Magic Un-Eraser

Generated by Doxygen 1.9.8

1 Magic Un-Eraser 1
1.1 Description . . . 1
1.2 Usage / Re-Production . . . 2
1.2.1 1. Set-up . . . 2
1.2.2 2. Training . . . 2
1.2.3 3. Evaluation & Plotting . . . 2
1.3 Timing . . . 3

2 Namespace Index 5
2.1 Package List . . . 5

3 Hierarchical Index 7
3.1 Class Hierarchy . . . 7

4 Class Index 9
4.1 Class List . . . 9

5 File Index 11
5.1 File List . . . 11

6 Namespace Documentation 13
6.1 degradation Namespace Reference . . . 13
6.1.1 Detailed Description . . . 13
6.2 eval Namespace Reference . . . 13
6.2.1 Variable Documentation . . . 14
6.2.1.1 _ . . . 14
6.2.1.2 args . . . 14
6.2.1.3 cfg . . . 14
6.2.1.4 dataloader . . . 14
6.2.1.5 default . . . 14
6.2.1.6 device . . . 14
6.2.1.7 fid . . . 14
6.2.1.8 fid_score . . . 14
6.2.1.9 grid . . . 14
6.2.1.10 help . . . 14
6.2.1.11 int . . . 15
6.2.1.12 model . . . 15
6.2.1.13 n_params . . . 15
6.2.1.14 parser . . . 15
6.2.1.15 real . . . 15
6.2.1.16 real_images . . . 15
6.2.1.17 save_path . . . 15
6.2.1.18 SEED . . . 15
6.2.1.19 state_dict . . . 15

6.2.1.20 str . . . 15
6.2.1.21 t0 . . . 16
6.2.1.22 testset . . . 16
6.2.1.23 tf . . . 16
6.2.1.24 transform_for_inceptionv3 . . . 16
6.2.1.25 type . . . 16
6.2.1.26 x_gen . . . 16
6.3 make_plots Namespace Reference . . . 16
6.3.1 Function Documentation . . . 17
6.3.1.1 plot_loss() . . . 17
6.3.1.2 save_fig() . . . 18
6.3.2 Variable Documentation . . . 18
6.3.2.1 args . . . 18
6.3.2.2 axs . . . 18
6.3.2.3 background . . . 18
6.3.2.4 cmap . . . 18
6.3.2.5 colour . . . 18
6.3.2.6 dataset . . . 18
6.3.2.7 degrador . . . 18
6.3.2.8 epochs . . . 18
6.3.2.9 eraserhead . . . 19
6.3.2.10 eraserheads . . . 19
6.3.2.11 fig . . . 19
6.3.2.12 figsize . . . 19
6.3.2.13 fontsize . . . 19
6.3.2.14 fontweight . . . 19
6.3.2.15 help . . . 19
6.3.2.16 img . . . 19
6.3.2.17 index . . . 19
6.3.2.18 letter . . . 19
6.3.2.19 letters . . . 20
6.3.2.20 mask . . . 20
6.3.2.21 masks . . . 20
6.3.2.22 metric_logger . . . 20
6.3.2.23 models . . . 20
6.3.2.24 noise . . . 20
6.3.2.25 parser . . . 20
6.3.2.26 path . . . 20
6.3.2.27 pixels . . . 20
6.3.2.28 s . . . 20
6.3.2.29 str . . . 21
6.3.2.30 tf . . . 21

6.3.2.31 trajectory . . . 21
6.3.2.32 trajectory_img . . . 21
6.3.2.33 transAxes . . . 21
6.3.2.34 transform . . . 21
6.3.2.35 type . . . 21
6.3.2.36 x . . . 21
6.4 src Namespace Reference . . . 21
6.4.1 Detailed Description . . . 22
6.5 src.cnn Namespace Reference . . . 22
6.6 src.config_parsing Namespace Reference . . . 22
6.6.1 Function Documentation . . . 22
6.6.1.1 parse_config() . . . 22
6.6.1.2 parse_degradation_config() . . . 23
6.7 src.degradation Namespace Reference . . . 23
6.8 src.degradation.base Namespace Reference . . . 23
6.9 src.degradation.eraser Namespace Reference . . . 23
6.10 src.degradation.eraser_utils Namespace Reference . . . 24
6.10.1 Function Documentation . . . 24
6.10.1.1 all_pixels_in_trajectory() . . . 24
6.10.1.2 disk_mask() . . . 24
6.10.1.3 eraser_trajectory() . . . 25
6.10.1.4 eraserhead_masks() . . . 25
6.10.1.5 gaussian_mask() . . . 25
6.10.1.6 get_zigzag_trajectory() . . . 26
6.11 src.models Namespace Reference . . . 26
6.12 src.noise_schedules Namespace Reference . . . 26
6.12.1 Function Documentation . . . 27
6.12.1.1 cosine_schedule() . . . 27
6.12.1.2 linear_schedule() . . . 27
6.13 train Namespace Reference . . . 27
6.13.1 Variable Documentation . . . 28
6.13.1.1 accelerator . . . 28
6.13.1.2 cfg . . . 28
6.13.1.3 config_file . . . 28
6.13.1.4 device . . . 28
6.13.1.5 exist_ok . . . 28
6.13.1.6 grid . . . 28
6.13.1.7 loss . . . 28
6.13.1.8 losses . . . 28
6.13.1.9 metric_logger . . . 28
6.13.1.10 model . . . 28
6.13.1.11 optim . . . 29

6.13.1.12 results_dir . . . 29
6.13.1.13 SEED . . . 29
6.13.1.14 t0 . . . 29
6.13.1.15 test_loader . . . 29
6.13.1.16 test_loss . . . 29
6.13.1.17 testset . . . 29
6.13.1.18 tf . . . 29
6.13.1.19 total_loss . . . 29
6.13.1.20 train_loader . . . 29
6.13.1.21 train_loss . . . 30
6.13.1.22 trainset . . . 30
6.13.1.23 xh . . . 30

7 Class Documentation **31**
7.1 src.cnn.CNN Class Reference . . . 31
7.1.1 Constructor & Destructor Documentation . . . 31
7.1.1.1 __init__() . . . 31
7.1.2 Member Function Documentation . . . 32
7.1.2.1 forward() . . . 32
7.1.2.2 time_encoding() . . . 32
7.1.3 Member Data Documentation . . . 32
7.1.3.1 blocks . . . 32
7.1.3.2 time_embed . . . 32
7.2 src.cnn.CNNBlock Class Reference . . . 33
7.2.1 Constructor & Destructor Documentation . . . 33
7.2.1.1 __init__() . . . 33
7.2.2 Member Function Documentation . . . 33
7.2.2.1 forward() . . . 33
7.2.3 Member Data Documentation . . . 33
7.2.3.1 net . . . 33
7.3 src.models.ColdDiffusion Class Reference . . . 34
7.3.1 Detailed Description . . . 34
7.3.2 Constructor & Destructor Documentation . . . 34
7.3.2.1 __init__() . . . 34
7.3.3 Member Function Documentation . . . 35
7.3.3.1 forward() . . . 35
7.3.3.2 sample() . . . 35
7.3.4 Member Data Documentation . . . 35
7.3.4.1 criterion . . . 35
7.3.4.2 degrador . . . 35
7.3.4.3 reconstructor . . . 35
7.3.4.4 T . . . 35

7.4 src.models.DDPM Class Reference . . . 36
7.4.1 Detailed Description . . . 36
7.4.2 Constructor & Destructor Documentation . . . 36
7.4.2.1 __init__() . . . 36
7.4.3 Member Function Documentation . . . 36
7.4.3.1 forward() . . . 36
7.4.3.2 sample() . . . 37
7.4.4 Member Data Documentation . . . 37
7.4.4.1 criterion . . . 37
7.4.4.2 gt . . . 37
7.4.4.3 T . . . 37
7.5 src.degradation.base.DegredationOperator Class Reference . . . 37
7.5.1 Detailed Description . . . 38
7.5.2 Constructor & Destructor Documentation . . . 38
7.5.2.1 __init__() . . . 38
7.5.3 Member Function Documentation . . . 38
7.5.3.1 sampling() . . . 38
7.5.3.2 train() . . . 38
7.5.4 Member Data Documentation . . . 39
7.5.4.1 sampling_mode . . . 39
7.6 src.degradation.eraser.Eraser Class Reference . . . 39
7.6.1 Detailed Description . . . 40
7.6.2 Constructor & Destructor Documentation . . . 40
7.6.2.1 __init__() . . . 40
7.6.3 Member Function Documentation . . . 40
7.6.3.1 __call__() . . . 40
7.6.3.2 latent_sampler() . . . 40
7.6.4 Member Data Documentation . . . 40
7.6.4.1 background . . . 40
7.6.4.2 noise_std . . . 40

8 File Documentation **41**

8.1 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/eval.py File Reference . . . 41
8.1.1 Detailed Description . . . 42
8.2 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/make_plots.py File Reference . . . 42
8.2.1 Detailed Description . . . 43
8.3 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/README.md File Reference . . . 43
8.4 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/__init__.py File Reference . . . 43
8.5 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/__init__.py File Reference 44
8.6 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/cnn.py File Reference . . . 44
8.6.1 Detailed Description . . . 44
8.7 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/config_parsing.py File Reference . . . 44

8.7.1 Detailed Description 45
8.8 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/base.py File Reference . 45
8.8.1 Detailed Description 45
8.9 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/eraser.py File Reference . 46
8.9.1 Detailed Description 46
8.10 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/eraser_utils.py File Reference 46
8.10.1 Detailed Description 47
8.11 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/models.py File Reference 47
8.11.1 Detailed Description 48
8.12 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/noise_schedules.py File Reference . 48
8.12.1 Detailed Description 48
8.13 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/train.py File Reference 48
8.13.1 Detailed Description 49

Index **51**

Chapter 1

Magic Un-Eraser

1.1 Description

In this project we train a diffusion model to generate handwritten digits by reversing the custom 'Eraser' image degradation process. As a baseline, we also train two denoising diffusion probabilistic models (DDPM - standard diffusion).

Toggle Degradation/Reconstruction

Degradation

Generation

Project structure

- `configs/` - Contains model configurations of the 3 models trained in this project.
- `data/` - Contains the MNIST dataset.
- `docs/` - Contains documentation for the project.
- `models/` - Contains subdirectories for each trained model (automatically generated by the `train.py` script). Each sub-directory contains the model state dictionary, the configuration used to train it, the metric logger, and a subdirectory of samples generated throughout the training process.
- `plots/` - Contains the plots used in the report, and other visualisations.
- `src/` - Contains the source code for the project. (Re-usable components that are used by the scripts in the root directory)
- `eval.py` - Script for evaluating the trained models.
- `make_plots.py` - Script for generating plots used in the report.
- `train.py` - Script for training the models.
- `.gitignore` - Tells git which files to ignore
- `.pre-commit-config.yaml` - Specifies pre-commit hooks to protect the `main` branch
- `Dockerfile` - Dockerfile to generate docker image
- `requirements.txt` - List of packages/versions to re-create the environment for the project.
- `LICENSE` - MIT license.

1.2 Usage / Re-Production

Note: All commands assume they are being run from the root directory of the project.

1.2.1 1. Set-up

To re-create the environment used for the project, use the `requirements.txt` file. This can be done with `pip`, `conda` or docker. The docker container will not naturally have access to the `mps` device on Mac laptops (which was used to train the models), thus for best performance it is recommended to use `pip` or `conda` to re-createe the environment.

pip
```
$ pip install -r requirements.txt
```

Using conda
```
$ conda create --name <env-name> python
$ conda activate <env-name>
$ pip install -r requirements.txt
```

Docker (not recommended)
```
$ docker build -t <image-name> .
$ docker run -ti <image-name> bash
```

1.2.2 2. Training

We trained three models: DDPM model with default hyperparameters (low-capacity), DDPM model with high-capacity (twice the number of trainable parameters) and a Cold diffusion model (termed the 'Magic Un-Eraser') using the 'Eraser' degradation strategy. The Cold diffusion model had all the same hyperparameters as the default DDPM.
```
$ python train.py ./configs/ddpm_default.ini    # default DDPM model
$ python train.py ./configs/ddpm_high.ini       # high capacity DDPM model
$ python train.py ./configs/magic_uneraser.ini  # Cold diffusion model ("Magic Un-Eraser")
```

1.2.3 3. Evaluation & Plotting

To evaluate the models, we generate 100 samples and calculate the FID score between the samples and 500 images from the MNIST test set. Use the following commands:
```
$ python eval.py --model_dir ./models/ddpm_default --output_dir ./plots
$ python eval.py --model_dir ./models/ddpm_high --output_dir ./plots
$ python eval.py --model_dir ./models/magic_uneraser --output_dir ./plots
```

To make the plots, use
```
$ python make_plots.py --models ./models --output_dir ./plots
```

1.3 Timing

Times to run each script:

- `train.py` - Using the `mps` device, it took ∼25 minutes to train the high-capacity DDPM model, and ∼15 minutes to train the other two models.
- `eval.py` - No more than 200 seconds for each trained model. (Used the `cpu` only)
- `make_plots.py` - No more than 1-2 minutes.

I ran all scripts on my personal laptop. The `train.py` script used the `mps` device, which is essentially the macbook GPU. The specifications are:

- Operating System: macOS Sonoma v14.0

CPU:

- Chip: Apple M1 Pro
- Total Number of Cores: 8 (6 performance and 2 efficiency)
- Memory (RAM): 16 GB

GPU (`mps`):

- Chipset Model: Apple M1 Pro
- Type: GPU
- Bus: Built-In
- Total Number of Cores: 14
- Metal Support: Metal 3

Chapter 2

Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

degradation
This package contains the degradation operators for the project . . . 13
eval . . . 13
make_plots . . . 16
src
This package contains the source code for the project . . . 21
src.cnn . . . 22
src.config_parsing . . . 22
src.degradation . . . 23
src.degradation.base . . . 23
src.degradation.eraser . . . 23
src.degradation.eraser_utils . . . 24
src.models . . . 26
src.noise_schedules . . . 26
train . . . 27

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

nn.Module
- src.cnn.CNN . . . 31
- src.cnn.CNNBlock . . . 33
- src.degradation.base.DegredationOperator . . . 37
  - src.degradation.eraser.Eraser . . . 39
- src.models.ColdDiffusion . . . 34
- src.models.DDPM . . . 36

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

src.cnn.CNN 31
src.cnn.CNNBlock 33
src.models.ColdDiffusion
    Cold Diffusion model as described by Bansal et al 34
src.models.DDPM
    Denoising Diffusion Probabilistic Model (DDPM) as described by Ho et al 36
src.degradation.base.DegredationOperator
    Base class for all degradation operators 37
src.degradation.eraser.Eraser
    Eraser Degredation Operator 39

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/eval.py
Evaluation script for trained diffusion model . . . 41
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/make_plots.py
Script to make plots for the report . . . 42
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/train.py
Script to train diffusion model on MNIST dataset . . . 48
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/__init__.py . . . 43
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/cnn.py
We create a simple 2D convolutional neural network to use as the reconstructor network within the diffusion models . . . 44
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/config_parsing.py
Functions for parsing the model configuration files . . . 44
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/models.py
Diffusion models for image generation . . . 47
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/noise_schedules.py
Functions for generating noise schedules for DDPM sampling . . . 48
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/__init__.py . . . 44
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/base.py
Contains base class for all degradation operators . . . 45
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/eraser.py
Contains the Eraser Degredation Operator . . . 46
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/eraser_utils.py
Contains all utility functions for the Eraser Degredation Operator . . . 46

Chapter 6

Namespace Documentation

6.1 degradation Namespace Reference

This package contains the degradation operators for the project.

6.1.1 Detailed Description

This package contains the degradation operators for the project.

6.2 eval Namespace Reference

Variables

- int SEED = 1
- t0 = time()
- parser = argparse.ArgumentParser()
- type
- str
- help
- default
- int
- args = parser.parse_known_args()[0]
- device = torch.device("cpu")
- cfg
- model
- _
- state_dict = torch.load(join(args.model_dir, "state_dict.pth"), map_location=device)
- n_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
- x_gen = model.sample(args.n_samples, (1, 28, 28), device)
- grid = make_grid(x_gen, nrow=int(np.sqrt(args.n_samples)))
- save_path = join(args.output_dir, f"{cfg['model_name']}_samples.png")
- tf

  FID Score.
- testset = MNIST("./data", train=False, download=True, transform=tf)
- dataloader = DataLoader(testset, batch_size=500, shuffle=True)
- transform_for_inceptionv3
- fid = FrechetInceptionDistance(feature=2048)
- real_images = transform_for_inceptionv3(real_images)
- real
- fid_score = fid.compute()

6.2.1 Variable Documentation

6.2.1.1 _

```
eval._  [protected]
```

6.2.1.2 args

```
eval.args = parser.parse_known_args()[0]
```

6.2.1.3 cfg

```
eval.cfg
```

6.2.1.4 dataloader

```
eval.dataloader = DataLoader(testset, batch_size=500, shuffle=True)
```

6.2.1.5 default

```
eval.default
```

6.2.1.6 device

```
eval.device = torch.device("cpu")
```

6.2.1.7 fid

```
eval.fid = FrechetInceptionDistance(feature=2048)
```

6.2.1.8 fid_score

```
eval.fid_score = fid.compute()
```

6.2.1.9 grid

```
eval.grid = make_grid(x_gen, nrow=int(np.sqrt(args.n_samples)))
```

6.2.1.10 help

```
eval.help
```

6.2.1.11 int

```
eval.int
```

6.2.1.12 model

```
eval.model
```

6.2.1.13 n_params

```
eval.n_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
```

6.2.1.14 parser

```
eval.parser = argparse.ArgumentParser()
```

6.2.1.15 real

```
eval.real
```

6.2.1.16 real_images

```
eval.real_images = transform_for_inceptionv3(real_images)
```

6.2.1.17 save_path

```
eval.save_path = join(args.output_dir, f"{cfg['model_name']}_samples.png")
```

6.2.1.18 SEED

```
int eval.SEED = 1
```

6.2.1.19 state_dict

```
eval.state_dict = torch.load(join(args.model_dir, "state_dict.pth"), map_location=device)
```

6.2.1.20 str

```
eval.str
```

6.2.1.21 t0

```
eval.t0 = time()
```

6.2.1.22 testset

```
eval.testset = MNIST("./data", train=False, download=True, transform=tf)
```

6.2.1.23 tf

```
eval.tf
```

Initial value:

```
00001 =  transforms.Compose(
00002     [
00003         transforms.ToTensor(),
00004         transforms.Normalize((0.5,), (1.0)),
00005     ]
00006 )
```

FID Score.

6.2.1.24 transform_for_inceptionv3

```
eval.transform_for_inceptionv3
```

Initial value:

```
00001 =  transforms.Compose(
00002     [
00003         transforms.Resize((299, 299)),  # Resize the images to 299x299
00004         transforms.Lambda(lambda x: x.mul(255).byte()),  # Convert to 0-255
00005         transforms.Lambda(lambda x: x.repeat(1, 3, 1, 1)),  # Repeat 1-channel to 3-channel
00006     ]
00007 )
```

6.2.1.25 type

```
eval.type
```

6.2.1.26 x_gen

```
eval.x_gen = model.sample(args.n_samples, (1, 28, 28), device)
```

6.3 make_plots Namespace Reference

Functions

- plot_loss (metric_logger)

  Plot the training and test loss over epochs.
- save_fig (fig, save_path)

  Save the figure to the specified path.

Variables

- parser = argparse.ArgumentParser()
- type
- str
- help
- args = parser.parse_known_args()[0]
- list models = ["ddpm_default", "ddpm_high", "magic_uneraser"]

  Training visualisations.
- metric_logger = pickle.load(f)
- fig = plot_loss(metric_logger)
- list epochs = [1, 5, 10, 25]
- list letters = ["(a)", "(b)", "(c)", "(d)"]
- axs = axs.flatten()
- figsize
- path = join(args.models, model_name, "samples", f"epoch_{epoch - 1:04d}.png")
- img = Image.open(path)
- fontsize
- transform
- transAxes
- fontweight
- trajectory = eraser_trajectory(3)

  Visualisation of how the eraser degradation works.
- pixels = all_pixels_in_trajectory(trajectory)
- s = torch.tensor([0.16])
- eraserheads = eraserhead_masks(pixels, radius=3, sigma=3, size=28)
- masks = torch.exp(torch.cumsum(torch.log(eraserheads), dim=0))
- index = torch.round(masks.shape[0] ∗ s)
- trajectory_img = np.zeros((28, 28))
- eraserhead = eraserheads[index]
- mask = masks[index]
- tf = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (1.0))])
- dataset = MNIST("./data", train=True, download=True, transform=tf)
- x = dataset[22][0].unsqueeze(0)
- degrador = Eraser(trajectory, eraserhead_radius=3, sigma=3, size=28)
- colour = torch.tensor([0.3])
- float noise = torch.randn_like(x) ∗ 0.05
- background
- cmap
- letter

6.3.1 Function Documentation

6.3.1.1 plot_loss()

```
make_plots.plot_loss (
            metric_logger )
```

Plot the training and test loss over epochs.

6.3.1.2 save_fig()

```
make_plots.save_fig (
              fig,
              save_path )
```

Save the figure to the specified path.

6.3.2 Variable Documentation

6.3.2.1 args

```
make_plots.args = parser.parse_known_args()[0]
```

6.3.2.2 axs

```
make_plots.axs = axs.flatten()
```

6.3.2.3 background

```
make_plots.background
```

6.3.2.4 cmap

```
make_plots.cmap
```

6.3.2.5 colour

```
make_plots.colour = torch.tensor([0.3])
```

6.3.2.6 dataset

```
make_plots.dataset = MNIST("./data", train=True, download=True, transform=tf)
```

6.3.2.7 degrador

```
make_plots.degrador = Eraser(trajectory, eraserhead_radius=3, sigma=3, size=28)
```

6.3.2.8 epochs

```
list make_plots.epochs = [1, 5, 10, 25]
```

6.3.2.9 eraserhead

```
make_plots.eraserhead = eraserheads[index]
```

6.3.2.10 eraserheads

```
make_plots.eraserheads = eraserhead_masks(pixels, radius=3, sigma=3, size=28)
```

6.3.2.11 fig

```
make_plots.fig = plot_loss(metric_logger)
```

6.3.2.12 figsize

```
make_plots.figsize
```

6.3.2.13 fontsize

```
make_plots.fontsize
```

6.3.2.14 fontweight

```
make_plots.fontweight
```

6.3.2.15 help

```
make_plots.help
```

6.3.2.16 img

```
make_plots.img = Image.open(path)
```

6.3.2.17 index

```
make_plots.index = torch.round(masks.shape[0] * s)
```

6.3.2.18 letter

```
make_plots.letter
```

6.3.2.19 letters

```
list make_plots.letters = ["(a)", "(b)", "(c)", "(d)"]
```

6.3.2.20 mask

```
make_plots.mask = masks[index]
```

6.3.2.21 masks

```
make_plots.masks = torch.exp(torch.cumsum(torch.log(eraserheads), dim=0))
```

6.3.2.22 metric_logger

```
make_plots.metric_logger = pickle.load(f)
```

6.3.2.23 models

```
list make_plots.models = ["ddpm_default", "ddpm_high", "magic_uneraser"]
```

Training visualisations.

6.3.2.24 noise

```
float make_plots.noise = torch.randn_like(x) * 0.05
```

6.3.2.25 parser

```
make_plots.parser = argparse.ArgumentParser()
```

6.3.2.26 path

```
make_plots.path = join(args.models, model_name, "samples", f"epoch_{epoch - 1:04d}.png")
```

6.3.2.27 pixels

```
make_plots.pixels = all_pixels_in_trajectory(trajectory)
```

6.3.2.28 s

```
make_plots.s = torch.tensor([0.16])
```

6.3.2.29 str

```
make_plots.str
```

6.3.2.30 tf

```
make_plots.tf = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (1.↩
0))])
```

6.3.2.31 trajectory

```
make_plots.trajectory = eraser_trajectory(3)
```

Visualisation of how the eraser degradation works.

6.3.2.32 trajectory_img

```
make_plots.trajectory_img = np.zeros((28, 28))
```

6.3.2.33 transAxes

```
make_plots.transAxes
```

6.3.2.34 transform

```
make_plots.transform
```

6.3.2.35 type

```
make_plots.type
```

6.3.2.36 x

```
make_plots.x = dataset[22][0].unsqueeze(0)
```

6.4 src Namespace Reference

This package contains the source code for the project.

Namespaces

- namespace cnn
- namespace config_parsing
- namespace degradation
- namespace models
- namespace noise_schedules

6.4.1 Detailed Description

This package contains the source code for the project.

6.5 src.cnn Namespace Reference

Classes

- class CNN
- class CNNBlock

6.6 src.config_parsing Namespace Reference

Functions

- parse_degradation_config (config)

  Parse the degradation strategy from the config file.
- parse_config (config_file)

  Parse the model configuration file.

6.6.1 Function Documentation

6.6.1.1 parse_config()

```
src.config_parsing.parse_config (
              config_file )
```

Parse the model configuration file.

Parse the configuration file to create the model and optimizer, as well as dictionary containing other important configuration parameters.

Parameters

config_file	Path to the configuration file.

Returns

Configuration dictionary, model and optimizer.

6.6.1.2 parse_degradation_config()

```
src.config_parsing.parse_degradation_config (
            config )
```

Parse the degradation strategy from the config file.

If the diffusion type is 'ddpm', the noise schedule is parsed. If the diffusion type is 'cold', the degradation operator is parsed.

Parameters

config	ConfigParser object containing the configuration file.

Returns

Degradation operator or noise schedule.

6.7 src.degradation Namespace Reference

Namespaces

- namespace base
- namespace eraser
- namespace eraser_utils

6.8 src.degradation.base Namespace Reference

Classes

- class DegredationOperator

  Base class for all degradation operators.

6.9 src.degradation.eraser Namespace Reference

Classes

- class Eraser

  Eraser Degredation Operator.

6.10 src.degradation.eraser_utils Namespace Reference

Functions

- get_zigzag_trajectory (left, top, bottom, right, size=28)

  Generates a zigzag trajectory for the eraser head.
- eraser_trajectory (radius)

  Generates a zigzag trajectory for the eraser head with a given radius.
- all_pixels_in_trajectory (list trajectory)

  Generates all pixels in the trajectory of the eraser head.
- disk_mask (pixel, radius, size)

  Generates a solid disk mask for the center of the eraser head.
- gaussian_mask (pixel, sigma, size)

  Generates a Gaussian mask centered at a specified pixel location.
- torch.Tensor eraserhead_masks (torch.Tensor pixels, int radius=3, float sigma=3, int size=28)

  Generates a mask for the head of the eraser.

6.10.1 Function Documentation

6.10.1.1 all_pixels_in_trajectory()

```
src.degradation.eraser_utils.all_pixels_in_trajectory (
            list trajectory )
```

Generates all pixels in the trajectory of the eraser head.

The trajectory is a list of pixel coordinates that the eraser head moves between in straight lines. This function generates all pixels along that trajectory by using the `line` function from skimage.

6.10.1.2 disk_mask()

```
src.degradation.eraser_utils.disk_mask (
             pixel,
             radius,
             size )
```

Generates a solid disk mask for the center of the eraser head.

The mask has zeros inside the disk and ones outside.

Parameters

pixel	A tensor (row, column) representing the center of the disk.
radius	The radius of the disk.
size	The size of the image.

Returns

: A tensor of shape (1, size, size) containing the mask.

6.10.1.3 eraser_trajectory()

```
src.degradation.eraser_utils.eraser_trajectory (
              radius )
```

Generates a zigzag trajectory for the eraser head with a given radius.

The trajectory consists of a series of points on the edge of the image that the eraser moves between. We need to ensure that the solid disk of the eraser head will move across every pixel in the image, to ensure that the image is fully erased. This involved calculation by hand to determine the correct points for the zigzag pattern, so we only allow `radius` to be 2 or 3.

6.10.1.4 eraserhead_masks()

```
torch.Tensor src.degradation.eraser_utils.eraserhead_masks (
             torch.Tensor pixels,
             int  radius = 3,
             float  sigma = 3,
             int  size = 28 )
```

Generates a mask for the head of the eraser.

(for each pixel in `pixels`)

The mask consists of a solid disk of radius `radius`, with a Gaussian falloff around the edge. `pixels` has shape (N, 2), where N is the number of masks to generate. For each pixel in `pixels`, a separate mask is generated, centered on that pixel.

Parameters

pixels	A tensor of shape (N, 2) containing the coordinates of the center pixels.
radius	The radius of the solid disk mask.
sigma	The standard deviation of the Gaussian falloff.
size	The size of the image.

Returns

A tensor of shape (N, 1, size, size) containing the masks.

6.10.1.5 gaussian_mask()

```
src.degradation.eraser_utils.gaussian_mask (
             pixel,
             sigma,
             size )
```

Generates a Gaussian mask centered at a specified pixel location.

The mask's values increase with distance from the center according to a Gaussian distribution.

Parameters

pixel	A tensor (row, column) representing the center of the mask.
sigma	The standard deviation of the Gaussian distribution.
size	The size of the image.

Returns

: A tensor of shape (1, size, size) containing the Gaussian mask.

6.10.1.6 get_zigzag_trajectory()

```
src.degradation.eraser_utils.get_zigzag_trajectory (
              left,
              top,
              bottom,
              right,
              size = 28 )
```

Generates a zigzag trajectory for the eraser head.

The trajectory consists of a series of points on the edges of the image that the eraser moves between in straight lines. The points are defined by intersections with `left`, `top`, `bottom`, and `right` edges. The eraser moves from the first point in `left` to the first point in `top`, then to the second point in `left`, and so on until the last point in `top`. Then it moves to the first point in `bottom`, then `right`, and so on until it has completed the full trajectory.

6.11 src.models Namespace Reference

Classes

- class ColdDiffusion

  Cold Diffusion model as described by Bansal et al.
- class DDPM

  Denoising Diffusion Probabilistic Model (DDPM) as described by Ho et al.

6.12 src.noise_schedules Namespace Reference

Functions

- Dict[str, torch.Tensor] linear_schedule (float beta1, float beta2, int T)

  Returns pre-computed schedules for DDPM sampling with a linear noise schedule.
- dict cosine_schedule (int T, float s=0.002)

  Returns pre-computed schedules for DDPM sampling with a cosine noise schedule.

6.12.1 Function Documentation

6.12.1.1 cosine_schedule()

```
dict src.noise_schedules.cosine_schedule (
            int T,
            float  s = 0.002 )
```

Returns pre-computed schedules for DDPM sampling with a cosine noise schedule.

Unfortunately the model did not train when using this schedule & I don't have time to figure out why :(

https://arxiv.org/pdf/2102.09672.pdf

We clip beta_t to be no larger than 0.999 to prevent singularities at the end of the diffusion process near t = T.

We use a small offset s to prevent beta_t from being too small near t = 0, since we found that having tiny amounts of noise at the beginning of the process made it hard for the network to predict the noise accurately enough. In particular, we selected s such that sqrt(beta_1) was slightly smaller than the pixel bin size 1/255, which gives 0.002.

6.12.1.2 linear_schedule()

```
Dict[str, torch.Tensor] src.noise_schedules.linear_schedule (
            float beta1,
            float beta2,
            int T )
```

Returns pre-computed schedules for DDPM sampling with a linear noise schedule.

6.13 train Namespace Reference

Variables

- t0 = time()
- config_file = sys.argv[1]
- int SEED = 14 + len(config_file)
- cfg
- model
- optim
- tf = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (1.0))])
- trainset = MNIST("./data", train=True, download=True, transform=tf)
- testset = MNIST("./data", train=False, download=True, transform=tf)
- train_loader = DataLoader(trainset, batch_size=cfg["batch_size"], shuffle=True, drop_last=True)
- test_loader = DataLoader(testset, batch_size=cfg["batch_size"], shuffle=False, drop_last=False)
- results_dir = os.path.join("./models", cfg["model_name"])
- exist_ok
- accelerator = Accelerator()
- device = accelerator.device
- dict metric_logger = {"train_loss": [ ], "test_loss": [ ]}
- list losses = [ ]
- int total_loss = 0
- loss = model(x)
- int train_loss = total_loss / len(train_loader)
- int test_loss = total_loss / len(test_loader)
- xh = model.sample(n_sample=16, img_shape=(1, 28, 28), device=device)
- grid = make_grid(xh, nrow=4)

6.13.1 Variable Documentation

6.13.1.1 accelerator

```
train.accelerator = Accelerator()
```

6.13.1.2 cfg

```
train.cfg
```

6.13.1.3 config_file

```
train.config_file = sys.argv[1]
```

6.13.1.4 device

```
train.device = accelerator.device
```

6.13.1.5 exist_ok

```
train.exist_ok
```

6.13.1.6 grid

```
train.grid = make_grid(xh, nrow=4)
```

6.13.1.7 loss

```
train.loss = model(x)
```

6.13.1.8 losses

```
list train.losses = []
```

6.13.1.9 metric_logger

```
dict train.metric_logger = {"train_loss":  [], "test_loss":  []}
```

6.13.1.10 model

```
train.model
```

6.13.1.11 optim

```
train.optim
```

6.13.1.12 results_dir

```
train.results_dir = os.path.join("./models", cfg["model_name"])
```

6.13.1.13 SEED

```
int train.SEED = 14 + len(config_file)
```

6.13.1.14 t0

```
train.t0 = time()
```

6.13.1.15 test_loader

```
train.test_loader = DataLoader(testset, batch_size=cfg["batch_size"], shuffle=False, drop_↩
last=False)
```

6.13.1.16 test_loss

```
int train.test_loss = total_loss / len(test_loader)
```

6.13.1.17 testset

```
train.testset = MNIST("./data", train=False, download=True, transform=tf)
```

6.13.1.18 tf

```
train.tf = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (1.0))])
```

6.13.1.19 total_loss

```
int train.total_loss = 0
```

6.13.1.20 train_loader

```
train.train_loader = DataLoader(trainset, batch_size=cfg["batch_size"], shuffle=True, drop_↩
last=True)
```

6.13.1.21 train_loss

```
int train.train_loss = total_loss / len(train_loader)
```

6.13.1.22 trainset

```
train.trainset = MNIST("./data", train=True, download=True, transform=tf)
```

6.13.1.23 xh

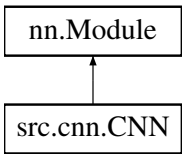
```
train.xh = model.sample(n_sample=16, img_shape=(1, 28, 28), device=device)
```

Chapter 7

Class Documentation

7.1 src.cnn.CNN Class Reference

Inheritance diagram for src.cnn.CNN:



Public Member Functions

- None __init__ (self, in_channels, expected_shape=(28, 28), n_hidden=(64, 128, 64), kernel_size=7, last_↩ kernel_size=3, time_embeddings=16, act=nn.GELU)
- torch.Tensor time_encoding (self, torch.Tensor s)

  Encode the time step into the latent space.
- torch.Tensor forward (self, torch.Tensor x, torch.Tensor s)

  Forward pass through the CNN.

Public Attributes

- blocks
- time_embed

7.1.1 Constructor & Destructor Documentation

7.1.1.1 __init__()

```
None src.cnn.CNN.__init__ (
              self,
              in_channels,
              expected_shape = (28, 28),
              n_hidden = (64, 128, 64),
              kernel_size = 7,
              last_kernel_size = 3,
              time_embeddings = 16,
              act = nn.GELU )
```

7.1.2 Member Function Documentation

7.1.2.1 forward()

```
torch.Tensor src.cnn.CNN.forward (
              self,
             torch.Tensor x,
             torch.Tensor s )
```

Forward pass through the CNN.

Parameters

x	Input tensor
s	Severity (timestep/T) - a scalar in [0, 1]

7.1.2.2 time_encoding()

```
torch.Tensor src.cnn.CNN.time_encoding (
              self,
             torch.Tensor s )
```

Encode the time step into the latent space.

Parameters

s	Severity (timestep/T) - a scalar in [0, 1]

7.1.3 Member Data Documentation

7.1.3.1 blocks

```
src.cnn.CNN.blocks
```

7.1.3.2 time_embed

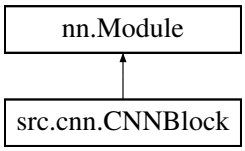
```
src.cnn.CNN.time_embed
```

The documentation for this class was generated from the following file:

- /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/cnn.py

7.2 src.cnn.CNNBlock Class Reference

Inheritance diagram for src.cnn.CNNBlock:



Public Member Functions

- __init__ (self, in_channels, out_channels, ∗expected_shape, act=nn.GELU, kernel_size=7)
- forward (self, x)

Public Attributes

- net

7.2.1 Constructor & Destructor Documentation

7.2.1.1 __init__()

```
src.cnn.CNNBlock.__init__ (
              self,
              in_channels,
              out_channels,
             * expected_shape,
              act = nn.GELU,
              kernel_size = 7 )
```

7.2.2 Member Function Documentation

7.2.2.1 forward()

```
src.cnn.CNNBlock.forward (
              self,
              x )
```

7.2.3 Member Data Documentation

7.2.3.1 net

```
src.cnn.CNNBlock.net
```

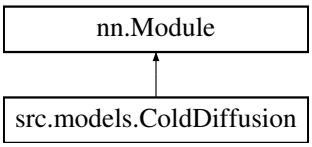
The documentation for this class was generated from the following file:

- /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/cnn.py

7.3 src.models.ColdDiffusion Class Reference

Cold Diffusion model as described by Bansal et al.

Inheritance diagram for src.models.ColdDiffusion:



Public Member Functions

- None __init__ (self, nn.Module reconstructor, DegredationOperator degrador, nn.Module criterion=nn.↩ MSELoss(), int T=1000)
- torch.Tensor forward (self, torch.Tensor x)
- torch.Tensor sample (self, int n_sample, img_shape=(1, 28, 28), device="cpu", bool visualise=False)

  Sample from the model using Bansal et al.

Public Attributes

- degrador
- reconstructor
- T
- criterion

7.3.1 Detailed Description

Cold Diffusion model as described by Bansal et al.

(2022)

The model accepts a reconstructor network, a degradation operator, a criterion and the number of steps T. The forward pass generates the time-step t in {1, ..., T}, converts this to severity, s = t/T, then generates the latent state z_t using the degradation operator. The reconstructor network then reconstructs the image from the latent state z_t and severity s, and the loss is calculated using the criterion (by default the mean squared error between the real and reconstructed image). The sample method generates samples from the model using Algorithm 2 from Bansal et al. (2022) - "Improved Sampling for Cold Diffusion". We actually implement a slight modification of this algorithm, where we do the step z_{t-1} = z_t - D(x_recon, t) + D(x_recon, t-1) for t in {T, ..., 2}, not t = 1. (We found better empirical results this way.)

7.3.2 Constructor & Destructor Documentation

7.3.2.1 __init__()

```
None src.models.ColdDiffusion.__init__ (
             self,
            nn.Module reconstructor,
            DegredationOperator degrador,
            nn.Module  criterion = nn.MSELoss(),
            int  T = 1000 )
```

7.3.3 Member Function Documentation

7.3.3.1 forward()

```
torch.Tensor src.models.ColdDiffusion.forward (
              self,
             torch.Tensor x )
```

7.3.3.2 sample()

```
torch.Tensor src.models.ColdDiffusion.sample (
              self,
             int n_sample,
              img_shape = (1, 28, 28),
              device = "cpu",
             bool  visualise = False )
```

Sample from the model using Bansal et al.

(2022) Algorithm 2

Parameters

n_sample	Number of samples to generate
img_shape	Shape of the image
device	Device to use
visualise	Whether to return all latent states for visualisation,

7.3.4 Member Data Documentation

7.3.4.1 criterion

```
src.models.ColdDiffusion.criterion
```

7.3.4.2 degrador

```
src.models.ColdDiffusion.degrador
```

7.3.4.3 reconstructor

```
src.models.ColdDiffusion.reconstructor
```

7.3.4.4 T

```
src.models.ColdDiffusion.T
```

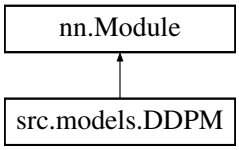
The documentation for this class was generated from the following file:

- /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/models.py

7.4 src.models.DDPM Class Reference

Denoising Diffusion Probabilistic Model (DDPM) as described by Ho et al.

Inheritance diagram for src.models.DDPM:



Public Member Functions

- None __init__ (self, gt, Dict[str, torch.Tensor] noise_schedule, nn.Module criterion=nn.MSELoss())
- torch.Tensor forward (self, torch.Tensor x)

  Algorithm 18.1 in Prince.
- torch.Tensor sample (self, int n_sample, img_shape=(1, 28, 28), device="cpu", visualise=False)

  Algorithm 18.2 in Prince.

Public Attributes

- gt
- T
- criterion

7.4.1 Detailed Description

Denoising Diffusion Probabilistic Model (DDPM) as described by Ho et al.

(2020)

7.4.2 Constructor & Destructor Documentation

7.4.2.1 __init__()

```
None src.models.DDPM.__init__ (
              self,
              gt,
             Dict[str, torch.Tensor] noise_schedule,
             nn.Module  criterion = nn.MSELoss() )
```

7.4.3 Member Function Documentation

7.4.3.1 forward()

```
torch.Tensor src.models.DDPM.forward (
              self,
             torch.Tensor x )
```

Algorithm 18.1 in Prince.

7.4.3.2 sample()

```
torch.Tensor src.models.DDPM.sample (
              self,
             int n_sample,
              img_shape = (1, 28, 28),
              device = "cpu",
              visualise = False )
```

Algorithm 18.2 in Prince.

Parameters

n_sample	Number of samples to generate
img_shape	Shape of the image
device	Device to use
visualise	Whether to return all latent states for visualisation

7.4.4 Member Data Documentation

7.4.4.1 criterion

```
src.models.DDPM.criterion
```

7.4.4.2 gt

```
src.models.DDPM.gt
```

7.4.4.3 T

```
src.models.DDPM.T
```

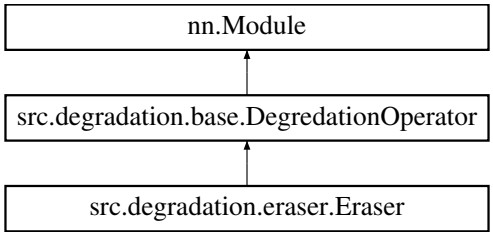
The documentation for this class was generated from the following file:

- /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/models.py

7.5 src.degradation.base.DegredationOperator Class Reference

Base class for all degradation operators.

Inheritance diagram for src.degradation.base.DegredationOperator:



Public Member Functions

- __init__ (self)
- sampling (self, bool mode=True)

  Set the sampling mode of the degradation operator.
- train (self, bool mode=True)

  Set the sampling mode to False if we are in training mode.

Public Attributes

- sampling_mode

7.5.1 Detailed Description

Base class for all degradation operators.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 __init__()

```
src.degradation.base.DegredationOperator.__init__ (
              self )
```

Reimplemented in src.degradation.eraser.Eraser.

7.5.3 Member Function Documentation

7.5.3.1 sampling()

```
src.degradation.base.DegredationOperator.sampling (
              self,
             bool  mode = True )
```

Set the sampling mode of the degradation operator.

7.5.3.2 train()

```
src.degradation.base.DegredationOperator.train (
              self,
             bool  mode = True )
```

Set the sampling mode to False if we are in training mode.

7.5.4 Member Data Documentation

7.5.4.1 sampling_mode

`src.degradation.base.DegredationOperator.sampling_mode`

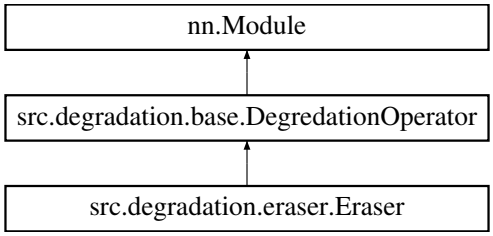
The documentation for this class was generated from the following file:

- /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/base.py

7.6 src.degradation.eraser.Eraser Class Reference

Eraser Degredation Operator.

Inheritance diagram for src.degradation.eraser.Eraser:



Public Member Functions

- __init__ (self, trajectory, int eraserhead_radius=3, float sigma=3, float noise_std=0.02, int size=28)
- __call__ (self, torch.Tensor x, torch.Tensor s)

  s - severity (t/T) in [0, 1]
- latent_sampler (self, n_sample, img_shape, device)

Public Member Functions inherited from src.degradation.base.DegredationOperator

- sampling (self, bool mode=True)

  Set the sampling mode of the degradation operator.
- train (self, bool mode=True)

  Set the sampling mode to False if we are in training mode.

Public Attributes

- noise_std
- background

Public Attributes inherited from src.degradation.base.DegredationOperator

- sampling_mode

7.6.1 Detailed Description

Eraser Degredation Operator.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 __init__()

```
src.degradation.eraser.Eraser.__init__ (
              self,
              trajectory,
             int  eraserhead_radius = 3,
             float  sigma = 3,
             float  noise_std = 0.02,
             int  size = 28 )
```

Reimplemented from src.degradation.base.DegredationOperator.

7.6.3 Member Function Documentation

7.6.3.1 __call__()

```
src.degradation.eraser.Eraser.__call__ (
              self,
             torch.Tensor x,
             torch.Tensor s )
```

s - severity (t/T) in [0, 1]

The index of the mask to apply is given by s $*$ M, where M is the number of pixels/masks in the trajectory.

7.6.3.2 latent_sampler()

```
src.degradation.eraser.Eraser.latent_sampler (
              self,
              n_sample,
              img_shape,
              device )
```

7.6.4 Member Data Documentation

7.6.4.1 background

```
src.degradation.eraser.Eraser.background
```

7.6.4.2 noise_std

```
src.degradation.eraser.Eraser.noise_std
```

The documentation for this class was generated from the following file:

- /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/eraser.py

Chapter 8

File Documentation

8.1 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/eval.py File Reference

Evaluation script for trained diffusion model.

Namespaces

- namespace eval

Variables

- int eval.SEED = 1
- eval.t0 = time()
- eval.parser = argparse.ArgumentParser()
- eval.type
- eval.str
- eval.help
- eval.default
- eval.int
- eval.args = parser.parse_known_args()[0]
- eval.device = torch.device("cpu")
- eval.cfg
- eval.model
- eval._
- eval.state_dict = torch.load(join(args.model_dir, "state_dict.pth"), map_location=device)
- eval.n_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
- eval.x_gen = model.sample(args.n_samples, (1, 28, 28), device)
- eval.grid = make_grid(x_gen, nrow=int(np.sqrt(args.n_samples)))
- eval.save_path = join(args.output_dir, f"{cfg['model_name']}_samples.png")
- eval.tf

  FID Score.
- eval.testset = MNIST("./data", train=False, download=True, transform=tf)
- eval.dataloader = DataLoader(testset, batch_size=500, shuffle=True)
- eval.transform_for_inceptionv3
- eval.fid = FrechetInceptionDistance(feature=2048)
- eval.real_images = transform_for_inceptionv3(real_images)
- eval.real
- eval.fid_score = fid.compute()

8.1.1 Detailed Description

Evaluation script for trained diffusion model.

This script generates samples from the trained model and computes the FID score between the generated samples and the MNIST test set. However many samples are generated, the FID score is computed on the same number of samples from the MNIST test set.

Example usage: `python eval.py --model_dir ./models/ddpm_default --output_dir ./plots`

8.2 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/make_↩ plots.py File Reference

Script to make plots for the report.

Namespaces

- namespace make_plots

Functions

- make_plots.plot_loss (metric_logger)

  Plot the training and test loss over epochs.
- make_plots.save_fig (fig, save_path)

  Save the figure to the specified path.

Variables

- make_plots.parser = argparse.ArgumentParser()
- make_plots.type
- make_plots.str
- make_plots.help
- make_plots.args = parser.parse_known_args()[0]
- list make_plots.models = ["ddpm_default", "ddpm_high", "magic_uneraser"]

  Training visualisations.
- make_plots.metric_logger = pickle.load(f)
- make_plots.fig = plot_loss(metric_logger)
- list make_plots.epochs = [1, 5, 10, 25]
- list make_plots.letters = ["(a)", "(b)", "(c)", "(d)"]
- make_plots.axs = axs.flatten()
- make_plots.figsize
- make_plots.path = join(args.models, model_name, "samples", f"epoch_{epoch - 1:04d}.png")
- make_plots.img = Image.open(path)
- make_plots.fontsize
- make_plots.transform
- make_plots.transAxes
- make_plots.fontweight
- make_plots.trajectory = eraser_trajectory(3)

Visualisation of how the eraser degradation works.

- make_plots.pixels = all_pixels_in_trajectory(trajectory)
- make_plots.s = torch.tensor([0.16])
- make_plots.eraserheads = eraserhead_masks(pixels, radius=3, sigma=3, size=28)
- make_plots.masks = torch.exp(torch.cumsum(torch.log(eraserheads), dim=0))
- make_plots.index = torch.round(masks.shape[0] ∗ s)
- make_plots.trajectory_img = np.zeros((28, 28))
- make_plots.eraserhead = eraserheads[index]
- make_plots.mask = masks[index]
- make_plots.tf = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (1.0))])
- make_plots.dataset = MNIST("./data", train=True, download=True, transform=tf)
- make_plots.x = dataset[22][0].unsqueeze(0)
- make_plots.degrador = Eraser(trajectory, eraserhead_radius=3, sigma=3, size=28)
- make_plots.colour = torch.tensor([0.3])
- float make_plots.noise = torch.randn_like(x) ∗ 0.05
- make_plots.background
- make_plots.cmap
- make_plots.letter

8.2.1 Detailed Description

Script to make plots for the report.

We make the following plots:

- Training and test loss over epochs for each model
- Samples from the models at epochs 1, 5, 10, 30
- Visualisation of the eraser degradation operator

Example usage: `python make_plots.py --models ./models --output_dir ./plots`

8.3 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/README.md File Reference

8.4 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/__init__.py File Reference

Namespaces

- namespace src

  This package contains the source code for the project.

8.5 /Users/willknott/Desktop/DIS/↩ SEM2/coursework/m2/wdk24/src/degradation/__init__.py File Reference

Namespaces

- namespace src

  This package contains the source code for the project.
- namespace src.degradation
- namespace degradation

  This package contains the degradation operators for the project.

8.6 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/cnn.py File Reference

We create a simple 2D convolutional neural network to use as the reconstructor network within the diffusion models.

Classes

- class src.cnn.CNNBlock
- class src.cnn.CNN

Namespaces

- namespace src

  This package contains the source code for the project.
- namespace src.cnn

8.6.1 Detailed Description

We create a simple 2D convolutional neural network to use as the reconstructor network within the diffusion models.

The CNN is a stack of convolutional blocks, each of which consists of a convolutional layer, followed by a layer normalization and a GELU activation.

First, we create a single CNN block which we will stack to create the full network. We use `LayerNorm` for stable training and no batch dependence.

We then create the full CNN model, which is a stack of these blocks according to the `n_hidden` tuple, which specifies the number of channels at each hidden layer.

8.7 /Users/willknott/Desktop/DIS/↩ SEM2/coursework/m2/wdk24/src/config_parsing.py File Reference

Functions for parsing the model configuration files.

Namespaces

- namespace src

  This package contains the source code for the project.
- namespace src.config_parsing

Functions

- src.config_parsing.parse_degradation_config (config)

  Parse the degradation strategy from the config file.
- src.config_parsing.parse_config (config_file)

  Parse the model configuration file.

8.7.1 Detailed Description

Functions for parsing the model configuration files.

8.8 /Users/willknott/Desktop/DIS/↩ SEM2/coursework/m2/wdk24/src/degradation/base.py File Reference

Contains base class for all degradation operators.

Classes

- class src.degradation.base.DegredationOperator

  Base class for all degradation operators.

Namespaces

- namespace src

  This package contains the source code for the project.
- namespace src.degradation
- namespace src.degradation.base

8.8.1 Detailed Description

Contains base class for all degradation operators.

Degradation operators are required in the training and sampling process of a cold diffusion model, and it is important to know whether we are sampling or training. The `sampling` method sets the sampling mode of the degradation operator. The `train` method is a wrapper around the `train` method of the parent class (nn.Module), and sets the sampling mode to False.

8.9 /Users/willknott/Desktop/DIS/↩ SEM2/coursework/m2/wdk24/src/degradation/eraser.py File Reference

Contains the Eraser Degredation Operator.

Classes

- class src.degradation.eraser.Eraser

  Eraser Degredation Operator.

Namespaces

- namespace src

  This package contains the source code for the project.
- namespace src.degradation
- namespace src.degradation.eraser

8.9.1 Detailed Description

Contains the Eraser Degredation Operator.

The idea is that the operator simulates the effect of an eraser 'rubbing out the image', where the eraser head is moved across the image in a zigzag pattern. This operator applies a mask to the input image, with severity controlled by the variable `s`. The mask is generated by a series of individual Eraser head masks, which are solid disks of radius `eraserhead_radius` with a Gaussian falloff at the edges with standard deviation `sigma`. The trajectory of the eraser head is defined by the `trajectory` parameter, which is a list of pixel coordinates at the edges of the image, that the eraser head moves between in straight lines. First, we calculate all pixels along this trajectory using the `all_pixels_in_trajectory` function. The mask is generated by applying the individual eraser head masks at each pixel in the trajectory, and multiplying them together. The severity parameter `s` (= t/T) is a float in the range [0, 1], which controls how far along the trajectory the eraser head has moved. For `s = 0`, the mask is the identity, and for `s = 1`, the mask is the final mask, leaving no trace of the original image. To ensure that the latent space is diverse, we make it so that the background image (the image left behind by the eraser) is a solid random colour with noise added. This noise is gaussian with standard deviation `noise_std`. The `__call__` method applies the degradation operator to the input image `x` with severity `s`.

Degradation operators are required in the training and sampling process of a cold diffusion model. If we are sampling from the model, we need to fix the background image for the whole generation process. The sampling mode is set using the `sampling` method from the DegredationOperator base class. The `latent_sampler` method generates a batch of background images for the given batch size and image shape. If we are in sampling mode, then it fixes these images as the background for the whole process.

8.10 /Users/willknott/Desktop/DIS/↩ SEM2/coursework/m2/wdk24/src/degradation/eraser_utils.py File Reference

Contains all utility functions for the Eraser Degredation Operator.

Namespaces

- namespace src

  This package contains the source code for the project.
- namespace src.degradation
- namespace src.degradation.eraser_utils

Functions

- src.degradation.eraser_utils.get_zigzag_trajectory (left, top, bottom, right, size=28)

  Generates a zigzag trajectory for the eraser head.
- src.degradation.eraser_utils.eraser_trajectory (radius)

  Generates a zigzag trajectory for the eraser head with a given radius.
- src.degradation.eraser_utils.all_pixels_in_trajectory (list trajectory)

  Generates all pixels in the trajectory of the eraser head.
- src.degradation.eraser_utils.disk_mask (pixel, radius, size)

  Generates a solid disk mask for the center of the eraser head.
- src.degradation.eraser_utils.gaussian_mask (pixel, sigma, size)

  Generates a Gaussian mask centered at a specified pixel location.
- torch.Tensor src.degradation.eraser_utils.eraserhead_masks (torch.Tensor pixels, int radius=3, float sigma=3, int size=28)

  Generates a mask for the head of the eraser.

8.10.1 Detailed Description

Contains all utility functions for the Eraser Degredation Operator.

8.11 /Users/willknott/Desktop/DIS/↩ SEM2/coursework/m2/wdk24/src/models.py File Reference

Diffusion models for image generation.

Classes

- class src.models.DDPM

  Denoising Diffusion Probabilistic Model (DDPM) as described by Ho et al.
- class src.models.ColdDiffusion

  Cold Diffusion model as described by Bansal et al.

Namespaces

- namespace src

  This package contains the source code for the project.
- namespace src.models

8.11.1 Detailed Description

Diffusion models for image generation.

This file contains the implementation of the DDPM and Cold Diffusion models.

8.12 /Users/willknott/Desktop/DIS/↩ SEM2/coursework/m2/wdk24/src/noise_schedules.py File Reference

Functions for generating noise schedules for DDPM sampling.

Namespaces

- namespace src
  This package contains the source code for the project.
- namespace src.noise_schedules

Functions

- Dict[str, torch.Tensor] src.noise_schedules.linear_schedule (float beta1, float beta2, int T)
  Returns pre-computed schedules for DDPM sampling with a linear noise schedule.
- dict src.noise_schedules.cosine_schedule (int T, float s=0.002)
  Returns pre-computed schedules for DDPM sampling with a cosine noise schedule.

8.12.1 Detailed Description

Functions for generating noise schedules for DDPM sampling.

8.13 /Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/train.py File Reference

Script to train diffusion model on MNIST dataset.

Namespaces

- namespace train

Variables

- train.t0 = time()
- train.config_file = sys.argv[1]
- int train.SEED = 14 + len(config_file)
- train.cfg
- train.model
- train.optim
- train.tf = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (1.0))])
- train.trainset = MNIST("./data", train=True, download=True, transform=tf)
- train.testset = MNIST("./data", train=False, download=True, transform=tf)
- train.train_loader = DataLoader(trainset, batch_size=cfg["batch_size"], shuffle=True, drop_last=True)
- train.test_loader = DataLoader(testset, batch_size=cfg["batch_size"], shuffle=False, drop_last=False)
- train.results_dir = os.path.join("./models", cfg["model_name"])
- train.exist_ok
- train.accelerator = Accelerator()
- train.device = accelerator.device
- dict train.metric_logger = {"train_loss": [ ], "test_loss": [ ]}
- list train.losses = [ ]
- int train.total_loss = 0
- train.loss = model(x)
- int train.train_loss = total_loss / len(train_loader)
- int train.test_loss = total_loss / len(test_loader)
- train.xh = model.sample(n_sample=16, img_shape=(1, 28, 28), device=device)
- train.grid = make_grid(xh, nrow=4)

8.13.1 Detailed Description

Script to train diffusion model on MNIST dataset.

The model is specified in the config file. It can be cold diffusion or DDPM.

Example usage: `python train.py ./configs/ddpm_default.ini`

Index

/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/README.md, 43
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/eval.py, 41
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/make_plots.py, 42
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/__init__.py, 43
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/cnn.py, 44
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/config_parsing.py, 44
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/__init__.py, 44
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/base.py, 45
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/eraser.py, 46
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/degradation/eraser_utils.py, 46
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/models.py, 47
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/src/noise_schedules.py, 48
/Users/willknott/Desktop/DIS/SEM2/coursework/m2/wdk24/train.py, 48

_
    eval, 14
__call__
    src.degradation.eraser.Eraser, 40
__init__
    src.cnn.CNN, 31
    src.cnn.CNNBlock, 33
    src.degradation.base.DegredationOperator, 38
    src.degradation.eraser.Eraser, 40
    src.models.ColdDiffusion, 34
    src.models.DDPM, 36

accelerator
    train, 28
all_pixels_in_trajectory
    src.degradation.eraser_utils, 24
args
    eval, 14
    make_plots, 18
axs
    make_plots, 18

background
    make_plots, 18
blocks
    src.cnn.CNN, 32

cfg
    eval, 14
    train, 28
cmap
    make_plots, 18
colour
    make_plots, 18
config_file
    train, 28
cosine_schedule
    src.noise_schedules, 27
criterion
    src.models.ColdDiffusion, 35
    src.models.DDPM, 37
dataloader
    eval, 14
dataset
    make_plots, 18
default
    eval, 14
degradation, 13
degrador
    make_plots, 18
    src.models.ColdDiffusion, 35
device
    eval, 14
    train, 28
disk_mask
    src.degradation.eraser_utils, 24

epochs
    make_plots, 18
eraser_trajectory
    src.degradation.eraser_utils, 24
eraserhead
    make_plots, 18
eraserhead_masks
    src.degradation.eraser_utils, 25
eraserheads
    make_plots, 19
eval, 13
    _, 14
    args, 14
    cfg, 14
    dataloader, 14

default, 14
device, 14
fid, 14
fid_score, 14
grid, 14
help, 14
int, 14
model, 15
n_params, 15
parser, 15
real, 15
real_images, 15
save_path, 15
SEED, 15
state_dict, 15
str, 15
t0, 15
testset, 16
tf, 16
transform_for_inceptionv3, 16
type, 16
x_gen, 16
exist_ok
train, 28

fid
eval, 14
fid_score
eval, 14
fig
make_plots, 19
figsize
make_plots, 19
fontsize
make_plots, 19
fontweight
make_plots, 19
forward
src.cnn.CNN, 32
src.cnn.CNNBlock, 33
src.models.ColdDiffusion, 35
src.models.DDPM, 36

gaussian_mask
src.degradation.eraser_utils, 25
get_zigzag_trajectory
src.degradation.eraser_utils, 26
grid
eval, 14
train, 28
gt
src.models.DDPM, 37

help
eval, 14
make_plots, 19

img
make_plots, 19
index
make_plots, 19
int
eval, 14

latent_sampler
src.degradation.eraser.Eraser, 40
letter
make_plots, 19
letters
make_plots, 19
linear_schedule
src.noise_schedules, 27
loss
train, 28
losses
train, 28

Magic Un-Eraser, 1
make_plots, 16
args, 18
axs, 18
background, 18
cmap, 18
colour, 18
dataset, 18
degrador, 18
epochs, 18
eraserhead, 18
eraserheads, 19
fig, 19
figsize, 19
fontsize, 19
fontweight, 19
help, 19
img, 19
index, 19
letter, 19
letters, 19
mask, 20
masks, 20
metric_logger, 20
models, 20
noise, 20
parser, 20
path, 20
pixels, 20
plot_loss, 17
s, 20
save_fig, 17
str, 20
tf, 21
trajectory, 21
trajectory_img, 21
transAxes, 21
transform, 21
type, 21
x, 21
mask

make_plots, 20
masks
make_plots, 20
metric_logger
make_plots, 20
train, 28
model
eval, 15
train, 28
models
make_plots, 20

n_params
eval, 15
net
src.cnn.CNNBlock, 33
noise
make_plots, 20
noise_std
src.degradation.eraser.Eraser, 40

optim
train, 28

parse_config
src.config_parsing, 22
parse_degradation_config
src.config_parsing, 23
parser
eval, 15
make_plots, 20
path
make_plots, 20
pixels
make_plots, 20
plot_loss
make_plots, 17

real
eval, 15
real_images
eval, 15
reconstructor
src.models.ColdDiffusion, 35
results_dir
train, 29

s
make_plots, 20
sample
src.models.ColdDiffusion, 35
src.models.DDPM, 36
sampling
src.degradation.base.DegredationOperator, 38
sampling_mode
src.degradation.base.DegredationOperator, 39
save_fig
make_plots, 17
save_path
eval, 15
SEED
eval, 15
train, 29
src, 21
src.cnn, 22
src.cnn.CNN, 31
__init__, 31
blocks, 32
forward, 32
time_embed, 32
time_encoding, 32
src.cnn.CNNBlock, 33
__init__, 33
forward, 33
net, 33
src.config_parsing, 22
parse_config, 22
parse_degradation_config, 23
src.degradation, 23
src.degradation.base, 23
src.degradation.base.DegredationOperator, 37
__init__, 38
sampling, 38
sampling_mode, 39
train, 38
src.degradation.eraser, 23
src.degradation.eraser.Eraser, 39
__call__, 40
__init__, 40
background, 40
latent_sampler, 40
noise_std, 40
src.degradation.eraser_utils, 24
all_pixels_in_trajectory, 24
disk_mask, 24
eraser_trajectory, 24
eraserhead_masks, 25
gaussian_mask, 25
get_zigzag_trajectory, 26
src.models, 26
src.models.ColdDiffusion, 34
__init__, 34
criterion, 35
degrador, 35
forward, 35
reconstructor, 35
sample, 35
T, 35
src.models.DDPM, 36
__init__, 36
criterion, 37
forward, 36
gt, 37
sample, 36
T, 37
src.noise_schedules, 26
cosine_schedule, 27

linear_schedule, 27
state_dict
eval, 15
str
eval, 15
make_plots, 20

T
src.models.ColdDiffusion, 35
src.models.DDPM, 37
t0
eval, 15
train, 29
test_loader
train, 29
test_loss
train, 29
testset
eval, 16
train, 29
tf
eval, 16
make_plots, 21
train, 29
time_embed
src.cnn.CNN, 32
time_encoding
src.cnn.CNN, 32
total_loss
train, 29
train, 27
accelerator, 28
cfg, 28
config_file, 28
device, 28
exist_ok, 28
grid, 28
loss, 28
losses, 28
metric_logger, 28
model, 28
optim, 28
results_dir, 29
SEED, 29
src.degradation.base.DegredationOperator, 38
t0, 29
test_loader, 29
test_loss, 29
testset, 29
tf, 29
total_loss, 29
train_loader, 29
train_loss, 29
trainset, 30
xh, 30
train_loader
train, 29
train_loss
train, 29
trainset
train, 30
trajectory
make_plots, 21
trajectory_img
make_plots, 21
transAxes
make_plots, 21
transform
make_plots, 21
transform_for_inceptionv3
eval, 16
type
eval, 16
make_plots, 21

x
make_plots, 21
x_gen
eval, 16
xh
train, 30