DLCV HW2 Report

R11943004 黃子青

Problem 1: Diffusion Models

1.(5%) Describe your implementation details and the difficulties you encountered.

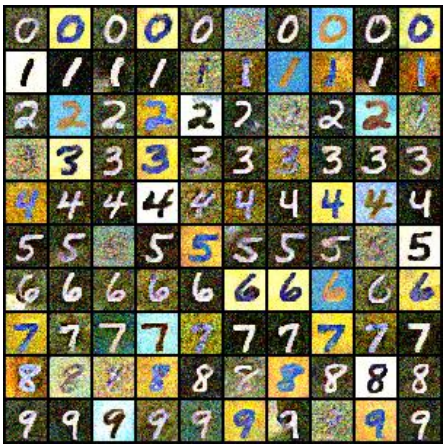
At first, I had no idea how to incorporate time step and context information into the model, let alone how to add the dataset types. Later, after looking it up online, I discovered that embedding is the solution, which also allows the dataset labels to be directly concatenated.
This is the model I referenced: "https://github.com/byrkbrk/conditional-ddpm/blob/main/models.py"
"The detailed approach is as follows: First, in P1_dataloader.py, I load each image along with its label and dataset_type. Then, in P1_training.py, I concatenate the label and dataset types before feeding them into the model."
"During training, I used a context mask to set parts of the context to zero, preventing the model from over-relying on the context values, enabling it to generate images even without context.
For sampling, I used a double batch approach, where each noise is processed separately with and without context. A hyperparameter then determines the weighting for combining the two outputs. The timesteps are set to 500, the learning rate is 0.001, and beta values range from 0.0001 to 0.02, increasing linearly."
The model will generate 10 images for each digit, and the accuracy of prediction will be determined by the digit classifier. I then save the model with the highest accuracy.

```python
def __getitem__(self, idx):
    if torch.is_tensor(idx):
        idx = idx.tolist()

    img_name = os.path.join(self.root_dir, self.dataset_type, 'data', self.data_frame.iloc[idx, 0])
    image = Image.open(img_name).convert('RGB')

    # Get label from CSV file
    label = int(self.data_frame.iloc[idx, 1])
    digit_cond = torch.nn.functional.one_hot(torch.tensor(label), num_classes=10).float()
    if self.transform:
        image = self.transform(image)

    # Generate conditional label for the dataset type: 0 for MNIST-M, 1 for SVHN
    dataset_label = 0 if self.dataset_type == 'mnistm' else 1
    dataset_cond = torch.nn.functional.one_hot(torch.tensor(dataset_label), num_classes=2).float()

    return image, digit_cond, dataset_cond
```

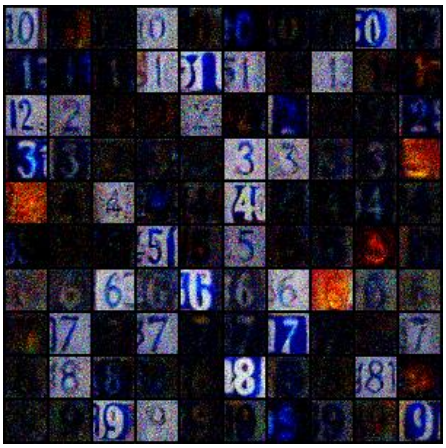
```python
digit_cond_mask = digit_cond_mask.repeat(1, digit_condition.size(1))
dataset_cond_mask = dataset_cond_mask.repeat(1, dataset_condition.size(1))

digit_condition = digit_condition * (1 - digit_cond_mask)
dataset_condition = dataset_condition * (1 - dataset_cond_mask)

context = torch.cat([digit_condition, dataset_condition], dim=1)
t = t_steps / self.timesteps
t = t[:, None]

predict_noise = self.model(x_t, t, context)
```

2.(5%) Please show 10 generated images for each digit (0-9) from both MNIST-M & SVHN dataset in your report. You can put all 100 outputs in one image with columns indicating different noise inputs and rows indicating different digits.

MNIST-M:



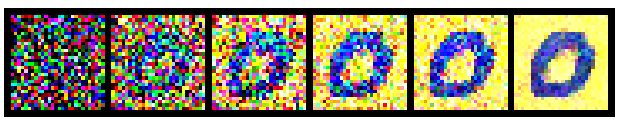
SVHN:



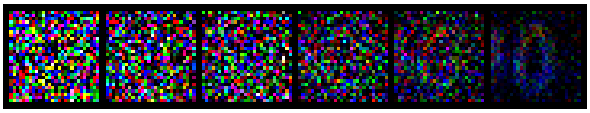
(5%) Visualize a total of six images from both MNIST-M & SVHN datasets in the reverse process of the first "0" in your outputs in (2) and with different time steps. [see the MNIST-M example below, but you need to visualize BOTH MNIST-M & SVHN]

In P1_report.py: timesteps=[1, 50, 100, 200, 300, 500])

MNIST-M:



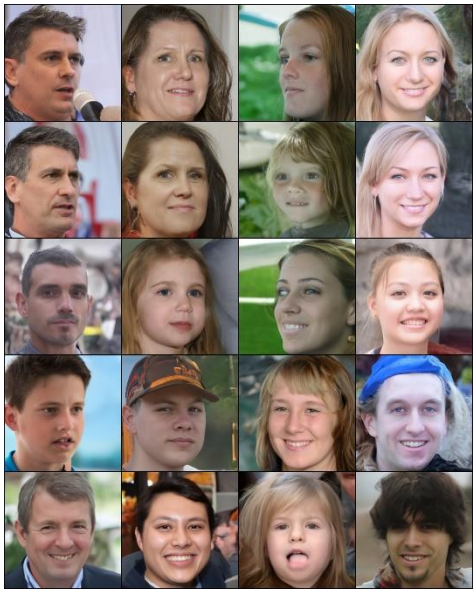
SVHN:



Problem 2: DDIM

1.(7.5%) Please generate face images of noise 00.pt ~ 03.pt with different eta in one grid. Report and explain your observation in this experiment.

"Eta values from top to bottom are 0, 0.25, 0.5, 0.75, and 1."



"When Eta = 0, the variance is 0, resulting in no randomness.(same as the GT images) At Eta=0.25, the variance is still low, so the generated image does not differ much from that with Eta=0.
As Eta increases, reaching Eta=1, the randomness goes up, and the image increasingly diverges from that with Eta=0, even sometimes producing a different gender, and we can clearly see the difference between generated images and GT images.

2.(7.5%) Please generate the face images of the interpolation of noise 00.pt ~ 01.pt. The interpolation formula is spherical linear interpolation, which is also known as slerp. What will happen if we simply use linear interpolation? Explain and report your observation.

spherical linear interpolation:

linear interpolation:

Spherical linear interpolation is ideal in latent spaces of deep learning models because both noise vector direction and magnitude affect the style and features of generated images. In the provided image, slerp smoothly transitions from the male face (00.pt) to the female face (01.pt) as α increases, yielding natural and realistic in-between images.

Linear interpolation, on the other hand, applies a simple weighted average between two noise vectors. In high-dimensional latent spaces, linear interpolation often leads to abrupt or unnatural transitions This is because linear interpolation does not account for changes in noise vector direction and magnitude, resulting in images that are visually distorted or lack realism in the middle of the transition.

Problem 3: Personalization

1.(7.5%) Conduct the CLIP-based zero shot classification on the hw2_data/clip_zeroshot/val, explain how CLIP do this, report the accuracy and 5 successful/failed cases.

In zero-shot classification, CLIP doesn't need training on specific classes in the target dataset. Instead, it matches images with natural language prompts for each class label. The process involves:

Encoding: The image is encoded into a latent space by the image encoder, while each possible class description (e.g., "a photo of a beetle") is encoded by the text encoder.

Similarity Calculation: CLIP computes the similarity (e.g., cosine similarity) between the image embedding and each class embedding.

Classification: The class with the highest similarity score is selected as the predicted label for the image.

Below are examples of successful and failed classifications:

```
doing zero shot classificatin:
Accuracy: 58.64%
Successful Cases: ('20_478.png', 'beetle')
Successful Cases: ('7_489.png', 'fox')
Successful Cases: ('1_481.png', 'chair')
Successful Cases: ('32_494.png', 'bus')
Successful Cases: ('44_481.png', 'elephant')
Failed Cases: ('6_499.png', 'GT:forest', 'Predicted:pine_tree')
Failed Cases: ('13_455.png', 'GT:willow_tree', 'Predicted:oak_tree')
Failed Cases: ('25_451.png', 'GT:wolf', 'Predicted:kangaroo')
Failed Cases: ('30_482.png', 'GT:mushroom', 'Predicted:oak_tree')
Failed Cases: ('13_496.png', 'GT:willow_tree', 'Predicted:pine_tree')
```

The successful cases show CLIP's ability to accurately associate visual features with relevant class descriptions, such as identifying animals and objects (like "bus" and "elephant"). However, the failed cases indicate limitations in CLIP's zero-shot performance, especially with visually similar classes (e.g., "oak_tree" vs. "willow_tree"). This is likely because certain natural objects or animals share visual characteristics that are challenging for CLIP to distinguish without task-specific fine-tuning.

2. (7.5%) What will happen if you simply generate an image containing multiple concepts (e.g., a <new1> next to a <new2>)? You can use your own objects or the provided cat images in the dataset. Share your findings and survey a related paper that works on multiple concepts personalization, and share their method.

I used two prompts, as follows:

Prompt1 = 'a photo of a <new1> in the style of <new2>'

Prompt2 = "A <new1> perched on a park bench with the Taipei 101 behind in the style of <new2>"

The result of the first prompt met expectations, but the result of the second prompt was less than ideal."

I think the possible reason is that the model's generalization ability isn't strong enough. This is because the prompt I used during training was 'a photo of ' so when the test also uses 'a photo of,' the output result is closer to our training data. However, if the prompt becomes increasingly complex, such as in my second prompt, which may include too many concepts at once, the model may struggle to effectively handle multiple concepts personalization for <new1> and <new2> simultaneously.

I found a related paper titled "Concept Conductor: Orchestrating Multiple Personalized Concepts in Text-to-Image Synthesis"

It addresses the challenge of multi-concept personalization in text-to-image generation, enabling the model to represent multiple unique subjects within a single image while preserving each subject's features and maintaining specified spatial

relationships.

Figure 1: Results from existing multi-concept customization methods (second row) and our method (top right). Our method aims to address attribute leakage and layout confusion (concept omission, subject redundancy, appearance truncation), producing visually faithful and text-aligned images.

Method Summary:

1.Independent Concept Embeddings: Fine-tune each subject individually to generate separate embeddings, preventing concept blending and ensuring each subject's unique characteristics are preserved.
2.Attention Modulation for Spatial Control: Use attention mechanisms to control each concept's placement within the image, ensuring the model can position subjects according to text prompts and maintain accurate spatial layout.
3.Dynamic Composition Mechanism: Adjust each concept's influence during the generation process to ensure smooth interaction without interference, achieving natural integration.
4.Prompt Conditioning with Unique Identifiers: Assign unique identifiers to each concept, guiding the model to recognize and incorporate each subject accurately, ensuring that each concept is correctly represented as an individual in the image.

Ref: https://arxiv.org/abs/2408.03632