NYCU 2025 Spring DL Lab6
Generative Models

TA 楊賀弼

April 22, 2025

Outline

- Rule
- Lab description
- Model design guide
- Scoring criteria

Rule

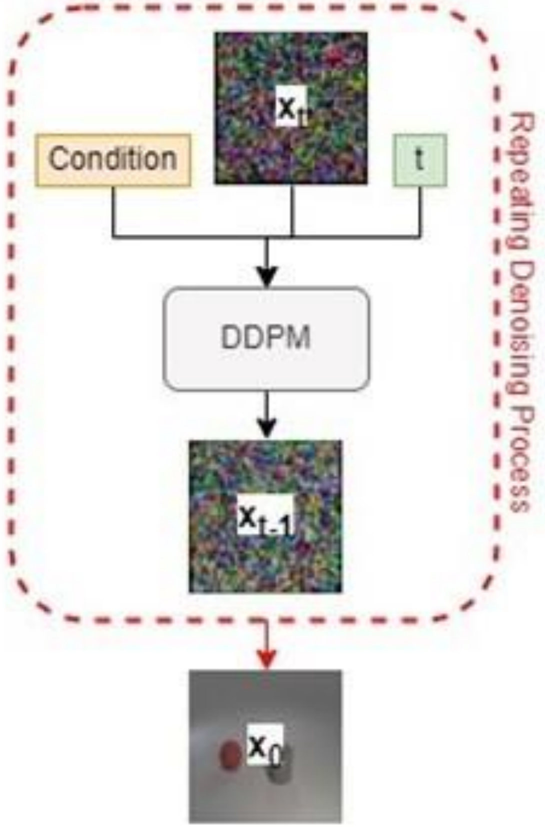
- Report submission deadline: May 6, 2025, 11:59 p.m.
- No need to demo this lab
- Zip all files in one file
  - Report (.pdf)
  - Source code
  - images Folders -|-- test
    |-- new_test
- Name it "DL_LAB6_YourStudentID_YourName.zip”
  - Example: “DL_LAB6_313553051_楊賀弼.zip”
- -5% to your score if you do not follow the format

```
images/
├── test/
│   ├── 0.png
│   ├── 1.png
│   ├── ...
│
├── new_test/
│   ├── 0.png
│   ├── 1.png
│   ├── ...
```

Lab description

Lab objective

- You need to implement a DDPM to generate synthetic images according to multi-label conditions



File Descriptions

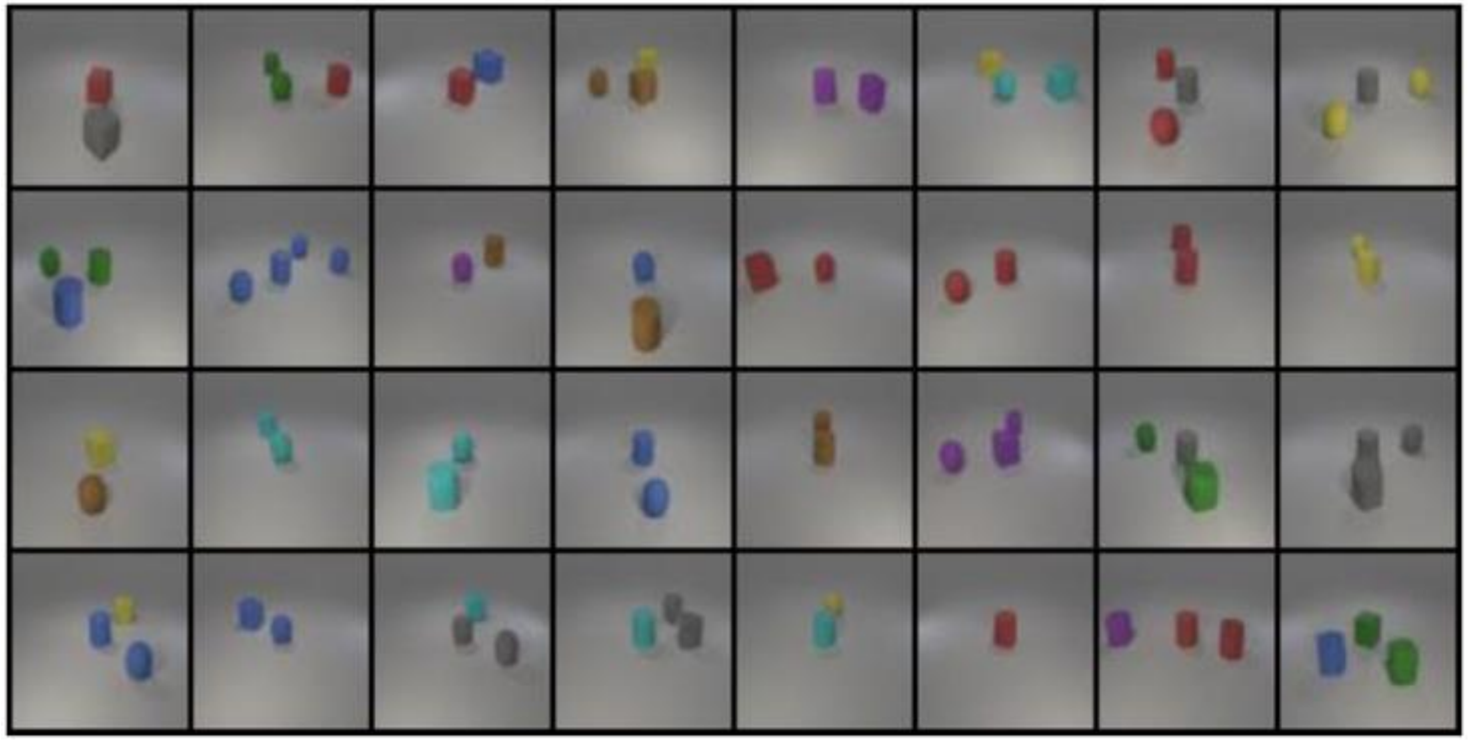
- You can download file.zip from e3 or ntu cool, except iclevr.zip, from the open-source Google Drive (link).
- There are 7 files in the file.zip: readme.txt, train.json, test.json, new_test.json, object.json, evaluator.py, and checkpoint.pth. All the details of the dataset are in the readme.txt.

Requirements

- Design and train your models
  - Implement a conditional DDPM
  - You can use any architecture you like
    (Feel free to use any library that helps your work, but you need to mention the implementation details and include the reference in your report)
- Synthesis image and evaluate the results
  - Show the synthetic images in grids (for test.json, new_test.json)
  - Show the denoising process in a grid for sampling from your DDPM
    (with the label set ["red sphere", "cyan cylinder", "cyan cube"])
  - Evaluate your model with the classification accuracies from the provided evaluator (for test.json, new_test.json)

Requirements

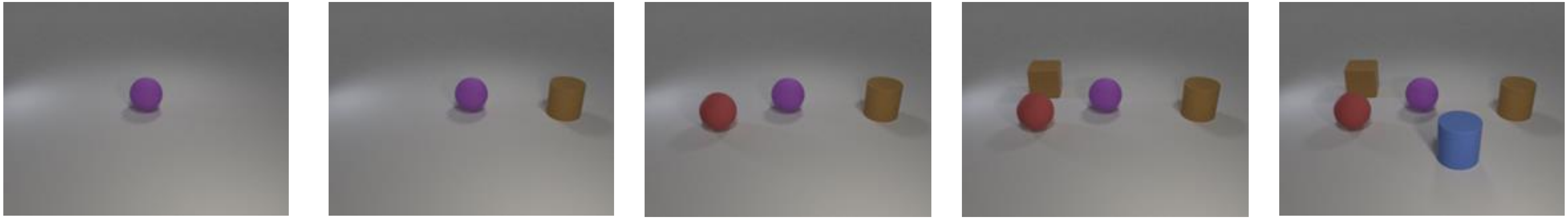
- Synthetic images in grids



- Denoising process

Dataset

- Provided files
  - readme.txt, train.json, test.json, new_test.json, object.json, iclevr.zip
- object.json
  - Dictionary of objects
  - 24 classes
- Example of labels:
  - [“cyan cylinder”, “red cube”], [“green sphere”], …
  - The same object will not appear twice in an image



Pretrained evaluator

- Provided files: evaluator.py, checkpoint.pth
  - DO NOT modify any of them
- Use **eval**(images, labels) to compute accuracy of your synthetic images
  - Labels should be one-hot vector. E.g. [[1,1,0,0,…],[0,1,0,0,…],…]
  - Images should be all generated images. E.g. (batch size, 3, 64, 64)
  - Images should be normalized with transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
- You can involve this evaluator in training or sampling for better result
  - E.g. classifier guidance for DDPM
  - Inherit the class if you need extra functionalities
  - Again, DO NOT modify the weight and class script

Implement a conditional DDPM

Design of DDPM

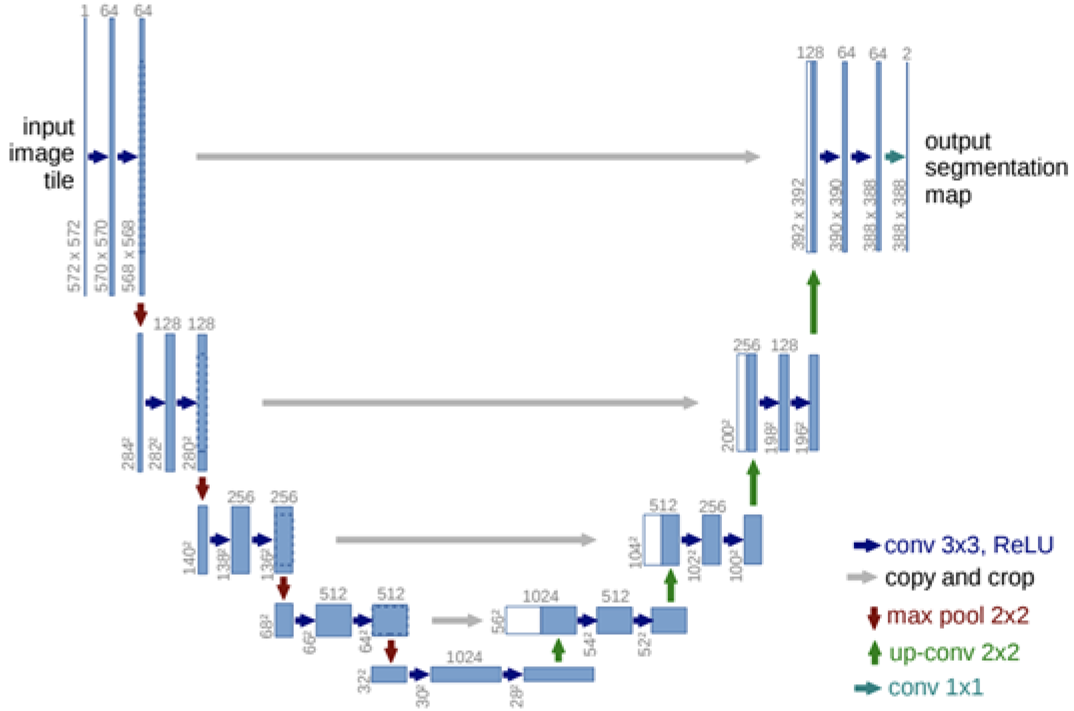
- Denoising diffusion model
- Latent diffusion model
  - encoder/decoder design

- Condition embedding
  - label embedding
  - Time embedding

- Noise schedule
  - linear
  - cosine
  - other

Design of UNet

- Number of blocks
- Number of layers, channels in each blocks
- Down/upsampling blocks design
  - Architecture for the blocks. e.g. Resnet block
  - Self/cross attention
  - Where to add condition
  - other



Choice of sampling method

- DDPM, DDIM, etc.
- With classifier guidance (Diffusion models beat gans on image synthesis)

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log \boxed{p_\phi(y|x_t)}, \Sigma)$
end for
return x_0

You can include the pretrained evaluator (classifier) here

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1 - \bar{\alpha}_t}\, \nabla_{x_t} \log \boxed{p_\phi(y|x_t)}$ You can include the pretrained evaluator (classifier) here
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1 - \bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0

Simple suggestions (in the other PDF)

- Denoising Diffusion Probabilistic Models
- Hugging Face Diffusion Models Course

The Hugging Face Diffuser library offers comprehensive functionality to help you implement various diffusion model architectures, scheduling methods, sampling types, reparameterizations, etc. You can refer to the tutorial linked above to design your diffusion model for this lab.

Scoring criteria

Scoring criteria

- Report (60%)
  - Introduction (5%)
  - Implementation details (25%)
    - Describe how you implement your model, what is this step, including your choice of DDPM, noise schedule. (Please write in detail.)
  - Results and discussion (30%)
    - Show your synthetic image grids (16%: 8% * 2 testing data) and a denoising process image (4%)
    - Discussion of your extra implementations or experiments (10%)

Scoring criteria

- Result (40%) (based on results shown in your report)
  - Classification accuracy on test.json and new_test.json.
  - Show your accuracy screenshots.
  - 20% * 2 testing data

Accuracy	Grade
score ≥ 0.8	100%
$0.8 >$ score ≥ 0.7	90%
$0.7 >$ score ≥ 0.6	80%
$0.6 >$ score ≥ 0.5	70%
$0.5 >$ score ≥ 0.4	60%
score < 0.4	0%

Save Images

- Save your synthetic images for the 2 testing data (test.json , new_test.json)
  Save in png files !!!

- The data structure will be organised as follows:

```
images/
├── test/
|    ├── 0.png
|    ├── 1.png
|    ├── ...
|
├── new_test/
|    ├── 0.png
|    ├── 1.png
|    ├── ...
```

- The image name should follow the order in the test.json and new_test.json.

Output examples

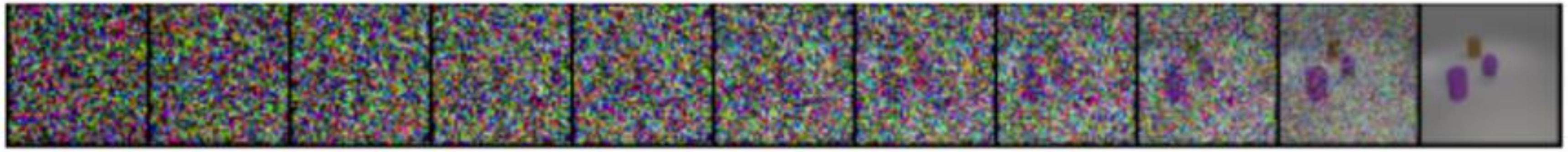


Fig. 2: Example of the denoising process image.

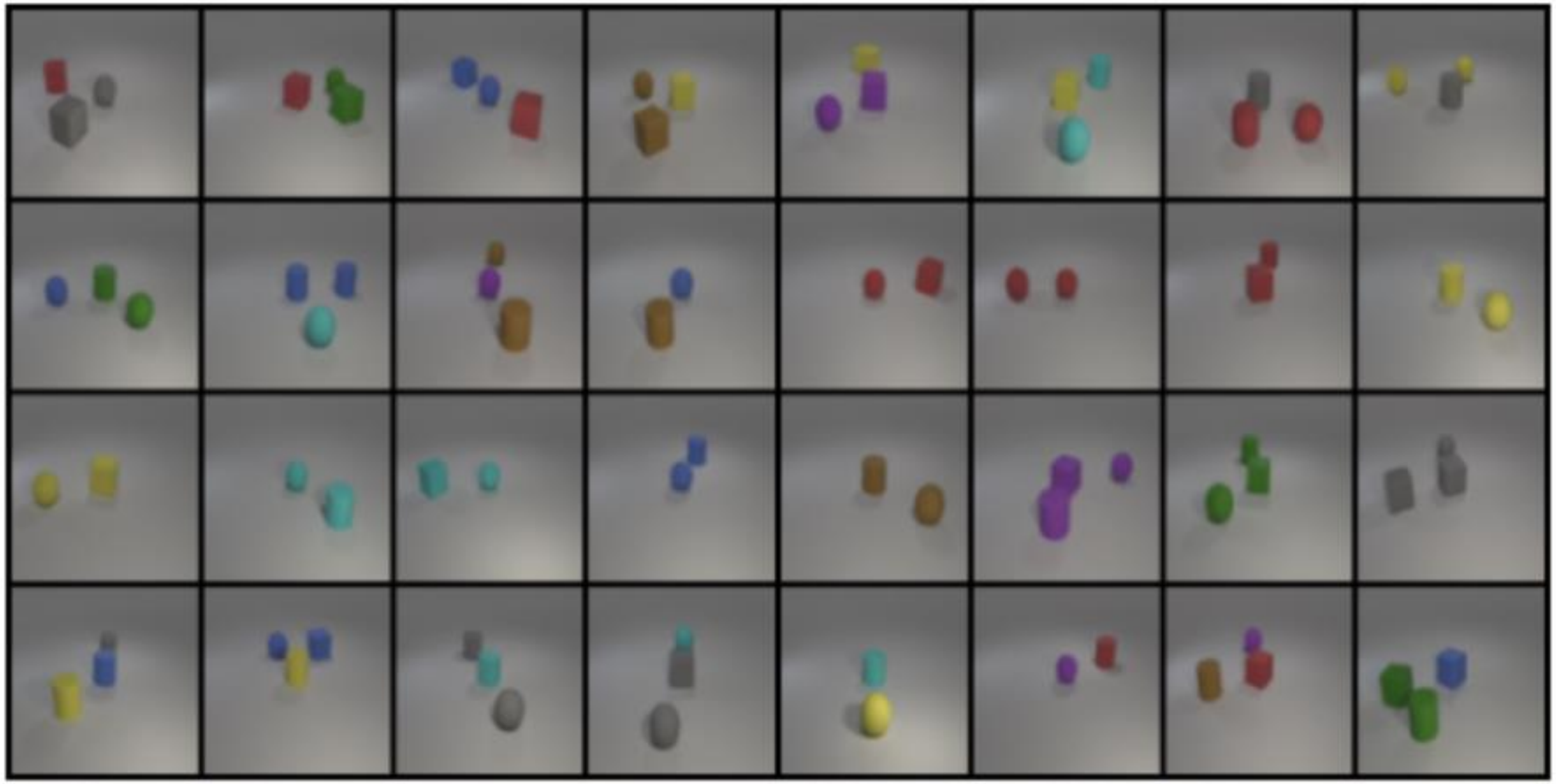


Fig. 3: The synthetic image grid on new_test.json. (F1-score 0.821)