312553038 楊奕儒
多工碩一

DLP Lab6 report

Introduction:

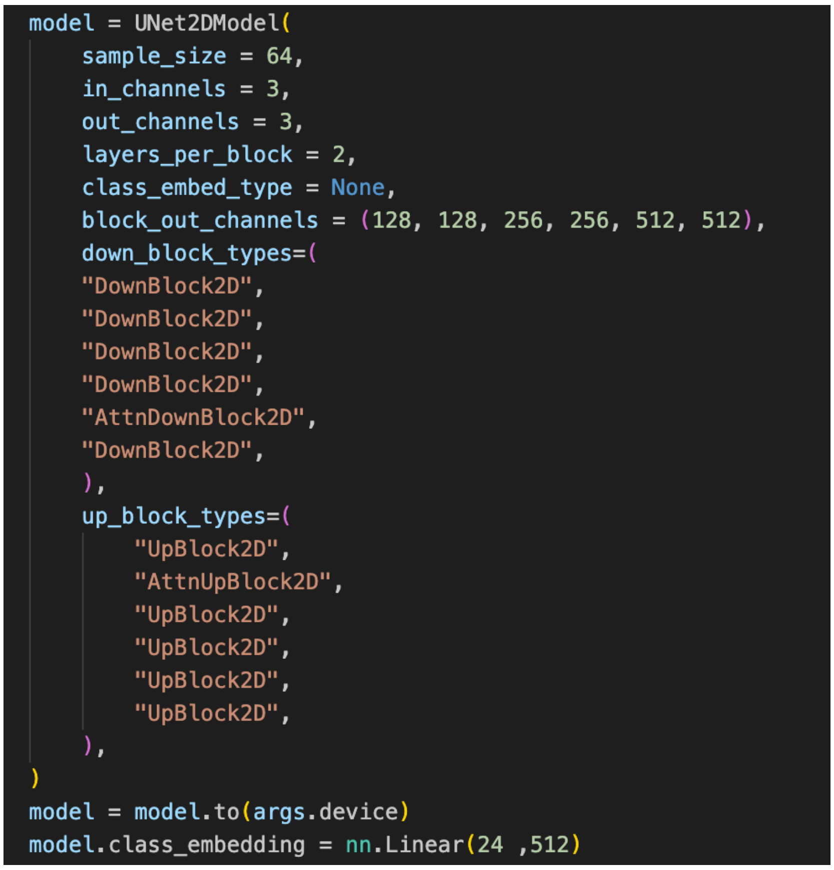
In this Lab, I need to implement a conditional DDPM and generate synthetic images based on multi-labels conditions. In training stage, we need to predict noise using Unet model and minimized the loss of noise and noise_prediction. In the sampling stage, we begin with a Gaussian noise and utilize our model to predict and remove the noise step by step. Eventually, we obtain a clear picture.

Implementation details

1.Describe how you implement your model, including your choice of DDPM, UNet architectures, noise schedule, and loss functions.

Unet:

I utilize the UNet2DModel as the foundational architecture. Within the down_block_types section, there are a total of five instances of the "DownBlock2D" type, complemented by a single instance of the "AttnDownBlock2D" type. Similarly, this pattern is mirrored within the up_block_types segment.



```
model = UNet2DModel(
    sample_size = 64,
    in_channels = 3,
    out_channels = 3,
    layers_per_block = 2,
    class_embed_type = None,
    block_out_channels = (128, 128, 256, 256, 512, 512),
    down_block_types=(
    "DownBlock2D",
    "DownBlock2D",
    "DownBlock2D",
    "DownBlock2D",
    "AttnDownBlock2D",
    "DownBlock2D",
    ),
    up_block_types=(
        "UpBlock2D",
        "AttnUpBlock2D",
        "UpBlock2D",
        "UpBlock2D",
        "UpBlock2D",
        "UpBlock2D",
    ),
)
model = model.to(args.device)
model.class_embedding = nn.Linear(24 ,512)
```

https://huggingface.co/docs/diffusers/v0.20.0/en/tutorials/basic_training

noise schedule：

I choose square cosin as my noise schedule

Loss function :

I use Mean Square Error as my loss function. The predict noise and the ground truth noise are the input to the loss function.

Train:

During the training phase, for each iteration in the trainloader enumeration, I introduce random noise which is then added to the original image, resulting in the creation of a modified input termed as noise_x. Subsequently, this modified input, alongside the corresponding label, is employed as the input to my UNet model. The primary objective of this utilization is to predict the noise inherent in the image. After the prediction, I compare the predict noise and ground truth noise to compute MSE loss.

Sample:

start from a Gaussian noise, iteratively input xt, timestep, class embed to denoise the image. Compute the xt-1.

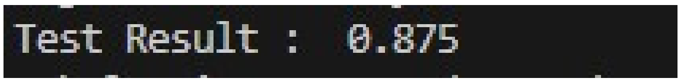
2.Specify the hyperparameters (learning rate, epochs, etc.)

learning rate : 1e-4 *0.5
epochs : 100
optimizer : AdamW
timestep = 1000

Results and discussion

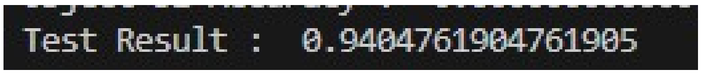
1.Show your accuracy screenshot based on the testing data.

Test:



```
Test Result :  0.875
```

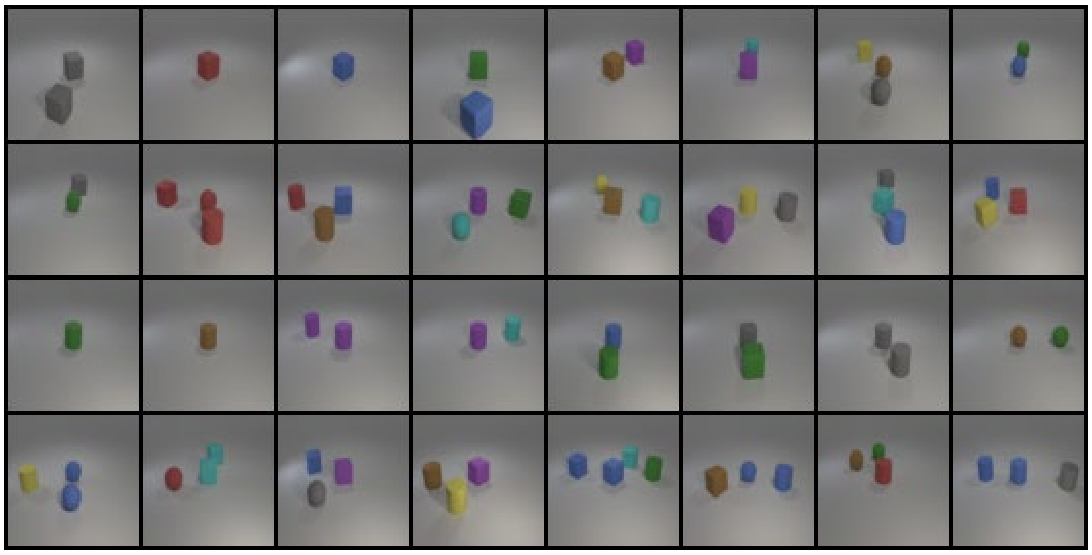
New_test:

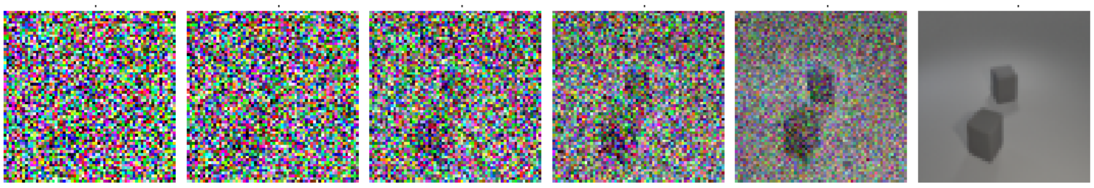


```
Test Result :  0.9404761904761905
```

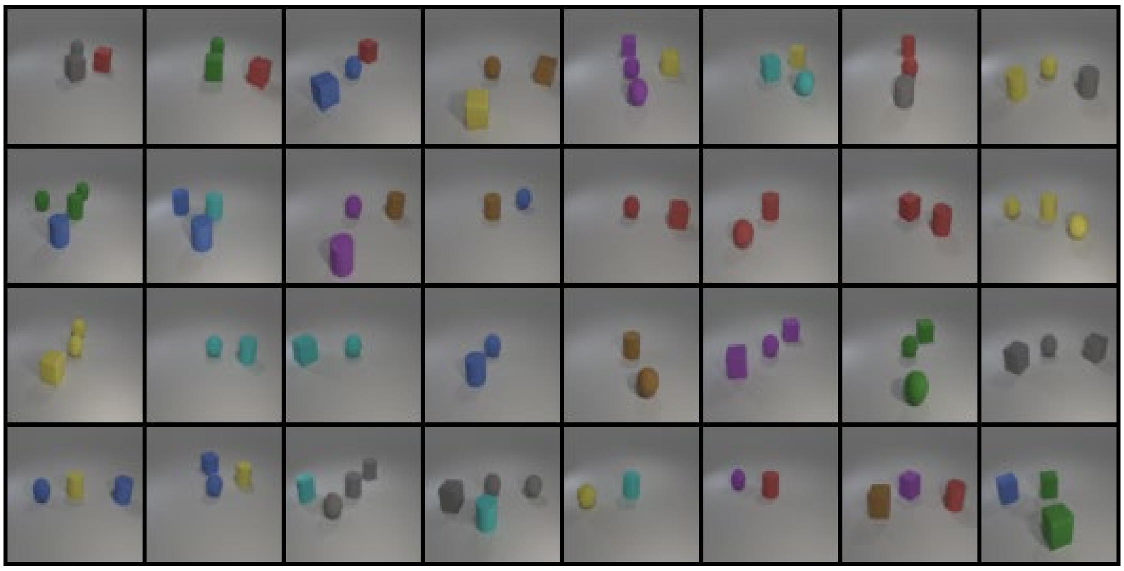
2.Show your synthetic image grids and a progressive generation image.

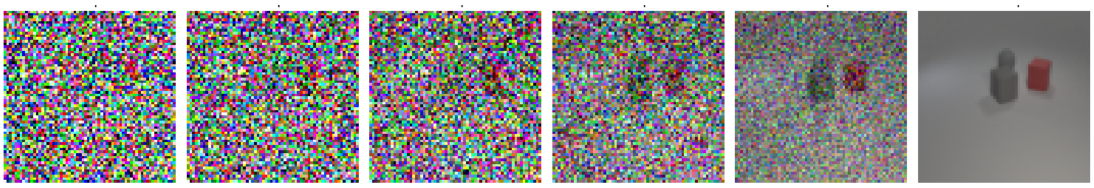
Test:





New_test:

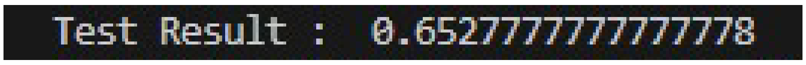




3.Discuss the results of different models architectures or methods.
Without model.class_embedding = nn.Linear(24, 512) after training for 20 epochs, we observe a significant drop in both test accuracy and new test accuracy. This suggests that even though the model remains capable of performing image processing tasks, it may lack the ability to capture subtle distinctions between different categories. This deficiency could potentially impact the model's performance on multi-class data, particularly when it comes to discerning complex relationships between distinct categories.

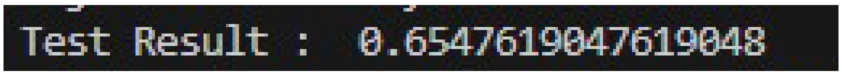
With class embedding:

Test:



```
Test Result :  0.6527777777777778
```

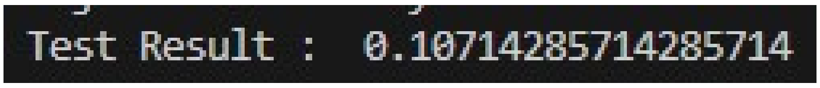
New_test:



```
Test Result :  0.6547619047619048
```

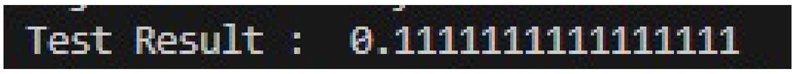
Without class embedding:

Test:



```
Test Result :  0.10714285714285714
```

New_test



```
Test Result :  0.1111111111111111
```