Deep Learning for Computer Vision

HW2 Report

b12901069 段奕鳴

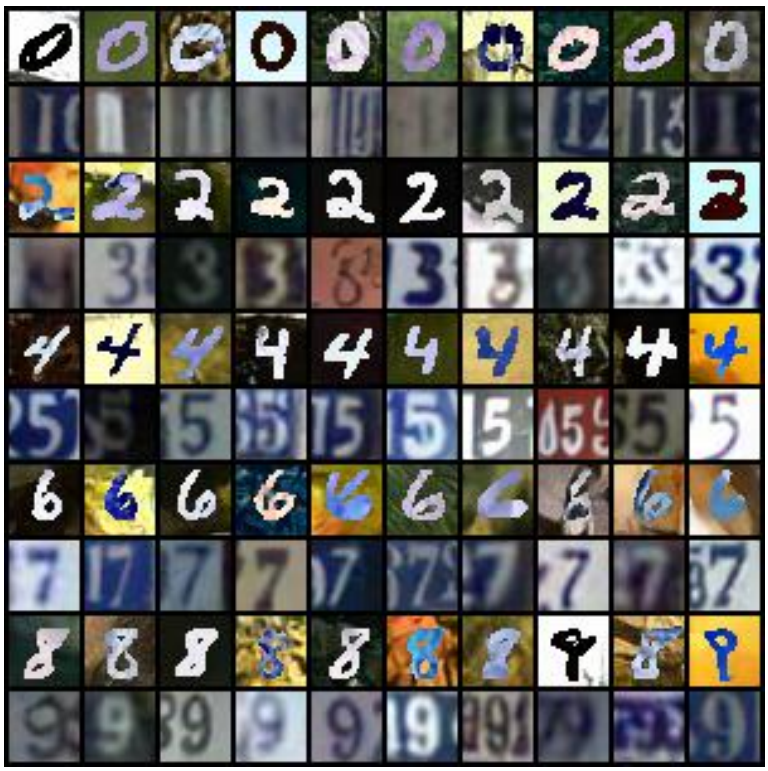
Problem 1: Diffusion Models

1. Describe your implementation details and the difficulties you encountered

I first spent some time deepening my understanding of diffusion models. I then implemented a conditional DDPM using a U-Net architecture (ContextUnet) with 3-channel RGB inputs of size 28×28 and 256 feature dimensions. The network employs residual convolutional blocks for stable training and uses separate embeddings for time steps and context to enable conditional generation. For noise scheduling, I applied a linear beta schedule ranging from 1e-4 to 0.02 over 1000 timesteps. Conditional generation uses classifier-free guidance: during training, a dropout probability of 0.2 is applied to the context, and during inference, a guidance weight of 2.0 is used to enforce stronger conditioning.

For training, I combined MNIST-M (even digits) and SVHN (odd digits) into a single dataset and trained the model for 200 epochs with a batch size of 64. The learning rate follows a linear decay from 2e-4, and the loss is the MSE between the predicted and actual noise. The training process is relatively straightforward, and convergence is achieved fairly swiftly. Although the submission version uses 200 epochs, the loss converged quite well before the full epochs.

2. Please show 10 generated images for each digit **(even digits for Mnist-M, odd digits for SVHN)** in your report. You can put all 100 outputs in one image with columns indicating different noise inputs and rows indicating different digits



3. Visualize a total of six images in the reverse process of the **first "0"** and **first "1"** in your outputs in (2) and with **different time steps**

first "0"

first "1"

Problem 2: DDIM

1. Please generate face images of noise **00.pt ~ 03.pt with different eta** in one grid. Report and explain your observation in this experiment.

We can observe that when η = 0, the DDIM sampling process becomes deterministic, producing the same image output for a given noise input across different runs. As η increases, the generated images become progressively more diverse, showing greater variation and randomness in the output. This indicates that a higher η introduces stronger stochasticity into the sampling process. When η > 0.75, the generated images show little to no visual correlation with those obtained under η = 0, suggesting that the outcome of the generation process becomes dominates by stochastic sampling.

2. Please generate the face images of the interpolation of noise **00.pt ~ 01.pt**. The interpolation formula is **spherical linear interpolation**, which is also known as **slerp**.

$$\boldsymbol{x}_T^{(\alpha)} = \frac{\sin((1-\alpha)\theta)}{\sin(\theta)}\boldsymbol{x}_T^{(0)} + \frac{\sin(\alpha\theta)}{\sin(\theta)}\boldsymbol{x}_T^{(1)}$$

where $\theta = \arccos\left(\frac{(\boldsymbol{x}_T^{(0)})^\top \boldsymbol{x}_T^{(1)}}{\|\boldsymbol{x}_T^{(0)}\|\|\boldsymbol{x}_T^{(1)}\|}\right)$. These values are used to produce DDIM samples.

in this case, $\alpha = \{0.0, 0.1, 0.2, \ldots, 1.0\}$

What will happen if we simply use linear interpolation? Explain and report your observation. (There should be two images in your report, one spherical linear and the other for linear)

SLERP (Spherical Linear Interpolation)

LERP (Linear Interpolation)

Observation:

From two generated images, SLERP produces smooth and fairly natural transitions between two noise samples. The interpolated faces show gradual and continuous changes in appearance. While the LERP produces perceptually distortions in intermediate steps, especially around $\alpha = 0.5$. The interpolation deviates from the original faces.

In high-dimensional spaces for diffusion models, linear interpolation does not stay on the manifold where valid noise samples lie. SLERP operates on a hypersphere, preserving the norm of the noise vectors. SLERP maintains constant norm throughout the path, while LERP cuts through the interior, create intermediate points with deviating norms. Also, SLERP maintains equal angular steps, meaning each interpolated point is equidistant in angle from its neighbors, which provides constant velocity interpolation.

Problem 3: ControlNet

1. Describe your implementation details and the difficulties you encountered

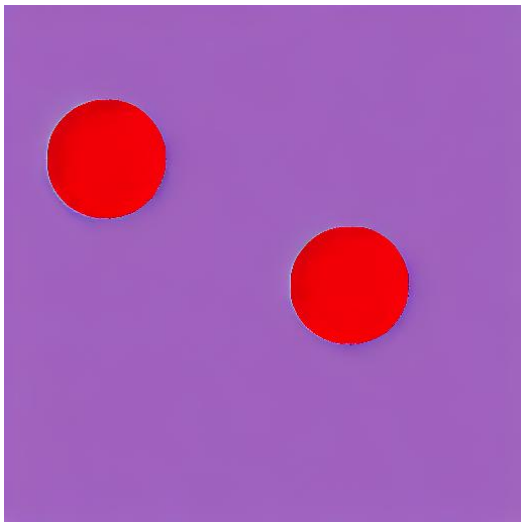
The implementation is heavily adopted from the official ControlNet repository (https://github.com/lllyasviel/ControlNet) with Stable Diffusion v1.4 as the backbone. The model was trained on the Fill50k dataset using PyTorch Lightning for 3 epochs. The training configuration used a batch size of 4, learning rate of 1e-5, with the SD encoder frozen while the ControlNet layers remained trainable, following the zero-convolution approach. For inference, the implementation uses a DDIM sampler with 50 diffusion steps, guidance scale of 7.5, and deterministic sampling with eta=0.0. For the testing data, one image was generated per condition with a fixed random seed of 42 to ensure reproducibility.

The first main difficulty encountered was setting up the environment. Package version conflicts and dependency issues proved annoying, especially this homework's requires to manage multiple conda environments. Beyond the technical setup, it also took considerable time to understand more about how diffusion models and ControlNet work.

Training was also computationally expensive, requiring approximately 4 hours per epoch on the GPU. Thankfully despite this long training duration, 2-3 epochs were sufficient to generate fairly well results, suggesting that the model converged relatively quickly to useful features.

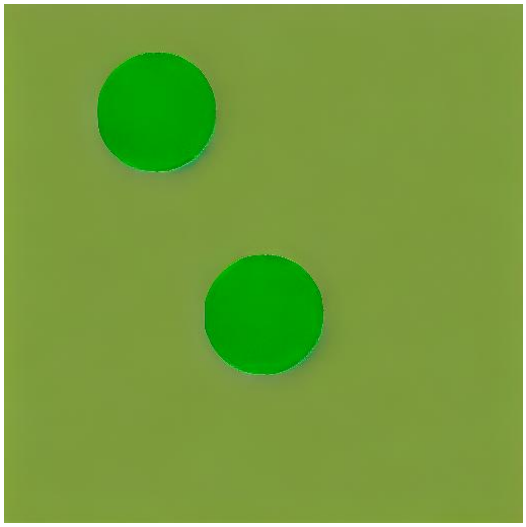
2. Generate images using ControlNet with a control image containing two circles

prompt 1: “a red circle and a blue circle with pink background”



prompt 2: “a green circle and a yellow circle with blue background

”



Observations:

The generated results show that ControlNet struggles to accurately control the specified colors. In most test cases, the model correctly applies the first color but paints both circles with the same shade, often failing to render the second circle and the background in the intended colors. This failure primarily stems from a mismatch between the training data and the task. Although the training set includes some complex color descriptions, it only involves one color for the circle and one for the background. As a result, the model is unable to generalize effectively to scenarios involving two circles.