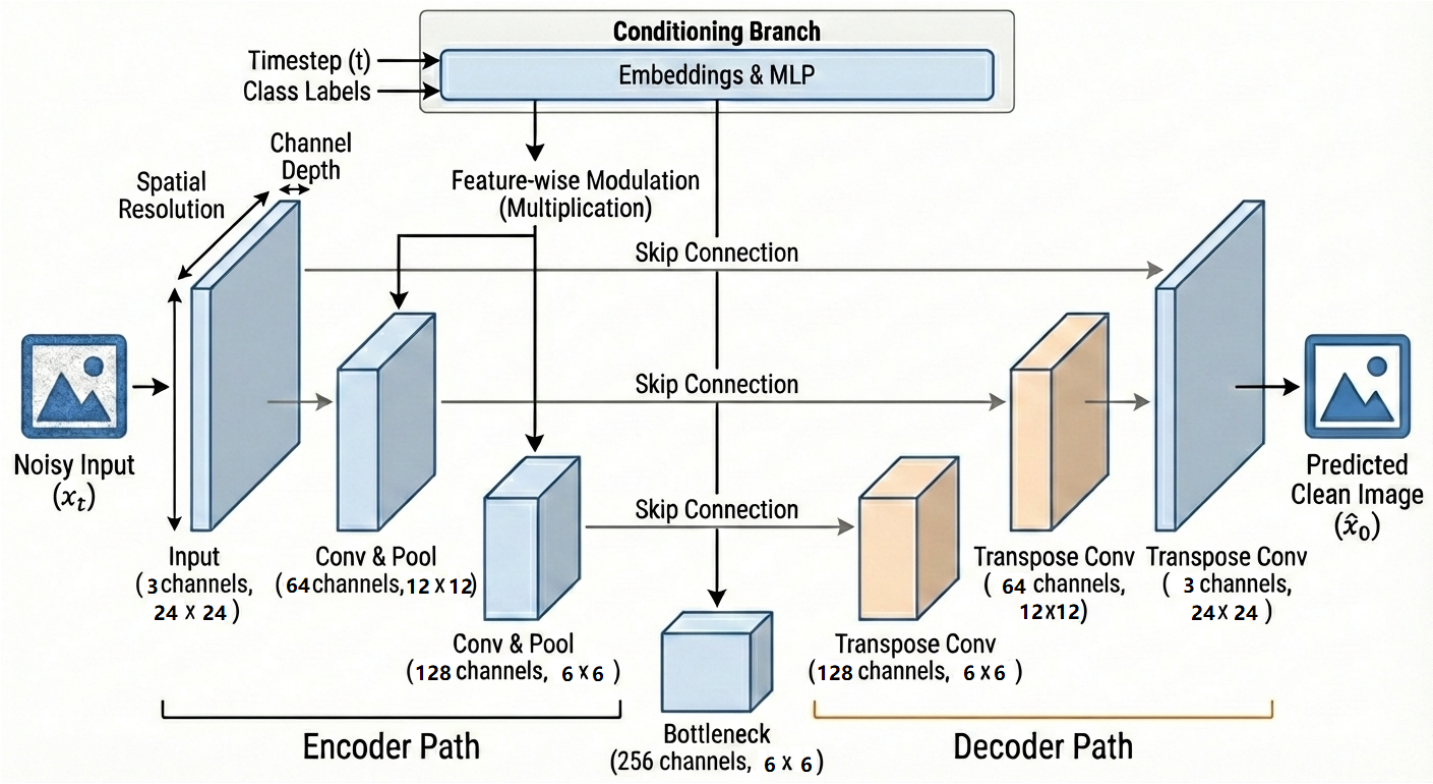
# Diffusion Report

## 1 General

In the first two part, the model adopts the traditional DDPM method for training. In the bonus point part, I have implemented a diffusion model which enables conditional generation using the method similar to that mentioned in He's recent work[1]. This method features letting the model predict the $x_{t-1}$ directly in the diffusion process instead of the noise $\epsilon_\theta(x_t, t)$ so that it can be subtracted to yield the target picture. I use the originally provided dataset, which turns out to be insufficient for the task. Thus, although there is remarkable drop in training loss, the images generated are far from satisfaction. However, during comparison study, the original DDPM method does prove to be inferior with higher loss after 1000 epochs (0.09) compared with 0.07 using the same encoder and decoder but different training objectives.

## 2 Architecture

### 2.1 Encoder-Decoder design




The model adopts a basic U-Net structure with convolutinal layers and pooling layers with Group Norm. As is mentioned in He's paper, a bottleneck design is added to force the model to learn a compact representation of the images.

### 2.2 Conditional generation

Time stamping and class indicators are handled using MLP. To include time information, each $t$ is encoded using sinusoidal functions similar to that in Transformers. Then, the embedded vectors are passed through a MLP to yield a 256-dim vector. For class labels, they are first embedded using PyTorch's builtin Embedding method. After that, the two representations are concatenated before passing a MLP layer which reduces them back to 256-dim. Later on they are converted to different dimensions using one linear layer. The embeddings are added to the $x_t$'s in an $x_t = x_t * (1 + cond\_embed)$ fashion.

### 2.3 Sampling and loss functions

According to the previous papers, model predicts the next image instead of the noise due to the fact that images are in essence in lower dimension than noise, which is distributed equally in high-dimensional space. The loss function is as follows.

$$\mathbb{E}[||x_0 - model(x_t, class\ label, t)||^2] \tag{1}$$

The $\mu$ of sampling process is as follows :

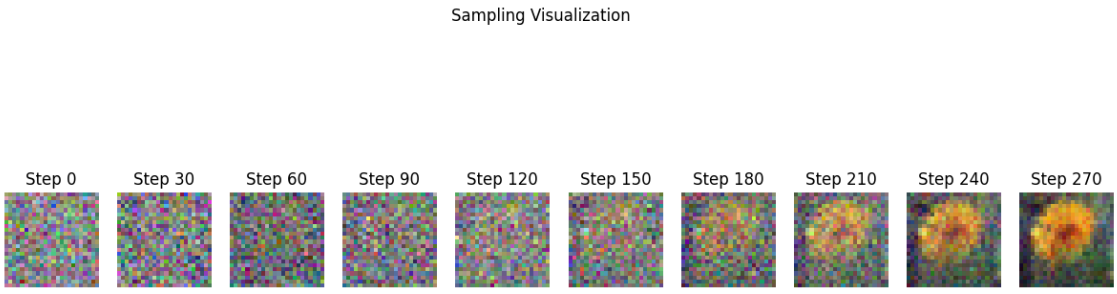
$$\mu = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{(1-\bar{\alpha}_t)}}(model(x_t, class\ label, t) - x_t)) \tag{2}$$

## 3 Results

Due to lack of data and computational resources, the performance of conditional generation is poor. However, the new method does perform better remarkably with lower loss and better quality.

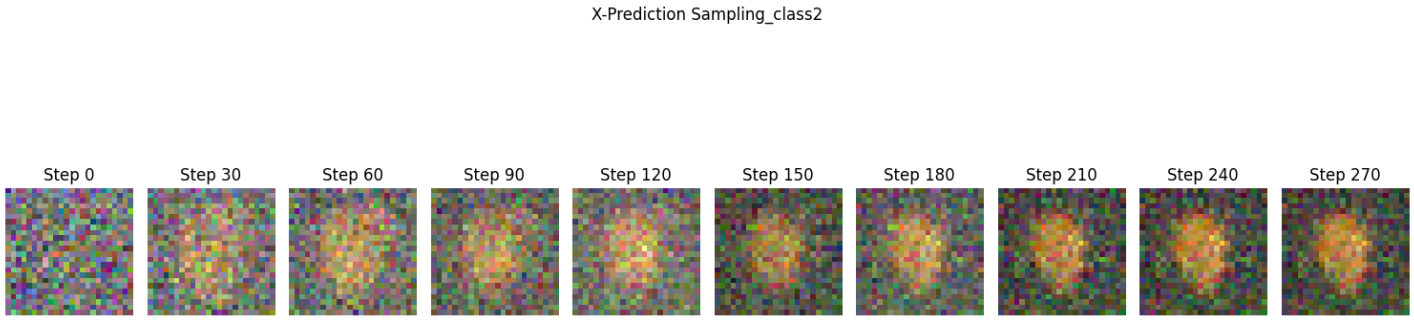
### 3.1 Random generation results (DDPM)

Below are the images generated after 930 epochs of training.

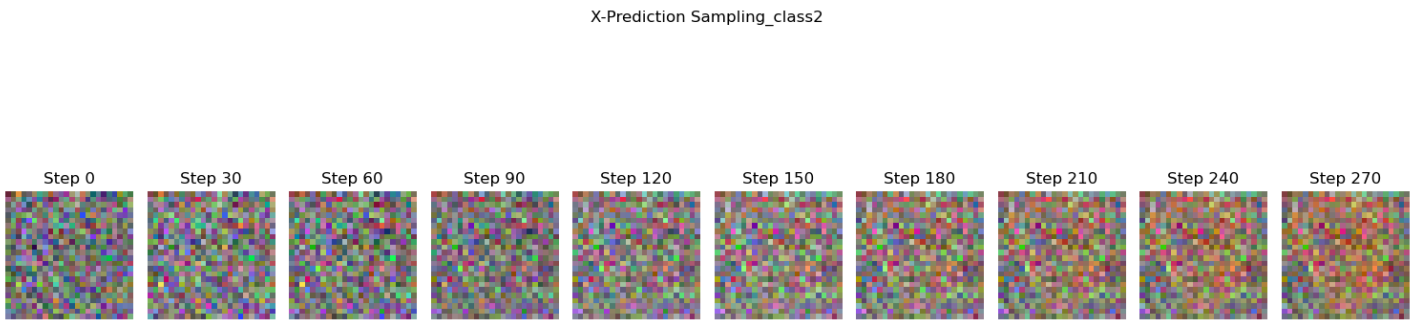



### 3.2 Conditional generation results

Examples of images generated by the model after 900 epochs of training: (using He's method)




compared to images generated by the same network but using DDPM loss function after 900 epochs of training:




More images and training log can be seen in the folder.

# 4 Appendix

[1] Tianhong Li, Kaiming He. Back to Basics: Let Denoising Generative Models Denoise (https://arxiv.org/abs/2511.13720)