NYCU 2024 Spring DL Lab6
Generative Models

TA 林廷翰

May 21, 2024

Outline

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

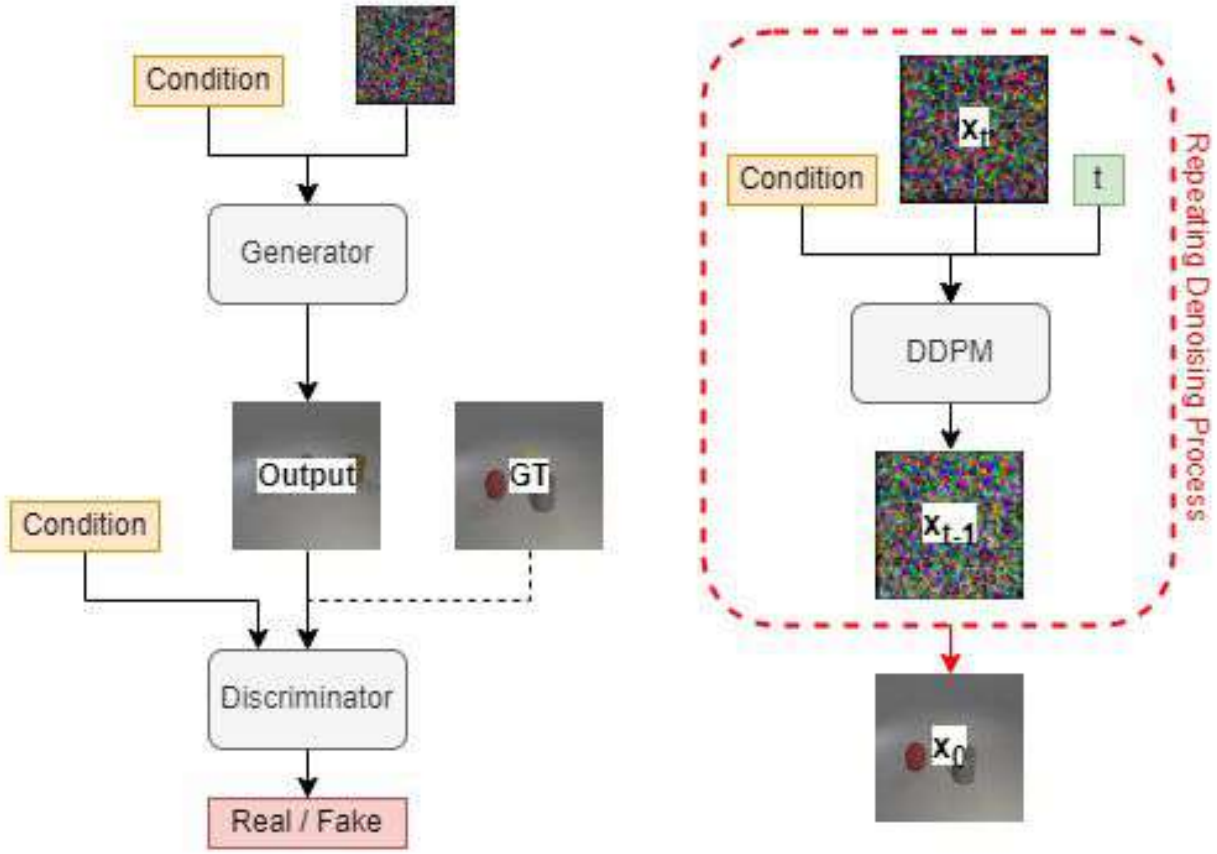
Rule

- Report submission deadline: June 6, 2024, 11:59 a.m.
- No need to demo this lab
- Zip all files in one file
  - Report (.pdf)
  - Source code
- Name it "DL_LAB6_YourStudentID_YourName.zip”
  - Example: “DL_LAB6_311605003_林廷翰.zip”
- -5% to your score if you do not follow the format

Lab description

Lab objective

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

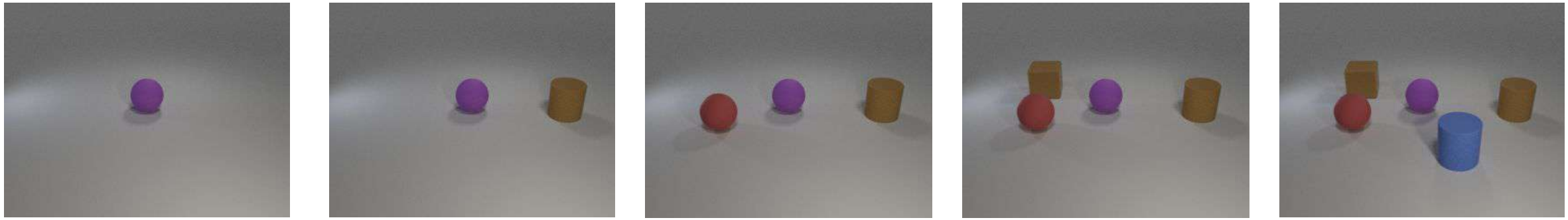


Requirements

- Design and train your models
  - Implement a conditional GAN
  - Implement a conditional DDPM
  - You can use any architecture you like
- 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,” “yellow cube,” “cyan cylinder”])
  - Evaluate your 2 models with the classification accuracies from the provided evaluator (for test.json, new_test.json)

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. discriminator for GAN, classifier guidance for DDPM
  - Inherit the class if you need extra functionalities
  - Again, DO NOT modify the weight and class script

Implement a conditional GAN

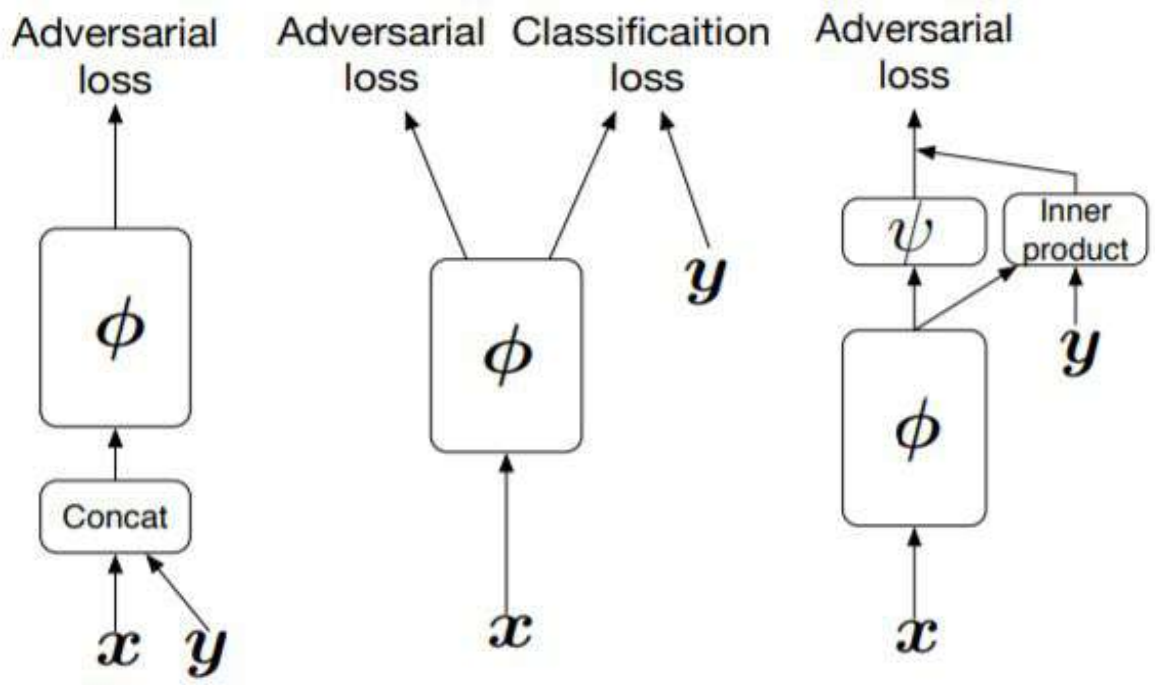
Design of GAN

- De-convolution layers
- Basic block
- Bottleneck block
- Residual
- Self-attention

- E.g.
  - DCGAN
  - SA-GAN
  - Progressive GAN

Choice of conditional GAN

- Generator
  - Concatenation, multiplication, batch normalization, etc.
- Discriminator
  - Conditional GAN
  - InfoGAN
  - Auxiliary GAN
  - Projection discriminator
- Hybrid version



Choice of loss functions

- GAN loss function

$$L_D^{GAN} = E\left[\log\left(D(x)\right)\right] + E\left[\log\left(1 - D(G(z))\right)\right]$$

$$L_G^{GAN} = E\left[\log\left(D(G(z))\right)\right]$$

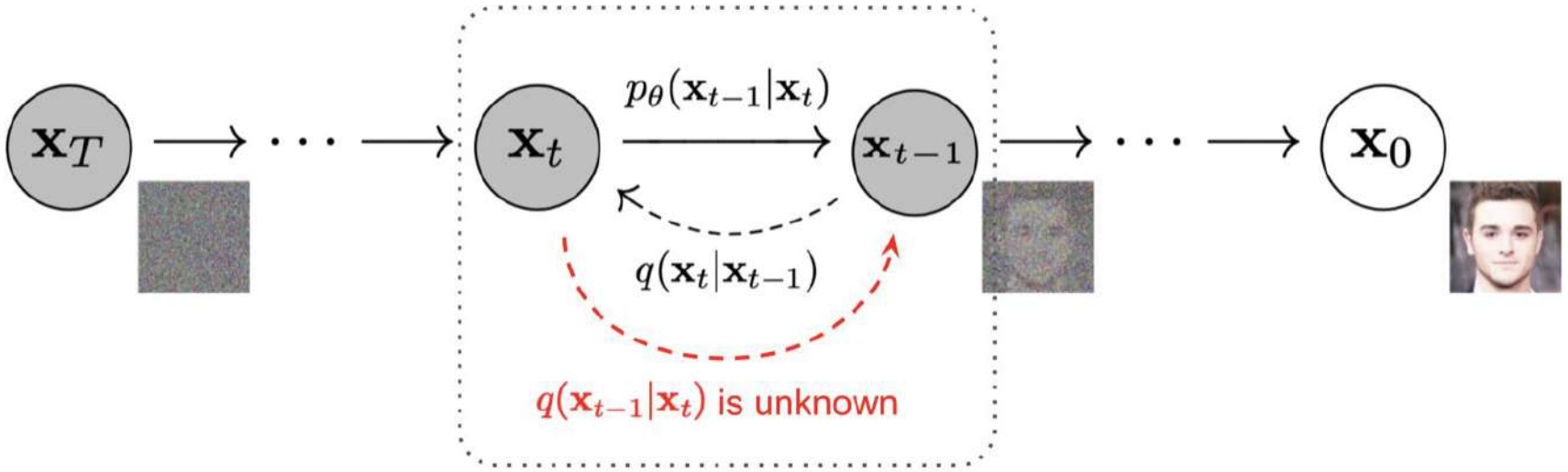
- LSGAN
- WGAN
- WGAN-GP

- Hybrid version

- Combine with your choice of conditioning method

Implement a conditional DDPM

Diffusion model basic



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Design of DDPM

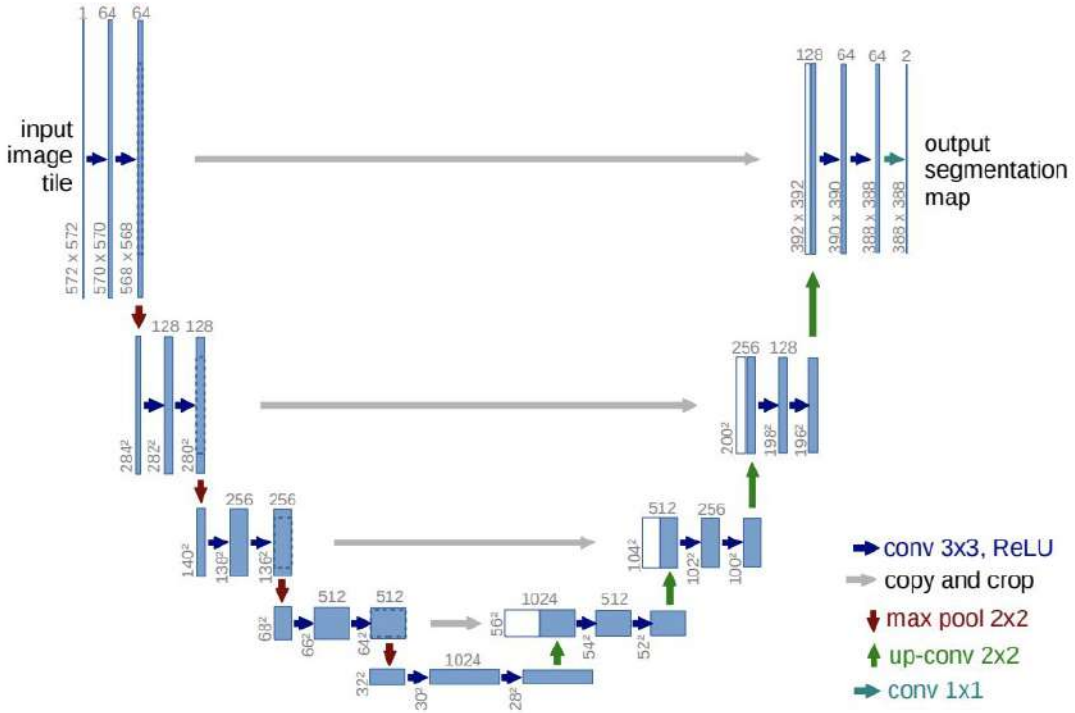
- Denoising diffusion model
- Latent diffusion model
  - encoder/decoder design

- Condition embedding
  - time embedding
  - label 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 prediction type (reparameterization)

- Mean prediction

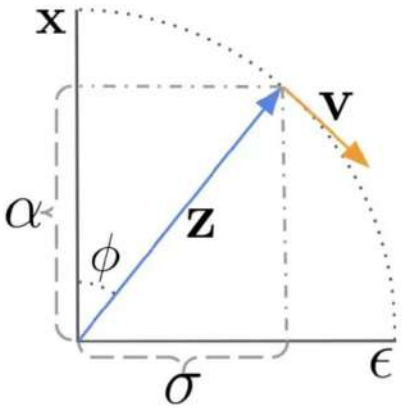
$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

- Noise prediction

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2} \left\| \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\, t\right) \right\|_2^2 \right]$$

- Simplified noise prediction

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\left\| \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\, t\right) \right\|_2^2 \right]$$



- V prediction (Progressive distillation for fast sampling of diffusion models)

$$\alpha_\phi = \cos(\phi), \sigma_\phi = \sin(\phi) \qquad \mathbf{z_t} = \alpha_t \mathbf{x} + \sigma_t \epsilon$$
$$\mathbf{v}_\phi = \alpha_\phi \epsilon - \sigma_\phi \mathbf{x} \qquad \hat{\mathbf{x}} = \alpha_t \mathbf{z}_t - \sigma_t \hat{\mathbf{v}}_\theta(\mathbf{z}_t)$$

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 $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
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**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0

You can include the pretrained evaluator (classifier) here

Simple suggestions (in the other PDF)

3.2 DCGAN

- Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks
- Pytorch DCGAN tutorial

You can follow the above paper and tutorial to learn how to implement an unconditional DCGAN. Further modifying the design into conditional architecture and applying some advanced techniques will help.

3.3 DDPM

- 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 (20%)
    - Describe the details of your two models (10% for each model)
  - Results and discussion (35%)
    - Show your synthetic image grids (12%: 3% * 2 models * 2 testing data) and a denoising process image (3%)
    - Compare the advantages and disadvantages of GAN and DDPM (10%)
    - 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
  - 10% * 2 models * 2 testing data

Accuracy	Grade
$\text{score} \geq 0.8$	100%
$0.8 > \text{score} \geq 0.7$	90%
$0.7 > \text{score} \geq 0.6$	80%
$0.6 > \text{score} \geq 0.5$	70%
$0.5 > \text{score} \geq 0.4$	60%
$\text{score} < 0.4$	0%

Output examples

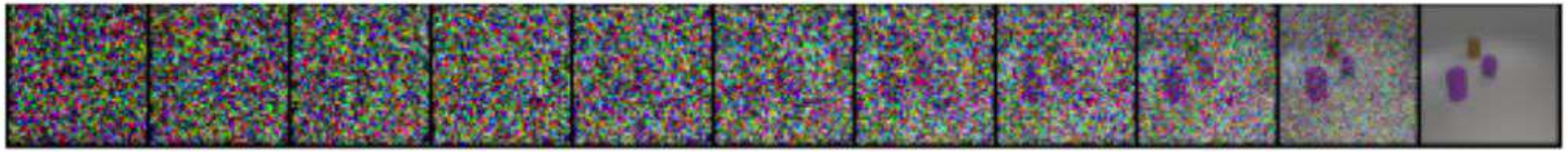


Fig. 2: Example of the denoising process image.

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