Report on DDPM Training on CIFAR-10

Santanu Das (MA24M021)

August 31, 2025

Introduction

This project implements and trains a **Denoising Diffusion Probabilistic Model (DDPM)** on the CIFAR-10 dataset. The goal is to learn a generative model capable of synthesizing realistic images from Gaussian noise through an iterative denoising process.

DDPMs are a class of likelihood-based generative models that gradually corrupt training images with Gaussian noise and train a neural network to reverse this process. After training, the model can generate new samples by starting from pure noise and iteratively denoising it.

Diffusion Technique

The diffusion process consists of two stages:

Forward Diffusion (noising)

Starting from a clean image x_0, Gaussian noise is added step by step:

$$q(x_t|x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\big).$$

After T steps, the image becomes nearly isotropic Gaussian noise.

Reverse Diffusion (denoising)

A neural network $\epsilon_\theta(x_t, t)$ is trained to predict the added noise at each timestep. The reverse process gradually denoises the noisy sample:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\big).$$

The training objective is simplified to minimizing the mean squared error (MSE) between the true noise and the predicted noise:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0,t,\epsilon}\Big[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\Big].$$

Sampling Procedure

Sampling involves starting from Gaussian noise and iteratively applying the learned reverse diffusion process:

1. Sample $x_T \sim \mathcal{N}(0, I)$ (pure noise).
2. For $t = T, T-1, \ldots, 1$:
   - Predict noise: $\hat{\epsilon} = \epsilon_\theta(x_t, t)$.
   - Compute the mean for denoising:
   $$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}\Big).$$
   - Sample x_{t-1} from a Gaussian centered at $\mu_\theta(x_t, t)$.

At the end of this iterative process, we obtain a generated image x_0.

Evaluation of Generative Models using FID

We evaluate the quality of generated images using the **Fréchet Inception Distance (FID)**. FID measures the similarity between real and generated image distributions in the feature space of a pretrained InceptionV3 model. Lower scores indicate closer alignment with the real data distribution.

Model Architecture

The model used is a U-Net backbone with the following design choices:

- ResNet blocks for feature extraction.
- Group Normalization (8 groups).
- Self-Attention layers at intermediate resolutions (16×16).
- Time-step embedding: sinusoidal encoding of diffusion step t, injected via MLPs.
- Skip connections between encoder and decoder layers.

This architecture allows the model to capture both local (texture) and global (structure) features, making it suitable for high-quality image synthesis.

Training Procedure

The training loop follows these steps:

1. Sample minibatch of clean images from CIFAR-10.
2. Randomly select timestep t for each image.
3. Add noise to obtain x_t.
4. Predict noise using the U-Net model.
5. Compute loss (MSE between true noise and predicted noise).
6. Apply backpropagation and optimizer step.

Optimizer: Adam with learning rate 2×10^{-4}.

Dataset

The dataset used is CIFAR-10, which contains 60,000 images (32×32 resolution, 10 classes):

- 50,000 training images
- 10,000 test images

For this project, only the image data is used (class labels are ignored), since DDPM is unconditional generation.

Experiments

Exp-1

- Model: U-Net (30.4.M parameters more precisely 30,471,427 parameters) with attention and timestep embeddings.
- Batch size: 16
- Epochs: 100
- Base channels: 128
- Channel multipliers: (1, 2, 2, 2) $\rightarrow$ feature maps [128, 256, 256, 256].
- Attention layers applied at resolution 16×16.
- Diffusion steps: 400
- GPU P100 (on kaggle)

Results:

I ran with this configuration in Kaggle, it took 7 hours and 3 minutes for 39 epochs, then it stopped due to no memory left. Actually for this i was saving the model after each run of the epoch and generate an image with that trained model to check the progress. But i don't have saved files due to the huge file size.

Exp-2

- Model: U-Net (326,707 parameters) with attention and timestep embeddings.
- Batch size: 8
- Epochs: 50
- Base channels: 8
- Channel multipliers: (1, 2, 2, 2) $\rightarrow$ feature maps [8, 16, 16, 16].
- Attention layers applied at resolution 16×16.
- Diffusion steps: 200
- GPU $T4 \times 2$ (on kaggle)

Results:

I ran with this configuration in Kaggle, it took around 3 hours for complete run. I have saved the model checkpoints every 10 epochs and generated the image along with the FID score. But due to fewer numbers of channels, the generated images are not good. (i have uploaded the folder kaggle_8 in the github.)

Exp-3

- Model: U-Net (326,707 parameters) with attention and timestep embeddings.
- Batch size: 8
- Epochs: 50
- Base channels: 16
- Channel multipliers: (1, 2, 2, 2) $\rightarrow$ feature maps [16, 32, 32, 32].
- Attention layers applied at resolution 16×16.
- Diffusion steps: 200
- $T4$ GPU (on google colab)

Results:

I ran with this configuration in google colab, it took around 4 hours for a complete run (i am guessing that due to small batch size this happens). I have saved the model checkpoints every 10 epochs and generated the image along with the FID score. (I have uploaded the folder colab_16 to GitHub.)
I have also tried with base channels: 32, but it stopped due to the usage limit exceeded.

Conclusion

This report summarises the DDPM training setup on CIFAR-10, describing the diffusion process, model architecture, sampling procedure, and experiments. I just trained a small prototype version of it. For a perfect and accurate result, we have to run more epochs (as mentioned in the paper) and have to save model checkpoints frequently and generate the images frequently for supervision and also try with different type of optimizer for better results.
I have taken help from ChatGpt, perplexity and Gemini. And here is some of the references that help me to understand the paper and the code. reference-1 reference-2 reference-3