Lab6: Let’ s Play DDPM

Student id / name: A113599 / 楊淨富

1. Introduction

Implement a conditional Denoising Diffusion Probabilistic Model (DDPM) to generate synthetic images according to multi-label conditions.

There’re 24 labels in this lab (object.json). The DDPM can be roughly separated into two parts: Generating an image from sampling Gaussian noise and Denoising process from the previous image, aiming to learn the features from noise.

2. Implementation details

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

- UNet: from diffusers import UNet2DModel

```
>>> from diffusers import UNet2DModel

>>> model = UNet2DModel(
...     sample_size=config.image_size,  # the target image resolution
...     in_channels=3,  # the number of input channels, 3 for RGB images
...     out_channels=3,  # the number of output channels
...     layers_per_block=2,  # how many ResNet layers to use per UNet block
...     block_out_channels=(128, 128, 256, 256, 512, 512),  # the number of output channels for each UNe
...     down_block_types=(
...         "DownBlock2D",  # a regular ResNet downsampling block
...         "DownBlock2D",
...         "DownBlock2D",
...         "DownBlock2D",
...         "AttnDownBlock2D",  # a ResNet downsampling block with spatial self-attention
...         "DownBlock2D",
...     ),
...     up_block_types=(
...         "UpBlock2D",  # a regular ResNet upsampling block
...         "AttnUpBlock2D",  # a ResNet upsampling block with spatial self-attention
...         "UpBlock2D",
...         "UpBlock2D",
...         "UpBlock2D",
...         "UpBlock2D",
...     ),
... )
```

Reference link:

https://huggingface.co/docs/diffusers/tutorials/basic_training

- Noise schedule: torch.linspace to create a linear beta range.
- Loss function: nn.MSEloss() to calculate the loss between predicted noise and the ground truth.

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

- Leaning rate = 1e-4
- Epoch = 200
- Optimizer = torch.optim.AdamW
- Scheduler = get_cosine_schedule_with_warmup
- Timestep = 1000

3. Experimental results

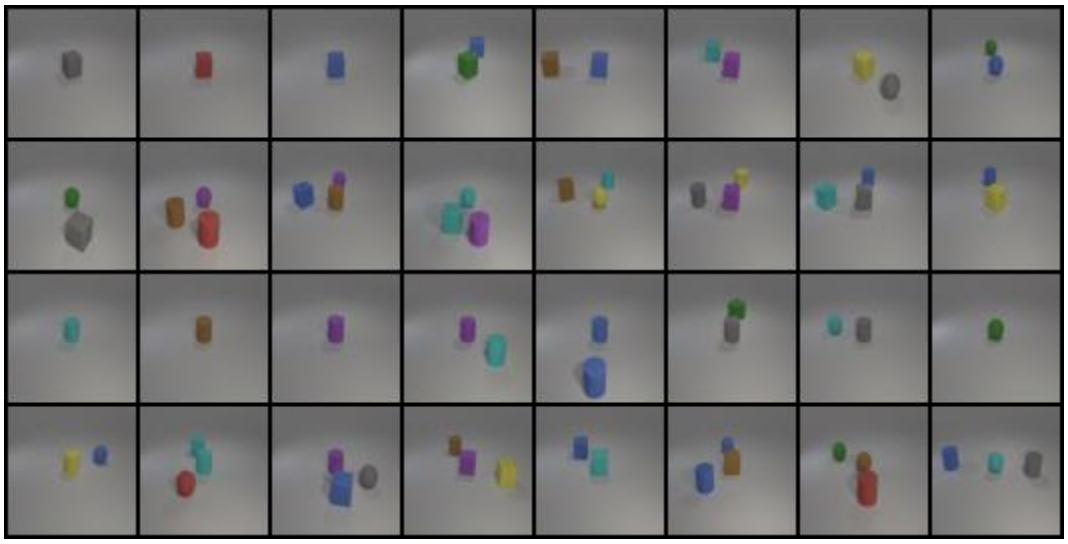
3.1. Show your accuracy screenshot based on the testing data.

- Test acc: 84.72%
- New test acc: 85.71%

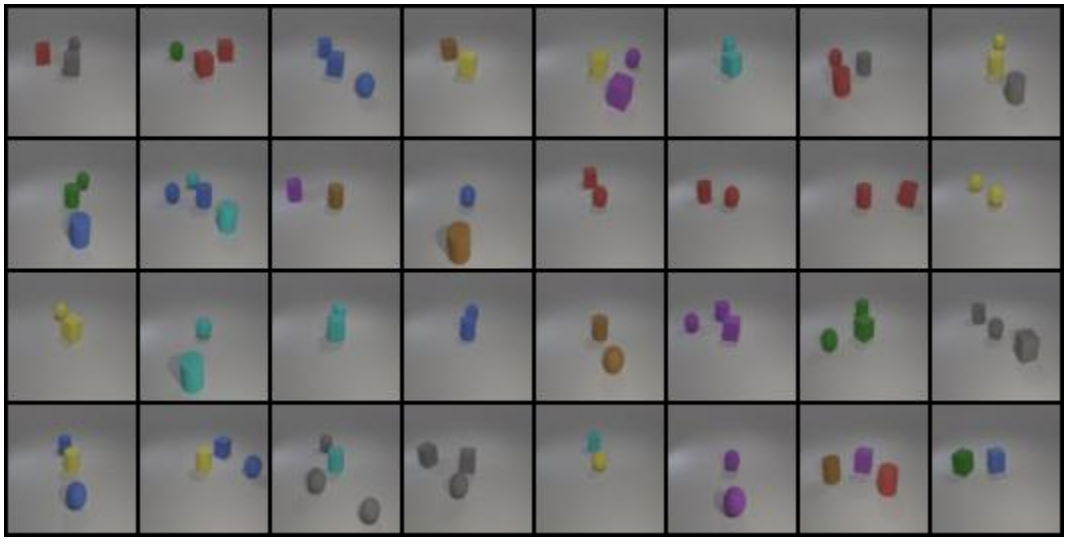
```
(lab6) PS C:\Users\User\Desktop\NYCU_DLP\Lab6_Lets_play_DDPM> python .\main.py
Some weights of the model checkpoint at ./model/Unet_epoch_200 were not used when initializing UNet2DModel: ['class_embe
dding.weight', 'class_embedding.bias']
- This IS expected if you are initializing UNet2DModel from the checkpoint of a model trained on another task or with an
other architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing UNet2DModel from the checkpoint of a model that you expect to be exactly
identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
C:\Users\User\.conda\envs\lab6\Lib\site-packages\torchvision\models\_utils.py:208: UserWarning: The parameter 'pretraine
d' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
C:\Users\User\.conda\envs\lab6\Lib\site-packages\torchvision\models\_utils.py:223: UserWarning: Arguments other than a w
eight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is e
quivalent to passing `weights=None`.
  warnings.warn(msg)
Testing acc: 84.72%
Testing acc: 85.71%
```

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

Test dataset:



New test dataset:



3.3. Discuss the results of different model architectures or methods. For example, what is the effect with or without some specific embedding methods, or what kind of prediction type is more effective in this case.

- I’ve tried Exponential Moving Average to make generative model to be more stable but it turns out to take too long to train so I didn’t apply on my source code. But it indeed helps model to smooth the updates of model parameters, offering benefits in terms of stability and generalization. EMA model can provide the following advantages: Stable Generated Image Quality, Preventing Mode Collapses, Enhancing Generalization, Generating More Realistic Images.