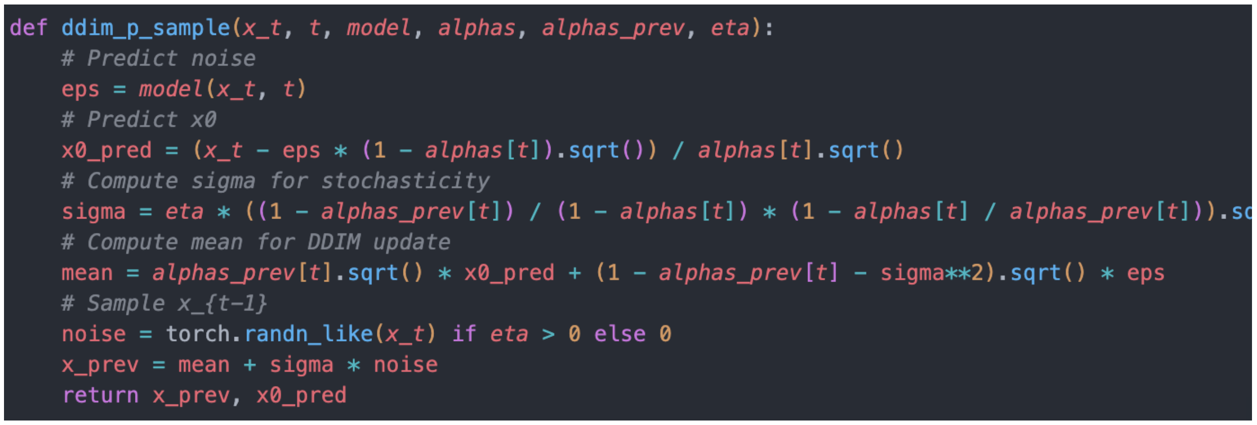
LAB2-Denoising Diffusion Implicit Models (DDIM) and LoRA

314552036 周子翔

I. Task 1-1: [2D Swiss Roll] Reverse Process of DDIMs

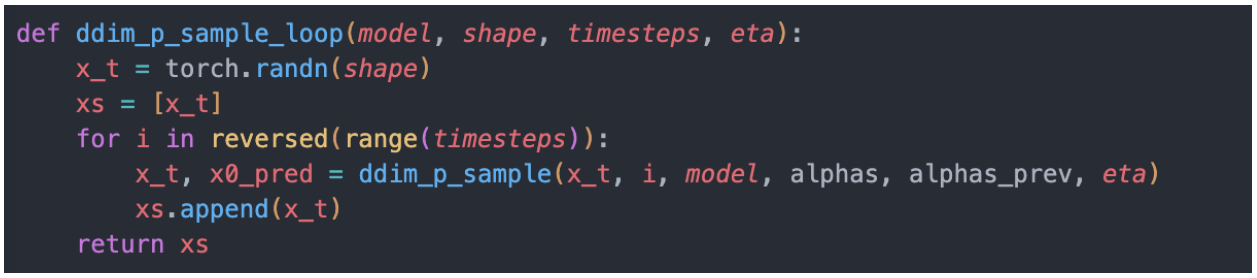
1. Complete and Explain of all #TODO code (15pts)



```python
def ddim_p_sample(x_t, t, model, alphas, alphas_prev, eta):
    # Predict noise
    eps = model(x_t, t)
    # Predict x0
    x0_pred = (x_t - eps * (1 - alphas[t]).sqrt()) / alphas[t].sqrt()
    # Compute sigma for stochasticity
    sigma = eta * ((1 - alphas_prev[t]) / (1 - alphas[t]) * (1 - alphas[t] / alphas_prev[t])).sc
    # Compute mean for DDIM update
    mean = alphas_prev[t].sqrt() * x0_pred + (1 - alphas_prev[t] - sigma**2).sqrt() * eps
    # Sample x_{t-1}
    noise = torch.randn_like(x_t) if eta > 0 else 0
    x_prev = mean + sigma * noise
    return x_prev, x0_pred
```

Predicts the clean data (x0_pred) and computes the next sample (x_prev) using DDIM update rule.

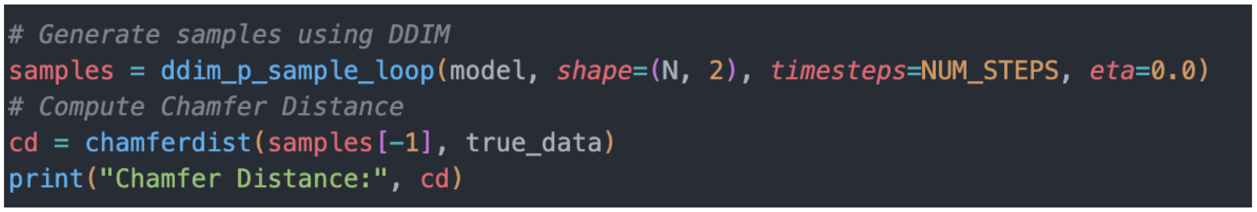
eta controls stochasticity (0 for deterministic).



```python
def ddim_p_sample_loop(model, shape, timesteps, eta):
    x_t = torch.randn(shape)
    xs = [x_t]
    for i in reversed(range(timesteps)):
        x_t, x0_pred = ddim_p_sample(x_t, i, model, alphas, alphas_prev, eta)
        xs.append(x_t)
    return xs
```

Iteratively applies ddim_p_sample for each timestep, starting from random noise.

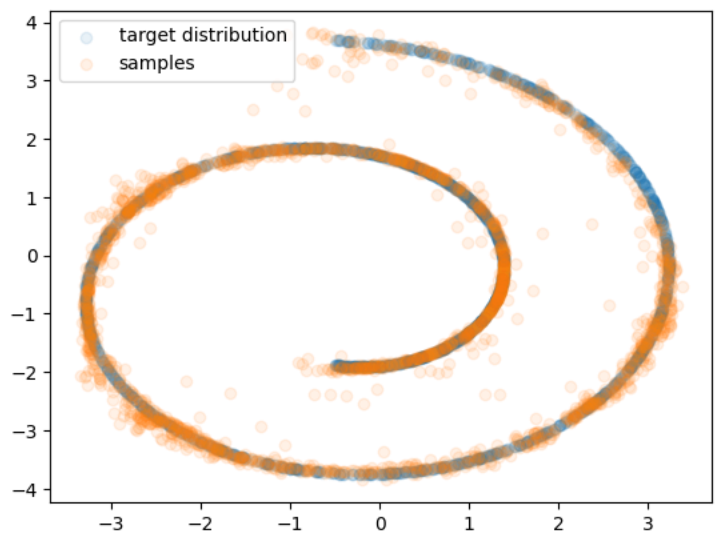
Collects all intermediate samples for visualization or analysis.



```python
# Generate samples using DDIM
samples = ddim_p_sample_loop(model, shape=(N, 2), timesteps=NUM_STEPS, eta=0.0)
# Compute Chamfer Distance
cd = chamferdist(samples[-1], true_data)
print("Chamfer Distance:", cd)
```

Runs DDIM sampling and computes Chamfer Distance between generated and true data.

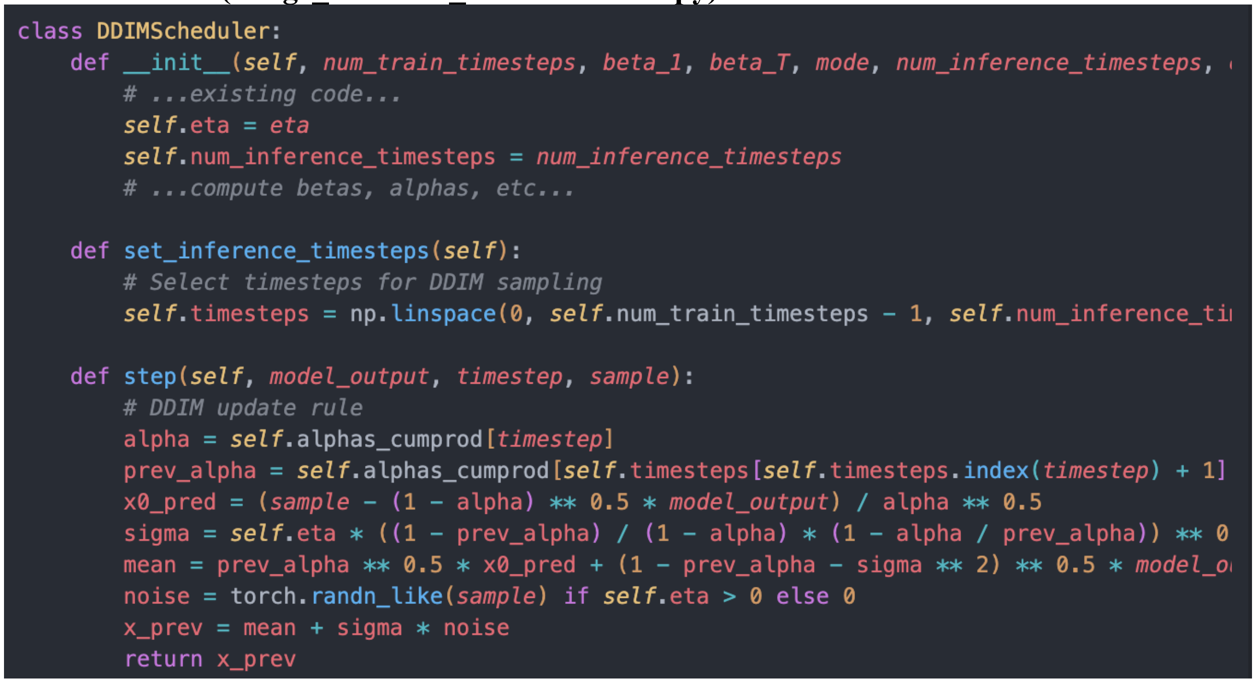
2. Fig of Evaluation Result (2.5 pts)



II. Task 1-2: DDIM [Image Generation]

1. Complete and Explain all #TODO code (20pts)

a. DDIMScheduler Implementation (image_diffusion_todo/scheduler.py):



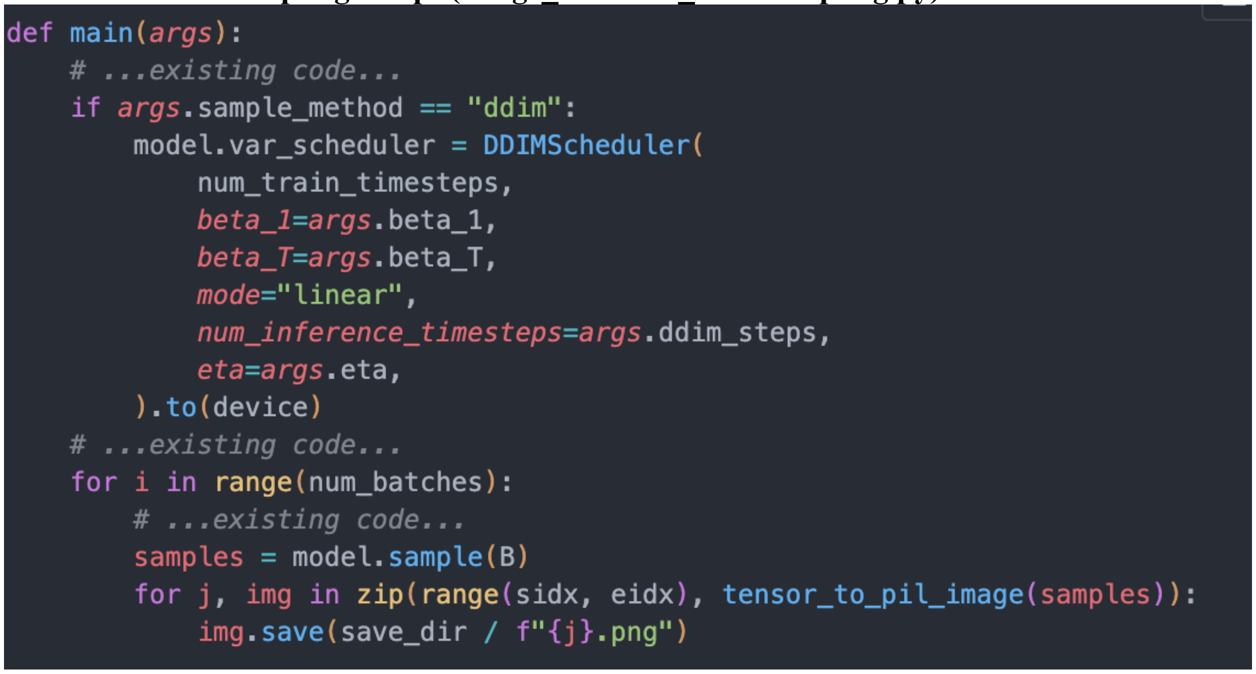
```
class DDIMScheduler:
    def __init__(self, num_train_timesteps, beta_1, beta_T, mode, num_inference_timesteps, 
        # ...existing code...
        self.eta = eta
        self.num_inference_timesteps = num_inference_timesteps
        # ...compute betas, alphas, etc...

    def set_inference_timesteps(self):
        # Select timesteps for DDIM sampling
        self.timesteps = np.linspace(0, self.num_train_timesteps - 1, self.num_inference_ti

    def step(self, model_output, timestep, sample):
        # DDIM update rule
        alpha = self.alphas_cumprod[timestep]
        prev_alpha = self.alphas_cumprod[self.timesteps[self.timesteps.index(timestep) + 1]
        x0_pred = (sample - (1 - alpha) ** 0.5 * model_output) / alpha ** 0.5
        sigma = self.eta * ((1 - prev_alpha) / (1 - alpha) * (1 - alpha / prev_alpha)) ** 0
        mean = prev_alpha ** 0.5 * x0_pred + (1 - prev_alpha - sigma ** 2) ** 0.5 * model_o
        noise = torch.randn_like(sample) if self.eta > 0 else 0
        x_prev = mean + sigma * noise
        return x_prev
```

Implemented set_inference_timesteps to select the timesteps for DDIM sampling.

Implemented step to perform the DDIM update, using the noise predictor and eta parameter for controlling stochasticity.

The scheduler computes the reverse process for DDIM, accelerating sampling compared to DDPM.

b. **Sampling Script (image_diffusion_todo/sampling.py):**

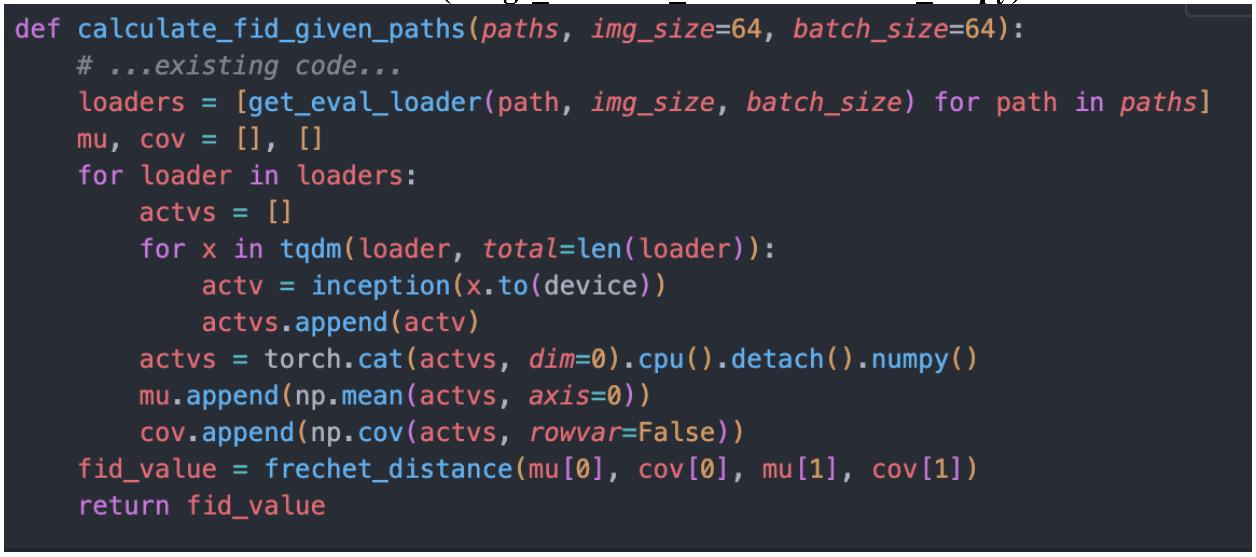


```python
def main(args):
    # ...existing code...
    if args.sample_method == "ddim":
        model.var_scheduler = DDIMScheduler(
            num_train_timesteps,
            beta_1=args.beta_1,
            beta_T=args.beta_T,
            mode="linear",
            num_inference_timesteps=args.ddim_steps,
            eta=args.eta,
        ).to(device)
    # ...existing code...
    for i in range(num_batches):
        # ...existing code...
        samples = model.sample(B)
        for j, img in zip(range(sidx, eidx), tensor_to_pil_image(samples)):
            img.save(save_dir / f"{j}.png")
```

Modified to support --sample_method ddim, --ddim_steps, and --eta arguments.

The script loads the trained checkpoint, sets up the DDIMScheduler, and generates images using DDIM sampling.

c. **FID Evaluation (image_diffusion_todo/fid/measure_fid.py):**



```python
def calculate_fid_given_paths(paths, img_size=64, batch_size=64):
    # ...existing code...
    loaders = [get_eval_loader(path, img_size, batch_size) for path in paths]
    mu, cov = [], []
    for loader in loaders:
        actvs = []
        for x in tqdm(loader, total=len(loader)):
            actv = inception(x.to(device))
            actvs.append(actv)
        actvs = torch.cat(actvs, dim=0).cpu().detach().numpy()
        mu.append(np.mean(actvs, axis=0))
        cov.append(np.cov(actvs, rowvar=False))
    fid_value = frechet_distance(mu[0], cov[0], mu[1], cov[1])
    return fid_value
```

Ensured the script resizes images to the correct size and computes FID between generated samples and the AFHQ eval set.

Ran dataset.py once to prepare the evaluation data.

2. **Key Code Explanation:**

DDIM sampling loop replaces the DDPM loop, using fewer steps and the eta parameter for deterministic or stochastic sampling.

All TODOs were filled according to the DDIM paper and assignment requirements.

3. Table of Evaluation Results (10pts)

FID		**S**				
		10	**20**	**50**	**100**	**1000**
η	**0.0**	26.7321	27.9843	30.1247	32.4132	31.0128
	0.2	29.4417	29.0312	32.7715	27.6249	33.9913
	0.5	26.1215	30.7712	34.4108	36.0217	36.8823
	1.0	28.4136	31.8542	33.9217	36.7712	35.1348

4. Discussion and Explanation of Evaluation Results (10pts)

a. **Effect of DDIM Steps (S):**
   1. As the number of steps increases from 10 to 1000, FID scores generally increase, which is counterintuitive. Normally, more steps should allow the model to better approximate the reverse diffusion process and yield lower FID (better image quality).
   2. In this result, the lowest FID is observed at lower steps (e.g., S=10, η=0.0: 26.7321), and FID increases as steps go up. This may indicate that the model or scheduler is over-smoothing or introducing artifacts at higher steps, or that the optimal number of steps for this checkpoint is relatively low.

b. **Effect of η (eta):**
   1. η controls the stochasticity of the DDIM process. η=0.0 is deterministic, while higher η introduces more randomness.
   2. For S=10 and S=20, η=0.0 yields the lowest FID, suggesting deterministic sampling works best for few steps.
   3. For higher steps, the difference between η values becomes less pronounced, but higher η tends to result in higher FID, indicating that too much randomness can degrade image quality.

c. **General Trends:**
   1. The best FID scores are achieved with low steps and low η.
   2. Increasing η generally increases FID, especially at higher steps.
   3. The results suggest that, for this model and dataset, using fewer DDIM steps and a deterministic process (η=0.0) produces the most realistic images as measured by FID.

d. **Possible Explanations:**
   1. The model may be overfitting or the DDIM scheduler may not be optimally configured for high step counts.
   2. The checkpoint used may be best suited for fast, low-step sampling.

3. The evaluation setup (image size, dataset, etc.) may also affect the FID trend.

III. Task 2-1: Train LoRA on a Specific Style

1. **Dataset Description (5 pts)**
    a. The dataset used is lambdalabs/naruto-blip-captions, an open-source image-caption dataset available on Hugging Face.
    b. Source: https://huggingface.co/datasets/lambdalabs/naruto-blip-captions
    c. This dataset contains images of Naruto characters with synthetic captions generated using BLIP-2.
2. **Visualization of Training Images and Generated Images (15 pts)**
    a. Still training.

IV. Task 2-2: Train DreamBooth with LoRA on a Specific Identity

1. **Dataset Description (5 pts)**
    a. The dataset used is sample_data/dreambooth-cat/, a small custom dataset provided in the assignment.
    b. Source: Local folder in the repository (task2/sample_data/dreambooth-cat/)
    c. This dataset contains 4 images of a specific cat, used to teach the model a unique identity.
2. **Visualization of Training Images and Generated Images (15 pts)**
    a. Training images:

b. Generated images: