Homework 1

Task1

В этой секции описаны шаги по отладке кода. Я начал с тестирования unet и диффузии, затем перешел к отладке пайплайна.

Тест: test_unet

```
pytest -q test_model.py -k "test_unet"
```

Тест завершился с ошибкой, теперь можно приступать к исправлению кода:

```
    def test_unet(input_tensor, num_timesteps):
        B, C, H, W = input_tensor.shape
        net = UnetModel(C, C, hidden_size=128)
        timestep = torch.randint(1, num_timesteps + 1, (B,)) / num_timesteps
>       out = net(input_tensor, timestep)

...

def forward(self, x: torch.Tensor, skip: torch.Tensor) -> torch.Tensor:
>       x = torch.cat((x, skip), 1)
E       RuntimeError: Sizes of tensors must match except in dimension 1.
Expected size 8 but got size 4 for tensor number 1 in the list.
```

Первая ошибка в строке 109 файла unet.py:

```
temb = self.timestep_embedding(t)[:, :, None, None]
```

После данного исправления тест прошел.

Тест: test_diffusion

```
pytest -q test_model.py -k "test_diffusion"
```

Ошибка:

```
F                                                    [100%]
=================================== FAILURES ===============
_____________________________ test_diffusion ______________________________

num_channels = 3, batch_size = 4

    def test_diffusion(num_channels=3, batch_size=4):
        # note: you should not need to change the thresholds or the hyperparamete
        net = UnetModel(num_channels, num_channels, hidden_size=128)
        model = DiffusionModel(eps_model=net, betas=(1e-4, 0.02), num_timesteps

        input_data = torch.randn((batch_size, num_channels, 32, 32))

        output = model(input_data)
        assert output.ndim == 0
>       assert 1.0 <= output <= 1.2
E       assert 1.0 <= tensor(0.1760, grad_fn=<MseLossBackward0>)

test_model.py:49: AssertionError
=========================== short test summary info ============
FAILED test_model.py::test_diffusion - assert 1.0 <= tensor(0.1760, grad_fn=<...
1 failed, 4 deselected in 2.78s
```

Для выявления бага, я выводил нормы x, x_t и eps. В случае variance preserving схемы они должны быть одного порядка. Вторая ошибка была очевидной, так как параметры расписания должны суммироваться в единицу в variance preserving схеме.

Ошибки были в функции forward на строчке 25:

```
eps = torch.randn_like(x)
```

и на строчке 29:

```
+ self.sqrt_one_minus_alpha_prod[timestep, None, None, None] * eps
```

Тест: test_train_on_one_batch

```
pytest -q test_pipeline.py -k "test_train_on_one_batch"
```

Ошибка была в строке 24 файла diffusion.py:

```
timestep = torch.randint(1, self.num_timesteps + 1, (x.shape[0],), device=x.device
```

Тест: test_training

На начало выполнения этой задачи покрытия modeling.training составлял 56%.

Для тестирования различных конфигураций на CPU и GPU используются следующие стратегии:

- CPU: модель меньшего размера, меньше временных шагов, меньший размер батча
- GPU: модель большего размера, больше временных шагов (пропускается, если CUDA недоступна)

Тест включает анализ следующих компонентов:

- Инициализация модели
- Шаг обучения
- Эпоха обучения
- Генерация сэмплов
- Улучшение функции потерь
- Функционал сохранения изображений

В процессе тестирования была найдена ошибка на строках 35, 38 файла diffusion.py:

```
x_i = torch.randn(num_samples, *size, device=device)
z = torch.randn(num_samples, *size, device=device) if i > 1 else 0
```

```
pytest --cov=modeling tests/
```

Покрытие до и после добавления теста:

	До	После
modeling/init.py	100%	100%
modeling/diffusion.py	82%	100%
modeling/training.py	56%	89%
modeling/unet.py	100%	100%
Total	86%	97%

Task 2

Я внедрил wandb и hydra, после чего обучил модель на протяжении 100 эпох.

Также добавил фиксирование сида перед обучением. Изображения логируются в wandb. Добавил отдельную функцию создания dataloader, также добавил аугментацию RandomHorizontalFlip.

Также я исправил функцию диффузионной генерации и сделал ее более аккуратной, так как она была реализована с ошибкой.

Также я добавил в необходимые для тестирования функции аргумент debug, чтобы wandb не логировал данные во время тестирования.

При настройке Hydra я заменил константы на параметры из конфигов.

Wandb project: https://wandb.ai/vmeshchaninov/EfDL-week02-homework

Wandb exp: https://wandb.ai/vmeshchaninov/EfDL-week02-homework/runs/2f0k3wvl?nw=nwuservmeshchaninov

Пример генерации:

Task 3

Я внедрил hydra и создал следующую иерархию конфигов:

```
conf/                 # Hydra config directory
├── config.yaml       # Main config file
├── model/            # Model architectures
│   ├── unet.yaml
├── dataset/          # Dataset settings
│   ├── cifar10.yaml
├── training/         # Training hyperparameters
│   ├── base.yaml
├── optimizer/        # Optimizer configs
│   ├── adam.yaml
│   ├── sgd.yaml
```

```
├── diffusion/          # Diffusion parameters
│   ├── diffusion.yaml
```

Конфигурация позволяет настроить все заданные атрибуты.

Логирование конфига происходит при помощи run.log_artifact.

Для демонстрации я запустил три варианта обучения:

```
python3 main.py training.num_epochs=5  optimizer=adam

python3 main.py training.num_epochs=5  optimizer=sgd

python3 main.py training.num_epochs=5  optimizer=sgd optimizer.lr=5e-4
```

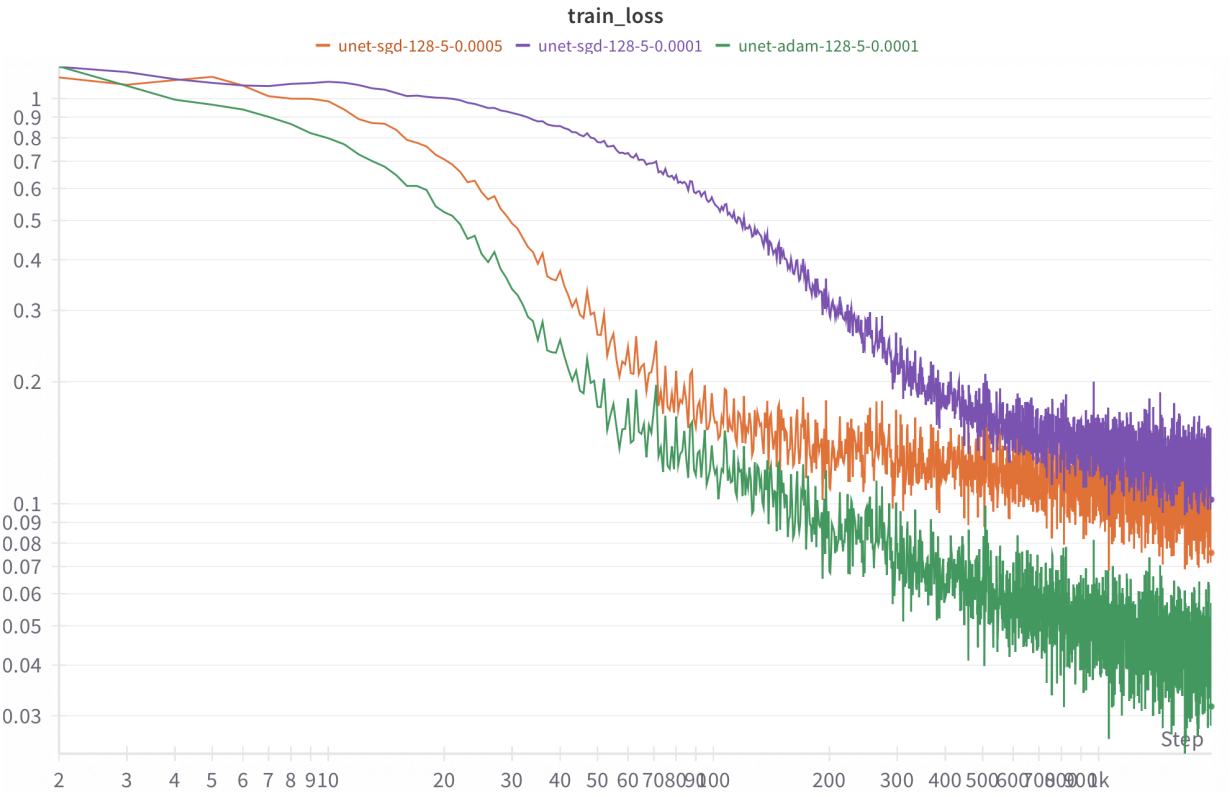
Wandb exps:

https://wandb.ai/vmeshchaninov/EfDL-week02-homework/runs/mvr26m3i

https://wandb.ai/vmeshchaninov/EfDL-week02-homework/runs/9yot79s6

https://wandb.ai/vmeshchaninov/EfDL-week02-homework/runs/eaqz7y9t

График функции потерь наглядно демонстрирует, как различные параметры влияют на процесс обучения:


train_loss
unet-sgd-128-5-0.0005
unet-sgd-128-5-0.0001
unet-adam-128-5-0.0001
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
Step
2 3 4 5 6 7 8 9 10 20 30 40 50 60 70 80 90 100 200 300 400 500 600 700 800 900 1k