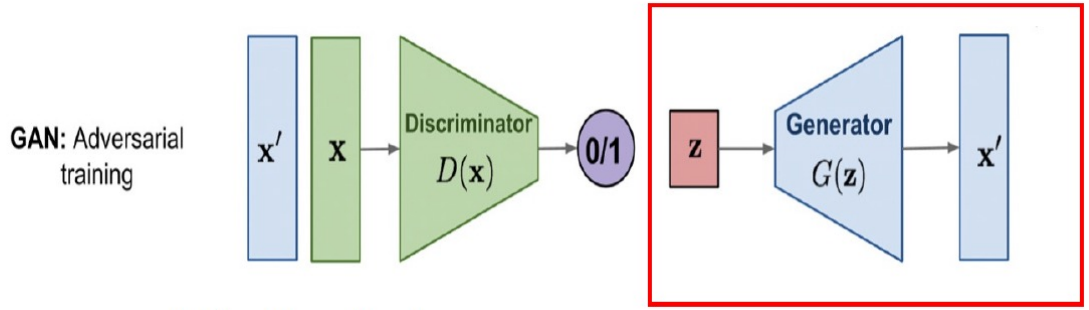
The variational autoencoder, instead of learning a code, learns a "**latent space**". The latent space represents the parameters of a (multivariate) distribution.



Umar Jamil - https://github.com/hkproj/pytorch-ddpm

GANs

- Convert latent noise vector to target distribution in one step

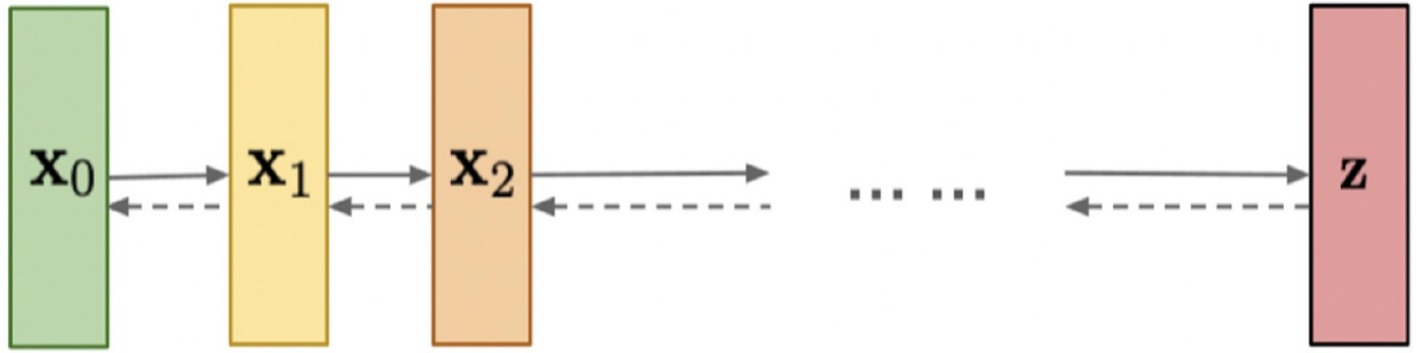


- Lots of issues with training:
  - Vanishing gradients: if discriminator is too good, gradients go to zero
  - Mode collapse: if the generator learns to generate an exceptionally plausible output, it will just continue generating it. Then the discriminator will learn to always reject it, and then the generator will produce the same outputs, which the discriminator will then reject… bad loop!

Diffusion

- Idea: Estimating and analyzing small step sizes is more tractable/easier than a single step from random noise to the learned distribution
- Convert a well-known and simple *base distribution* (like a Gaussian) to the *target (data) distribution* iteratively, with small step sizes, via a Markov chain:

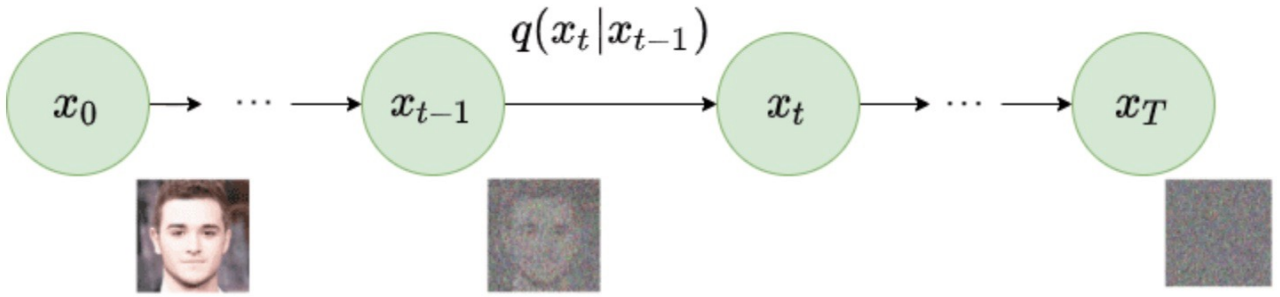
Diffusion models:
Gradually add Gaussian noise and then reverse



- Markov chain: outlines the probability associated with a sequence of events occurring based on the state in the previous event.

Forward Process

- You can prove with some math that as T approaches infinity, you eventually end up with an Isotropic Gaussian (i.e. pure random noise)
- *1. Sample an image from the dataset:*
- *2. Sample noise % & ' () * (from a standard normal distribution)*



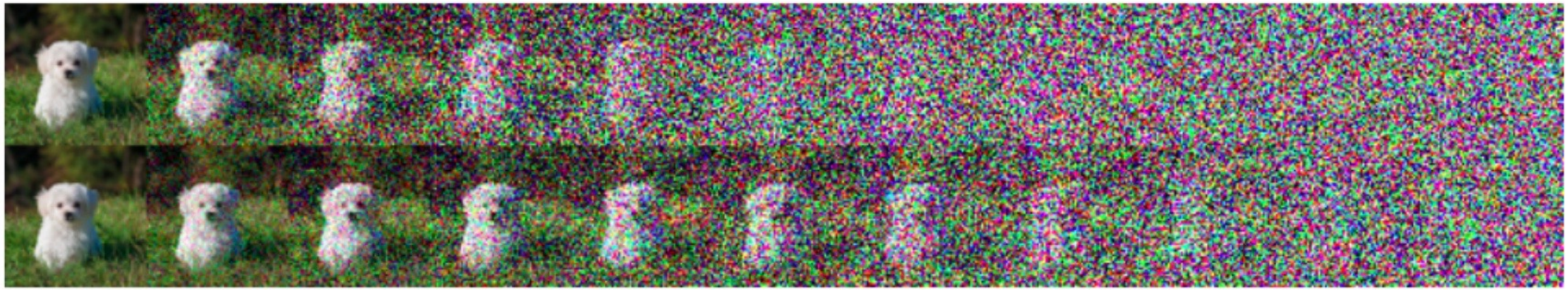
Forward diffusion process. Image modified by Ho et al. 2020

- Note: forward process is fixed

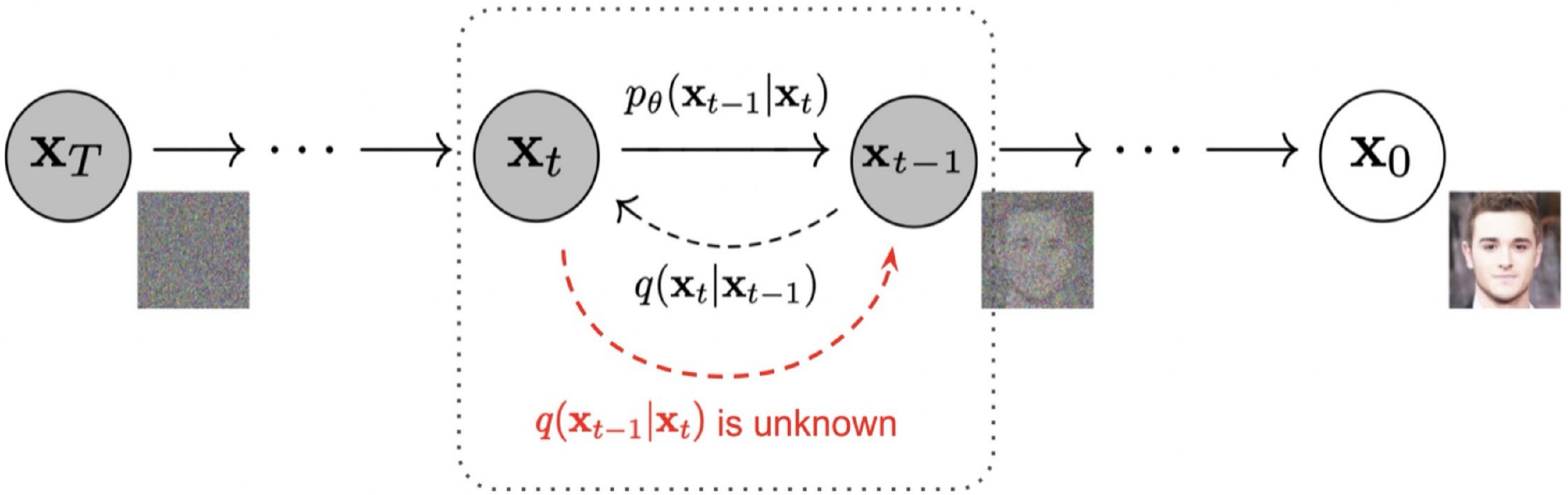
Schedule

- Linear: not that efficient as most of the information from the original image is lost after around half of the total steps.

- Cosine (OpenAI 2021)



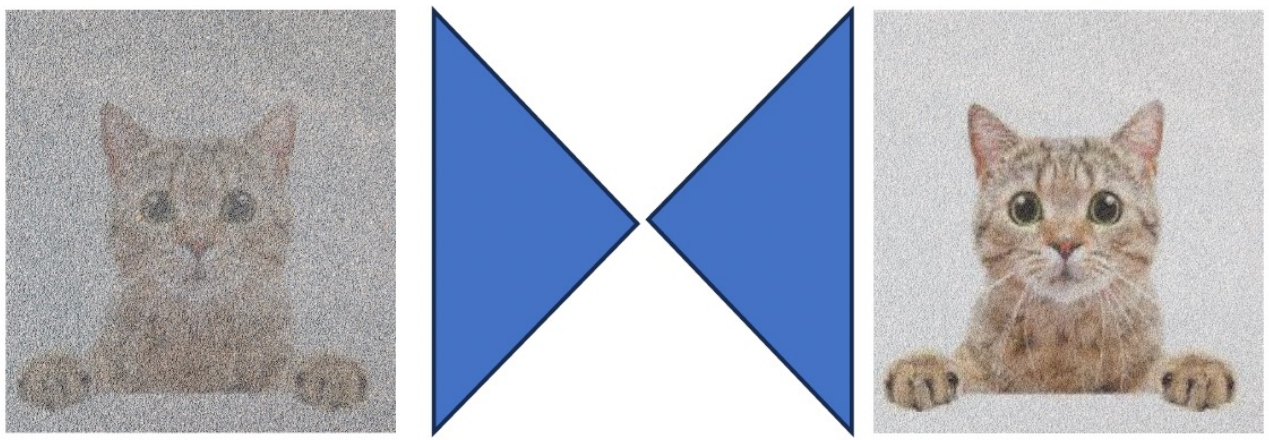
Reverse Process



- The goal of a diffusion model is to **learn** the reverse *denoising* process to iteratively **undo** the forward process
- In this way, the reverse process appears as if it is generating new data from random noise!

How do we do this in practice?

x_{t+1} $\hat{x}_t$

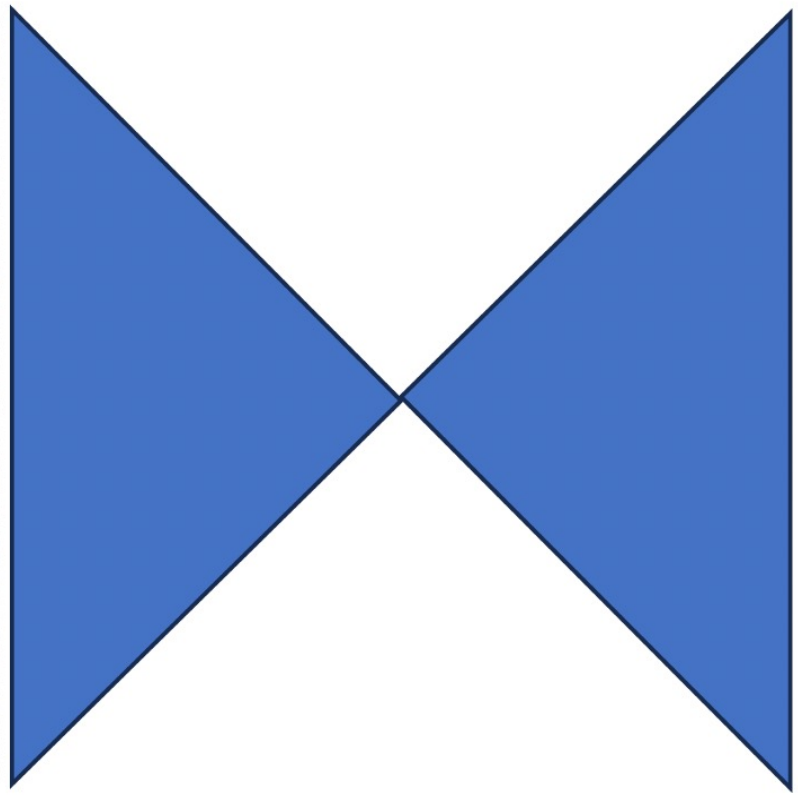


Loss: MSE$(x_t, \hat{x}_t)$

Neural Network that predicts noise

Input

U-net



Output

Diffusion Models Beats GANs

BigGAN

Diffusion

Training Set

Impressive Results

DALL·E 2

"a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese"

IMAGEN

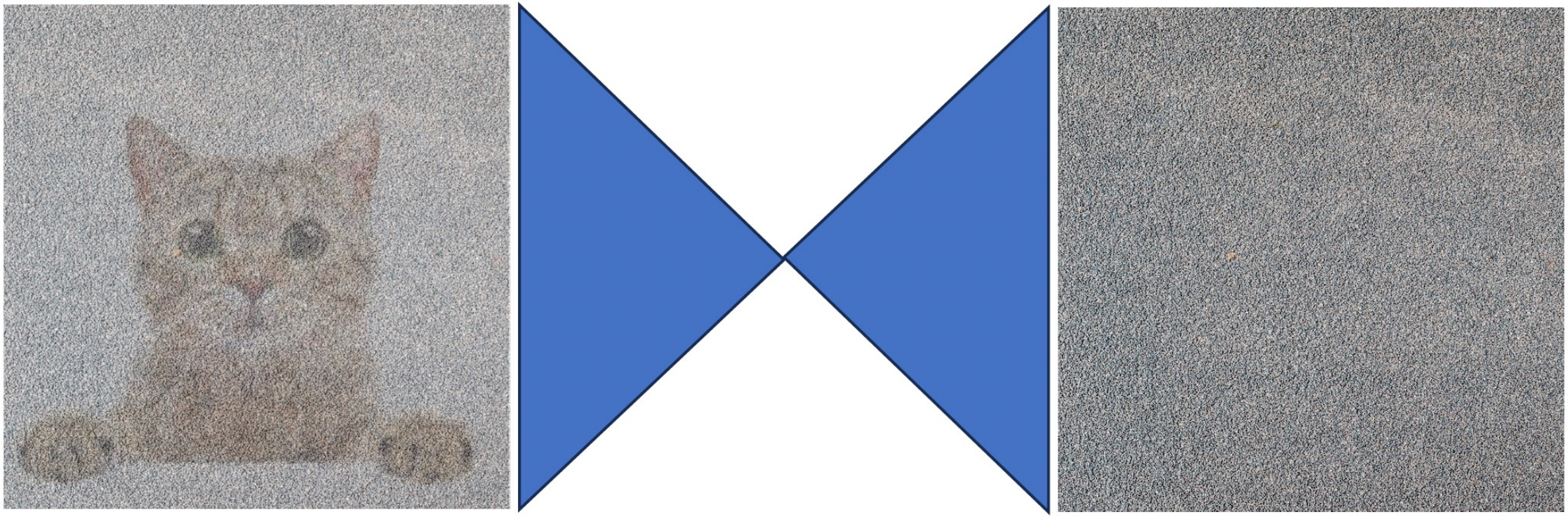
"A photo of a raccoon wearing an astronaut helmet, looking out of the window at night."

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.
Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Slide Credit: CVPR 2023 Diffusion Models Tutorial

U-net Problem

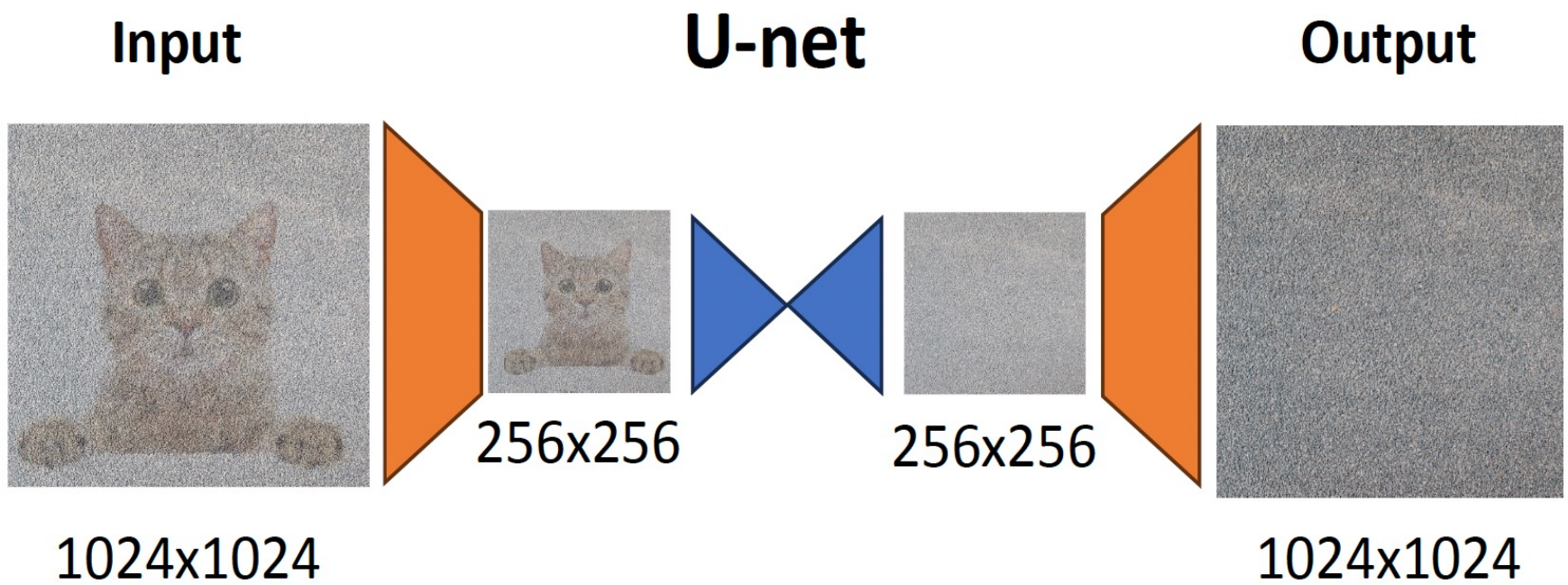
Input **U-net** **Output**



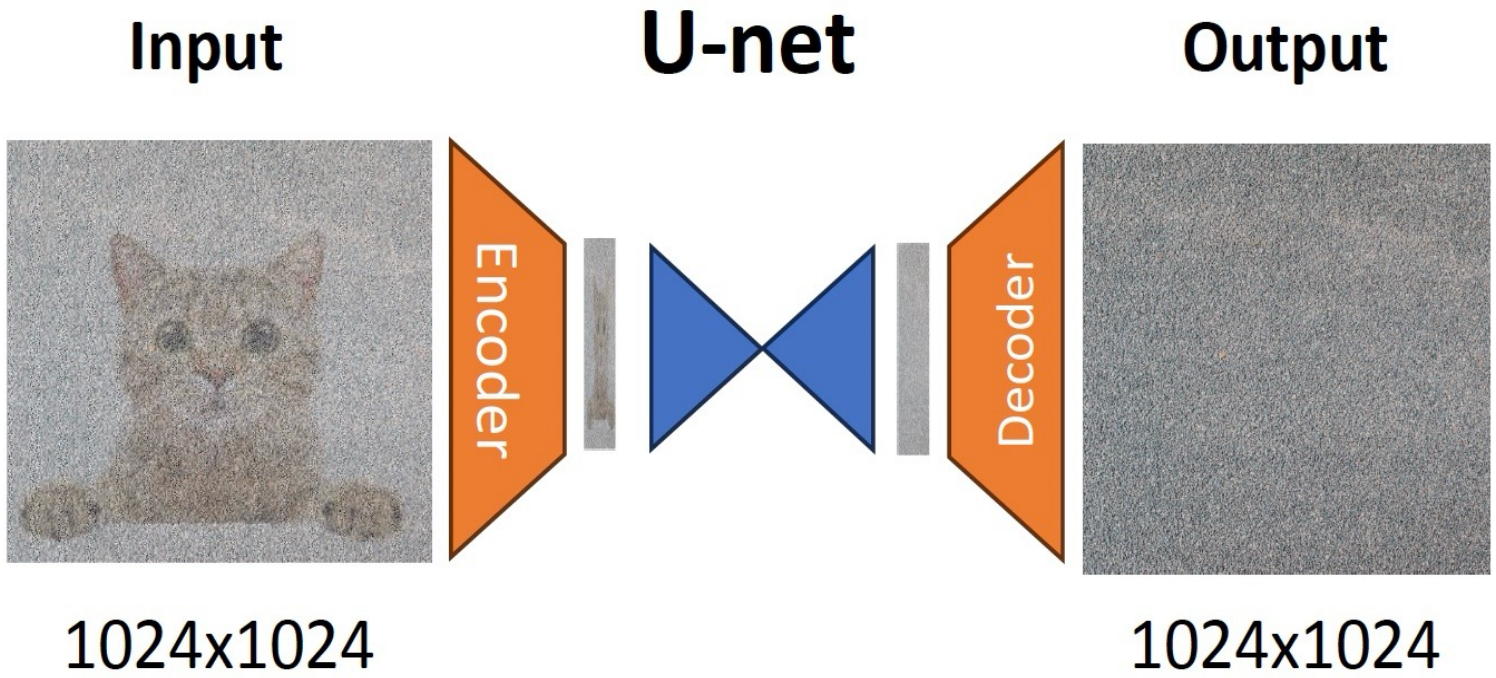
1024x1024 1024x1024

Problem: operating in the input space is very computationally expensive!

Option #1: Generate Low-Resolution + Upsample



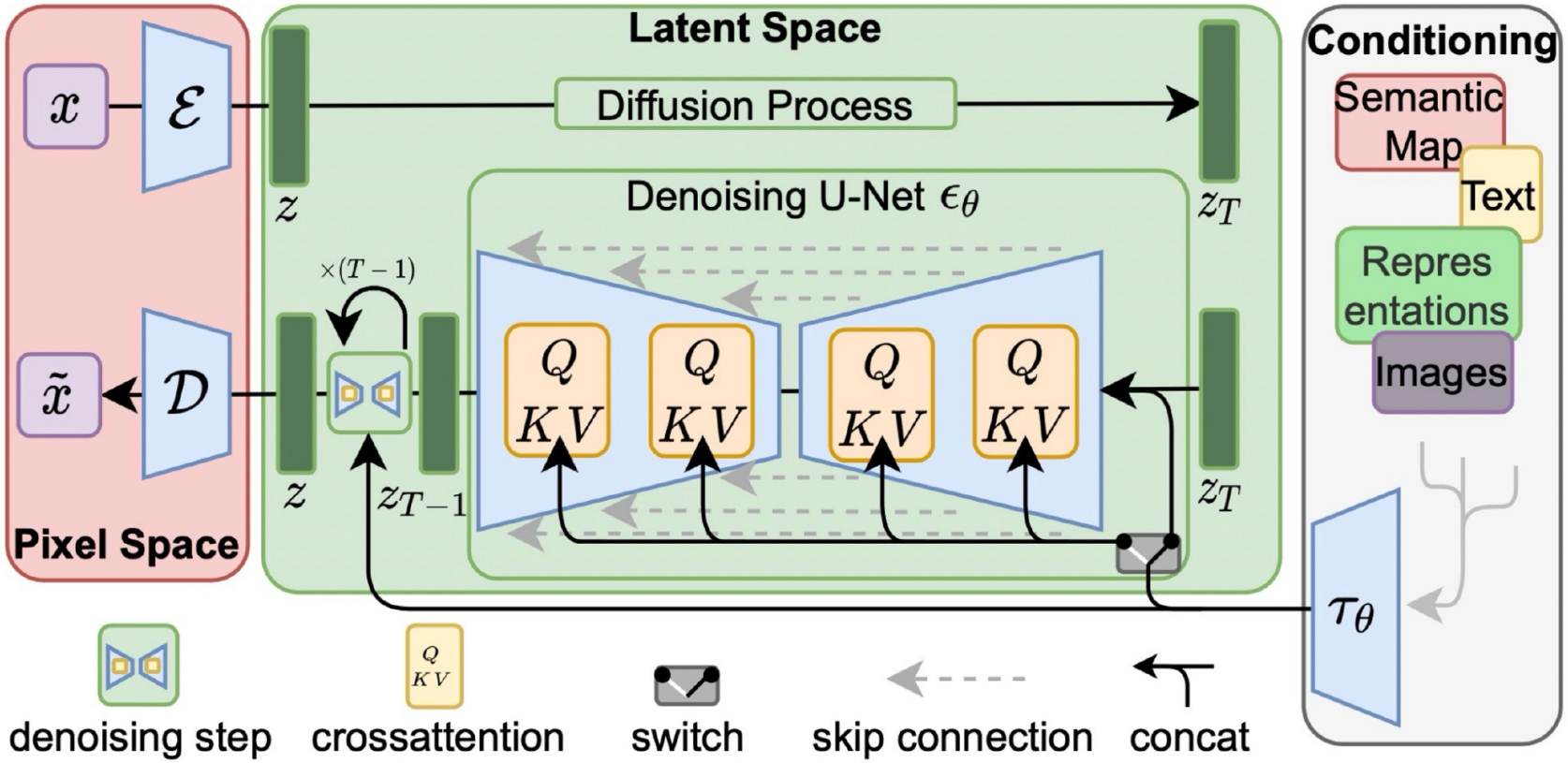
Option #2: Generate in Latent Space



https://huggingface.co/spaces/stabilityai/stable-diffusion

Stable Diffusion

What's going on here?

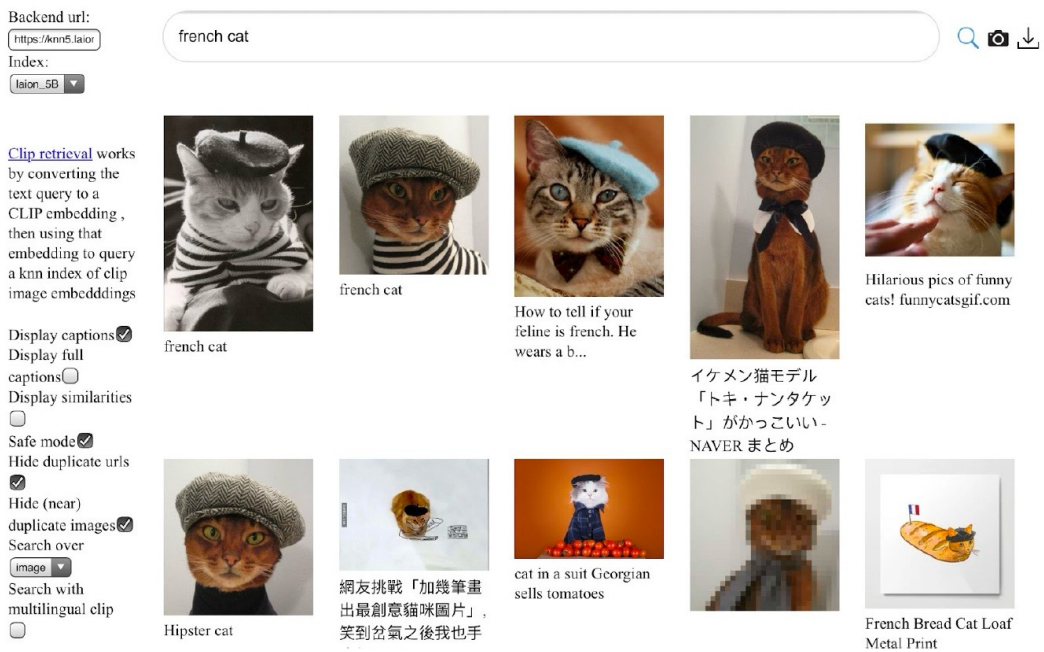


High-Resolution Image Synthesis with Latent Diffusion Models
Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer
[Submitted on 20 Dec 2021 (v1), last revised 13 Apr 2022 (this version, v2)]

Explicit Conditioning

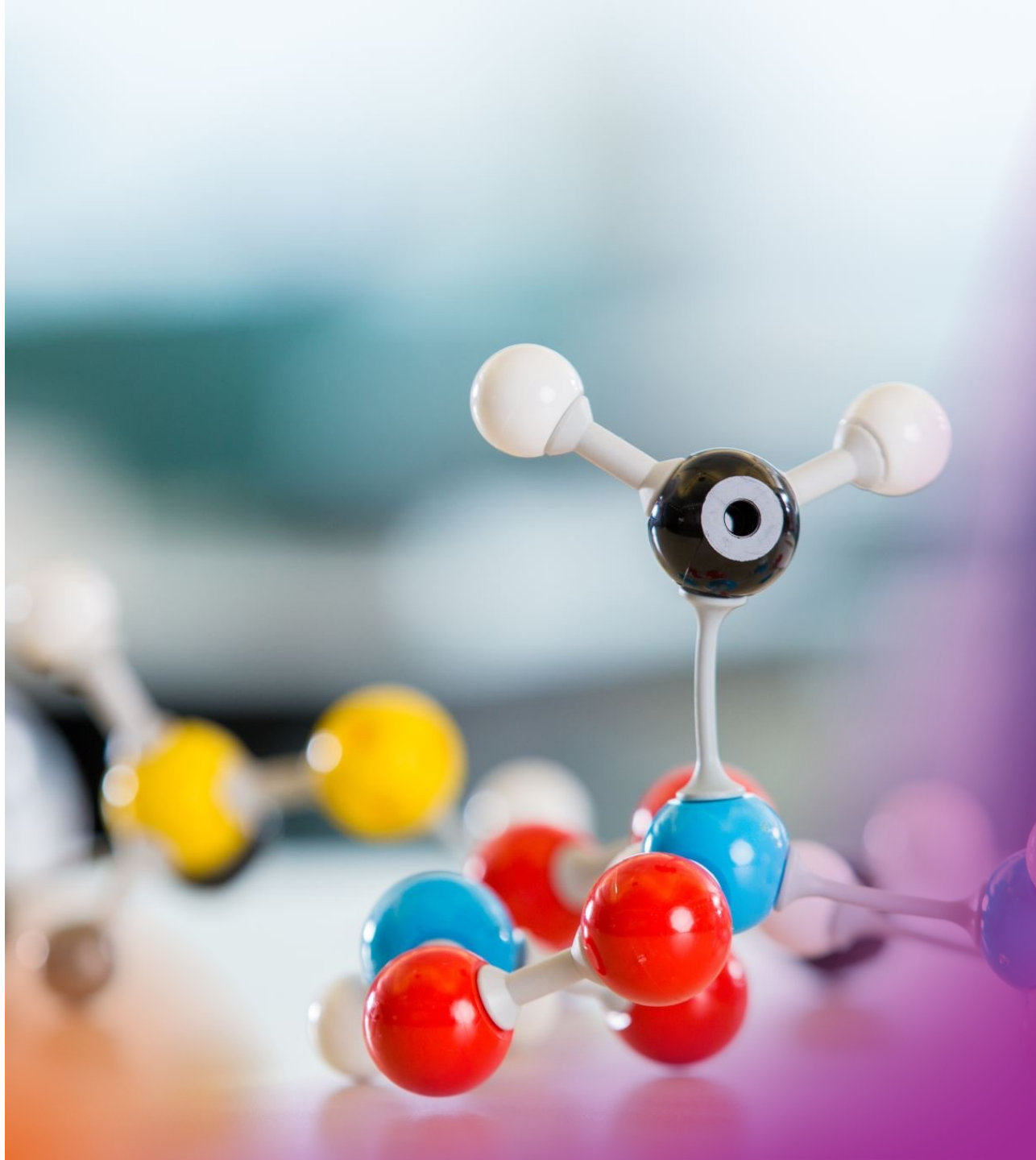
How do we train this?

Use an Image-Text dataset (for example, LAION 5B)



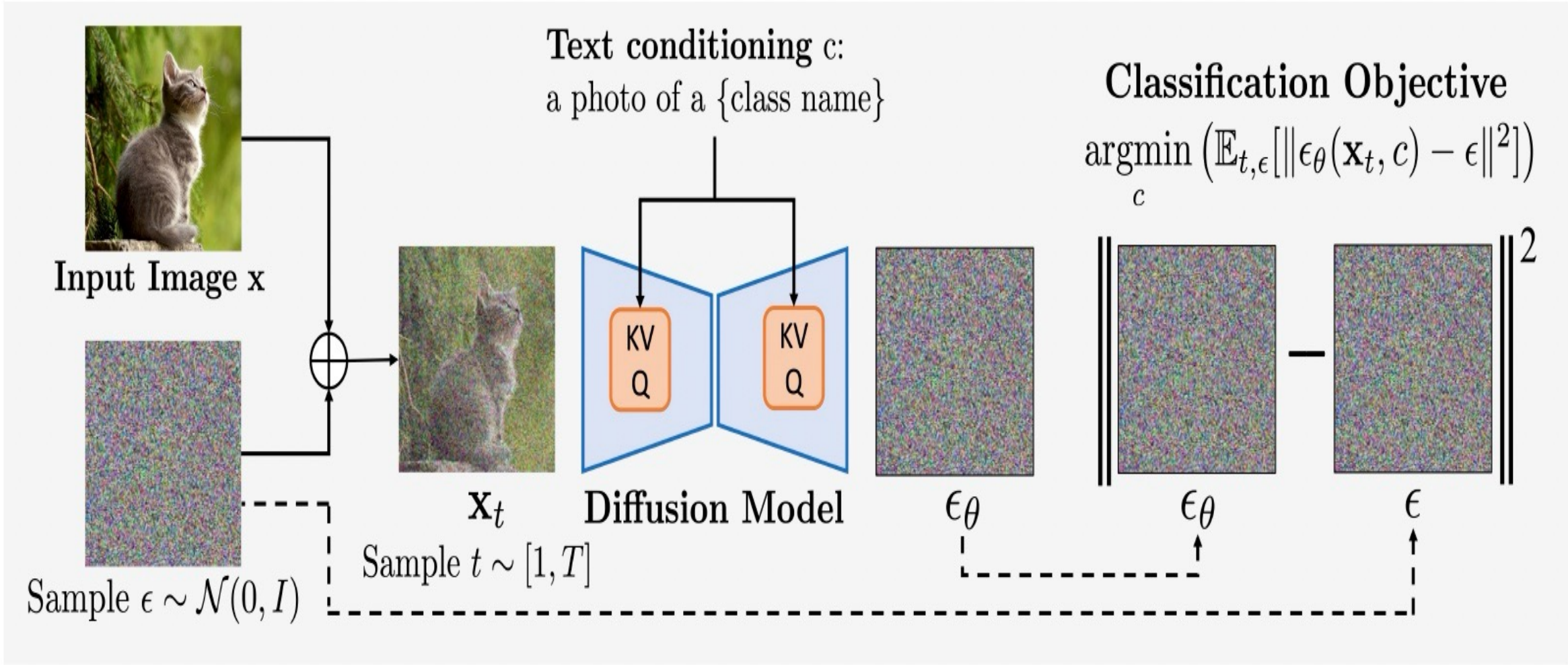
- Pretrained text encoder: OpenCLIP-ViT/H (OpenAI -2021)

Using a Diffusion Model as Classifier



"Diffusion Classifier"

[Submitted on 28 Mar 2023 (v1), last revised 13 Sep 2023 (this version, v3)]
Your Diffusion Model is Secretly a Zero-Shot Classifier
Alexander C. Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, Deepak Pathak



Dataset Preparation

- **Dataset**: UC Merced Land Use dataset (remote sensing data).
- **Classes**: 21 land use classes (e.g., agricultural, airplane, beach).
- **Images**: 100 images per class, 256x256 colored remote sensing data.
- **Split**: Only the test split is used. (No additional training)
- Resize images to 512x512 using bicubic interpolation.
- Center crop to 512x512 pixels.
- Convert to RGB and normalize.

Model Components

- **Latent Stable Diffusion**:

  Combines an autoencoder with a diffusion model trained in the latent space.

- **Components**:
  - **VAE**: Encodes images into latent representations.
  - **UNet**: Predicts noise added at each timestep.
  - **Text Encoder**: Encodes text prompts into embeddings.

Adaptive Probability Evaluation

- **Initialization**: Stable diffusion components (VAE, tokenizer, scheduler, UNet, text-encoder).
- **Noise generation**: Random noise of size (500, 4, 64, 64).
- **Process**:
  - Add noise to latent representation.
  - Pass through UNet to predict noise.
  - Evaluate prediction accuracy by comparing predicted noise with actual noise.
  - Iterate through prompts and timesteps, discarding high-loss prompts.

Error Evaluation

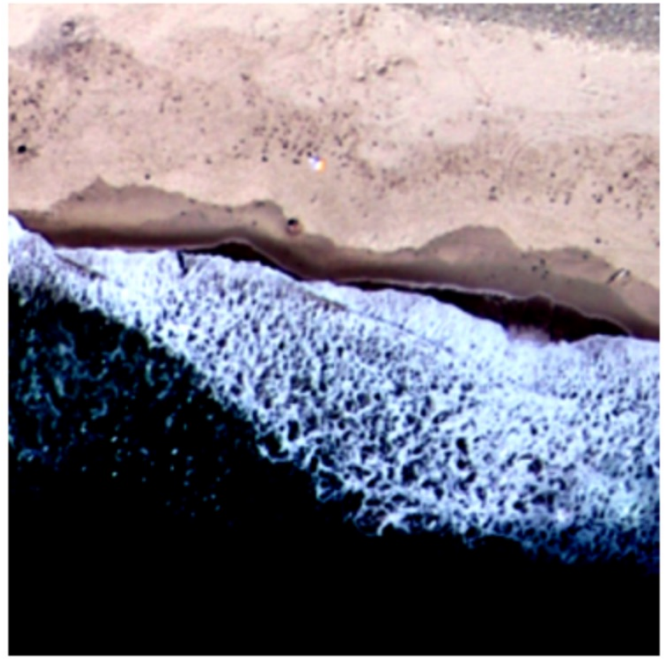
- **Steps**:
  - Generate noised latents.
  - Use UNet to predict noise at each timestep.
  - Compute mean squared error between actual and predicted noise.
  - Aggregate errors to assess model performance.

Evaluation Metrics

- **Metrics**:

```
Final Accuracy: 50.12%
Accuracy: 0.5012
Precision: 0.5349
Recall: 0.5012
F1 Score: 0.4881
```

True: beach | Predicted: beach



True: harbor | Predicted: harbor

True: parkinglot | Predicted: freeway

True: storagetanks | Predicted: agricultural