1. Introduction

In this lab, we will implement a conditional Denoising Diffusion Probabilistic Models (DDPM) to generate synthetic images.

- Implement a conditional DDPM
- Design our noise schedule and UNet architecture
- Implement the dataloader, training function and testing function
- Generate the synthetic images

2. Implementation

A. Describe how you implement our model, including our choice of DDPM, UNet architecture, noise schedule, and loss functions?

Compared with DDPM, conditional is almost identical but adds the encoding of the class label into the encoding of the class label into the timestep by passing the label through an embedding layer. DDPM contains UNet which we choose the classic one. Different from tradition one, up and down blocks support an extra timestep argument and add the encoding of the class label into timestep. In up and down blocks, we also use 4 blocks type and mainly use the regular ResNet downsampling block. DDPM also contains DDPM scheduler which we choose DDPMschduler. DDPM denoises an image by taking random noise the size of the desired output and passing it through the model. At each timestep, the model predicts the noise residual and the scheduler uses it to predict a less noisy image.

First, we load data from dataloader and normalize the input images by transforms. And go through forward process, we add noise to the clean images at each timestep. After getting noisy image, we feed the noisy image to the model and get the model prediction. Next, we evaluate how well the model denoises by MSEloss and update model by AdamW optimizer. In order to generate RGB images in the reverse diffusion process, we apply the denormalization to finish it. Finally, use ResNet18 to compute accuracy.

B. Specify the hyperparameters(learning rate, epoch, etc.)?

The hyper-parameters for experiments as shown below.

batch size: 16

epoch: 50

learning rate: 0.0001

sample size: 64

warmup step of learning rate: 500

optimizer: AdamW

loss function: MSE loss

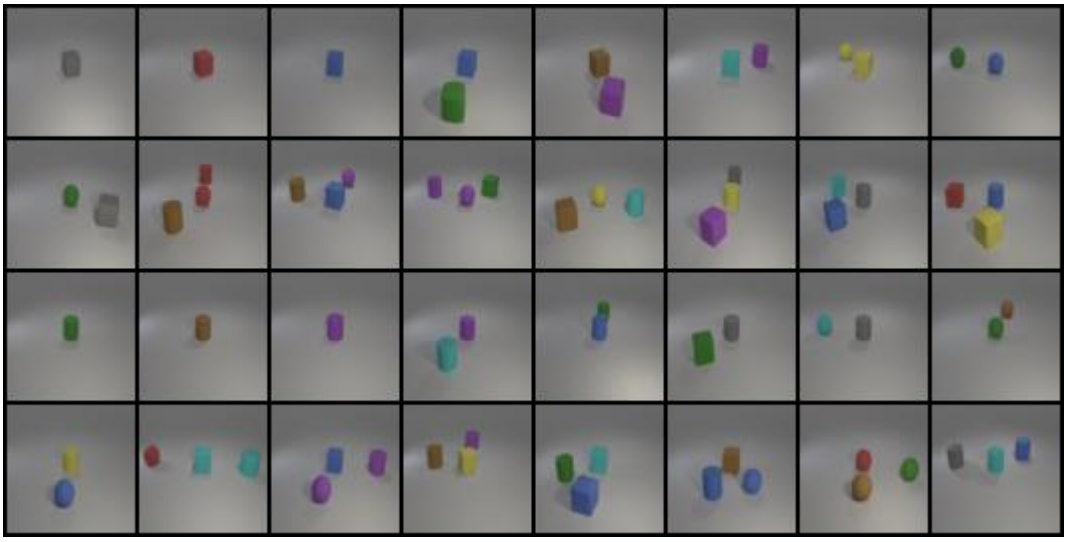
3. Results and discuss

A. Show your accuracy screenshot based on the testing data

```
Accuracy: [Test]: 0.8333, [New Test]: 0.8571
```

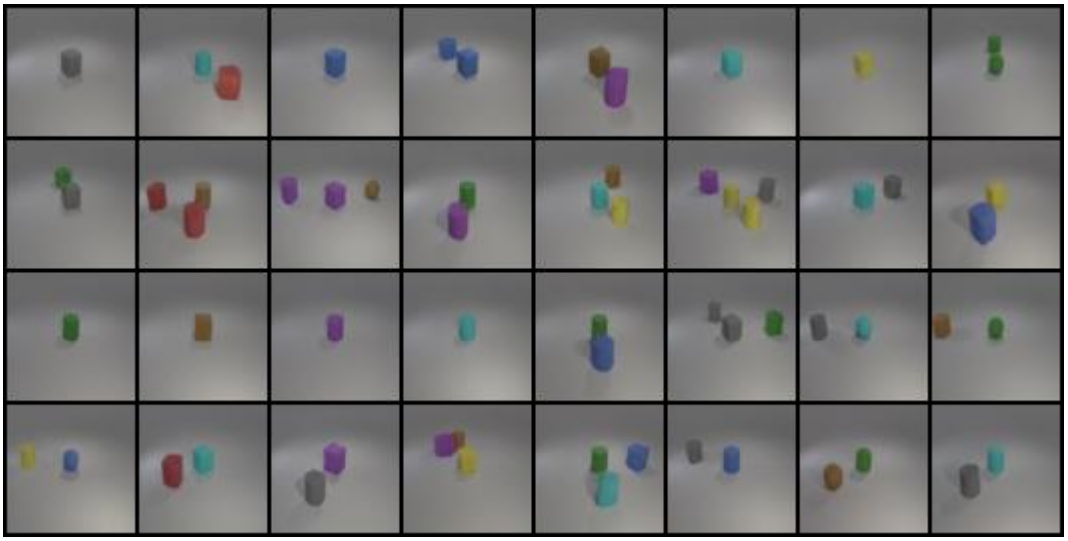
New_test:

Test:

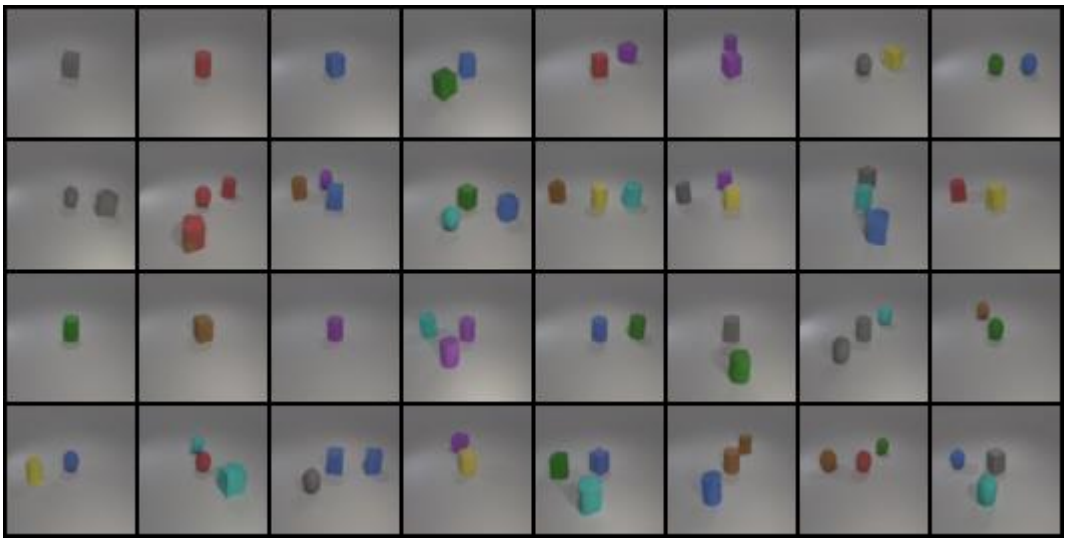


B. Show your synthetic image grids and a progressive generation image

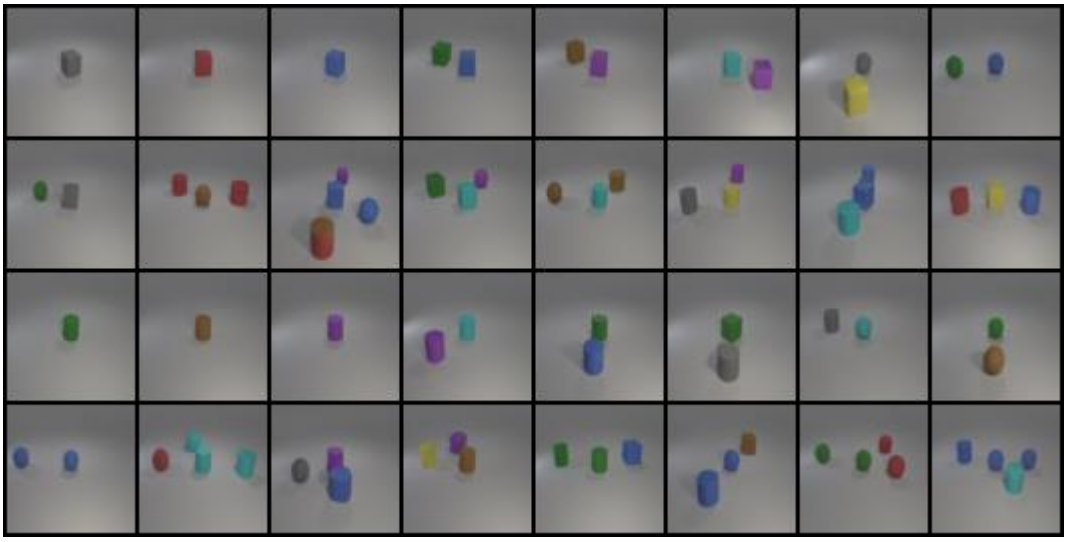
Epoch=10



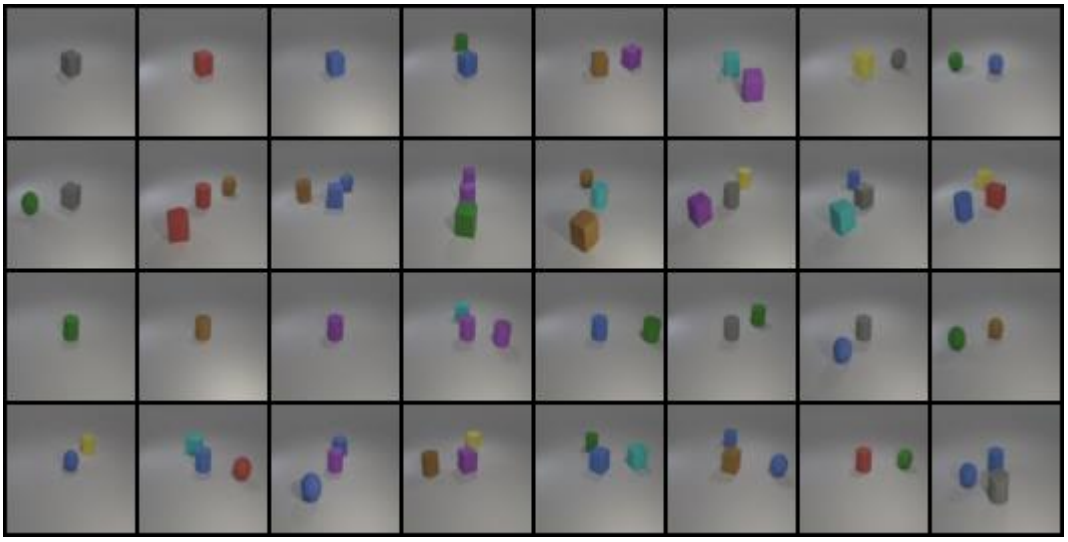
Epoch=20



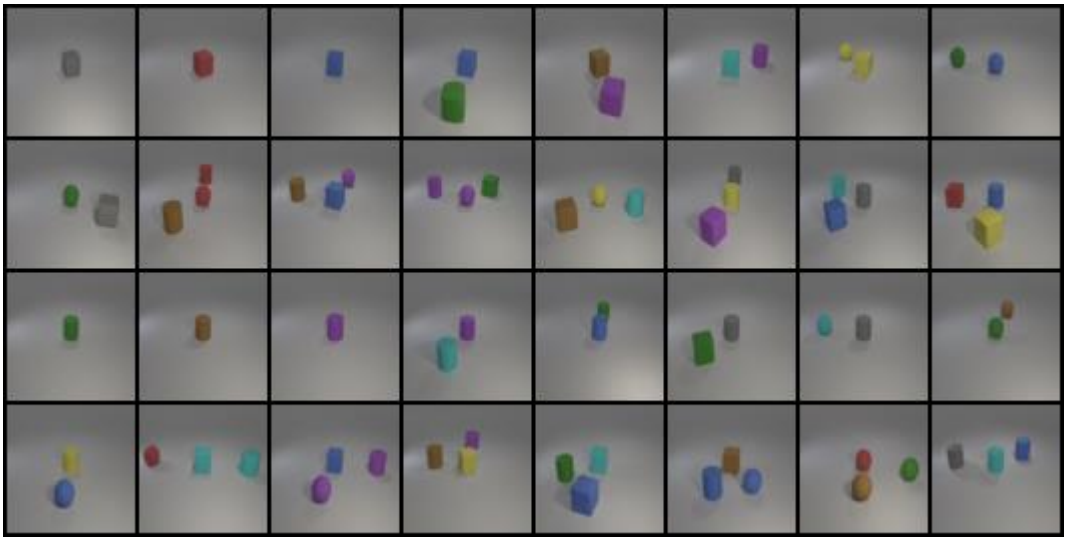
Epoch=30



Epoch=40



Epoch=50



C. Discuss the results of different model architectures pr methods

Different prediction types for DDPM scheduler:

when we choose the “epsilon” as our prediction type, it is obvious that the synthetic images have the background color shifting at the beginning of training phase. We try “v_prediction” avoids color shifting artifacts that affect high resolution diffusion models. But the result is not good, so we think that it may only fit for video problem.

Different beta scheduler for DDPM scheduler:

cosine schedule tends to reach optimal performance more quickly than those trained with the linear schedule thorough the training curve

Training curve with cosine scheduler