MNIST Diffusion Project

Generated by Doxygen 1.9.1

Chapter 1

MNIST Diffusion Project

Software used to investigate the use of generative diffusion modelling to recreate realistic monochrome images of handwritten digits, using the classical MNIST dataset as training data. The software utilises PyTorch to implement the neural network models and the training procedure.

1.1 Modules

1.1.1 models

This module contains code used to instantiate standard image-to-image convolutional neural networks, as well as a DDPM class, with methods that enable training the image-to-image model according to the diffusion paradigm, and similarly performing diffusion sampling.

1.1.2 train

This module contains the ddpm_train function, which implements the diffusion training process, and saves the model parameters and samples periodically, according to specified training parameters.

Author

Created by J. Hughes on 06/12/2023.

Chapter 2

Namespace Index

2.1 Packages

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 diffusiontools Namespace Reference

Namespaces

- models
- train

6.2 diffusiontools.models Namespace Reference

Classes

- class CNNBlock
- class CNN
- class DDPM

Functions

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

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

6.2.1 Function Documentation

6.2.1.1 ddpm_schedules()

```
Dict[str, torch.Tensor] diffusiontools.models.ddpm_schedules (
              float beta1,
              float beta2,
              int   T )
```

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

6.3 diffusiontools.train Namespace Reference

Functions

- def train_ddpm (nn.Module ddpm, torch.optim.Optimizer optim, DataLoader dataloader, Accelerator accelerator, int n_epoch, int save_interval, str sample_path, str checkpoint_path, str config_id)

6.3.1 Function Documentation

6.3.1.1 train_ddpm()

```
def diffusiontools.train.train_ddpm (
             nn.Module ddpm,
             torch.optim.Optimizer optim,
             DataLoader dataloader,
             Accelerator accelerator,
             int n_epoch,
             int save_interval,
             str sample_path,
             str checkpoint_path,
             str config_id )
```

6.4 main Namespace Reference

Variables

- config = cfg.ConfigParser()
- input_file = sys.argv[1]
- beta1 = config.getfloat("model", "beta1", fallback=1e-4)
- beta2 = config.getfloat("model", "beta2", fallback=0.02)
- n_T = config.getint("model", "n_T", fallback=1000)
- n_hidden = config.get("model", "n_hidden", fallback="16 32 32 16")
- n_epoch = config.getint("training", "n_epoch", fallback=100)
- batch_size = config.getint("training", "batch_size", fallback=128)
- lr_initial = config.getfloat("training", "lr_initial", fallback=2e-4)
- checkpoint_path
- sample_path
- save_interval = config.getint("output", "save_interval", fallback=10)
- config_id = config.getint("output", "config_id", fallback=1234)
- tf
- dataset = MNIST("./data", train=True, download=True, transform=tf)
- dataloader
- gt
- ddpm = DDPM(gt=gt, betas=(beta1, beta2), n_T=n_T)
- optim = torch.optim.Adam(ddpm.parameters(), lr=lr_initial)
- accelerator = Accelerator()

6.4.1 Variable Documentation

6.4.1.1 accelerator

```
main.accelerator = Accelerator()
```

6.4.1.2 batch_size

```
main.batch_size = config.getint("training", "batch_size", fallback=128)
```

6.4.1.3 beta1

```
main.beta1 = config.getfloat("model", "beta1", fallback=1e-4)
```

6.4.1.4 beta2

```
main.beta2 = config.getfloat("model", "beta2", fallback=0.02)
```

6.4.1.5 checkpoint_path

```
main.checkpoint_path
```

Initial value:

```
=  config.get(
    "output", "checkpoint_path", fallback="./data/DDPM/checkpoint/"
)
```

6.4.1.6 config

```
main.config = cfg.ConfigParser()
```

6.4.1.7 config_id

```
main.config_id = config.getint("output", "config_id", fallback=1234)
```

6.4.1.8 dataloader

```
main.dataloader
```

Initial value:

```
=  DataLoader(
     dataset, batch_size=batch_size, shuffle=True, num_workers=4, drop_last=True
 )
```

6.4.1.9 dataset

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

6.4.1.10 ddpm

```
main.ddpm = DDPM(gt=gt, betas=(beta1, beta2), n_T=n_T)
```

6.4.1.11 gt

```
main.gt
```

Initial value:

```
=  CNN(
     in_channels=1, expected_shape=(28, 28), n_hidden=n_hidden, act=nn.GELU
 )
```

6.4.1.12 input_file

```
main.input_file = sys.argv[1]
```

6.4.1.13 lr_initial

```
main.lr_initial = config.getfloat("training", "lr_initial", fallback=2e-4)
```

6.4.1.14 n_epoch

```
main.n_epoch = config.getint("training", "n_epoch", fallback=100)
```

6.4.1.15 n_hidden

```
main.n_hidden = config.get("model", "n_hidden", fallback="16 32 32 16")
```

6.4.1.16 n_T

```
main.n_T = config.getint("model", "n_T", fallback=1000)
```

6.4.1.17 optim

```
main.optim = torch.optim.Adam(ddpm.parameters(), lr=lr_initial)
```

6.4.1.18 sample_path

```
main.sample_path
```

Initial value:

```
=  config.get(
     "output", "sample_path", fallback="./data/DDPM/sample/"
)
```

6.4.1.19 save_interval

```
main.save_interval = config.getint("output", "save_interval", fallback=10)
```

6.4.1.20 tf

```
main.tf
```

Initial value:

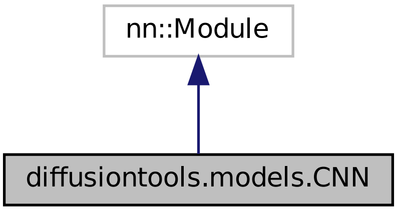
```
=  transforms.Compose(
     [transforms.ToTensor(), transforms.Normalize((0.5,), (1.0))]
)
```

Chapter 7

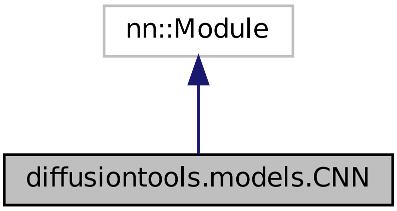
Class Documentation

7.1 diffusiontools.models.CNN Class Reference

Inheritance diagram for diffusiontools.models.CNN:



Collaboration diagram for diffusiontools.models.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 t)
- torch.Tensor forward (self, torch.Tensor x, torch.Tensor t)

Public Attributes

- blocks
- time_embed

7.1.1 Constructor & Destructor Documentation

7.1.1.1 __init__()

```
None diffusiontools.models.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 diffusiontools.models.CNN.forward (
              self,
             torch.Tensor x,
             torch.Tensor t )
```

7.1.2.2 time_encoding()

```
torch.Tensor diffusiontools.models.CNN.time_encoding (
              self,
             torch.Tensor t )
```

7.1.3 Member Data Documentation

7.1.3.1 blocks

```
diffusiontools.models.CNN.blocks
```

7.1.3.2 time_embed

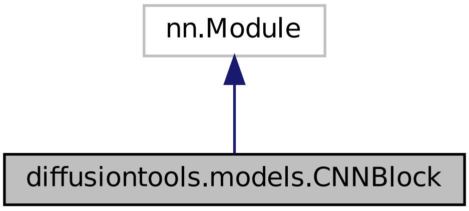
```
diffusiontools.models.CNN.time_embed
```

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

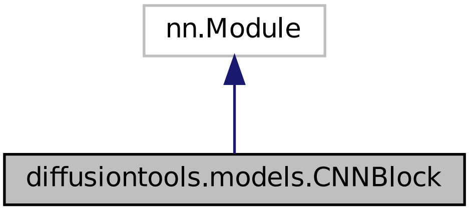
- /home/jhughes2712/projects/m2_assessment/jh2284/src/diffusiontools/models.py

7.2 diffusiontools.models.CNNBlock Class Reference

Inheritance diagram for diffusiontools.models.CNNBlock:



Collaboration diagram for diffusiontools.models.CNNBlock:



Public Member Functions

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

Public Attributes

- net

7.2.1 Constructor & Destructor Documentation

7.2.1.1 __init__()

```
def diffusiontools.models.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()

```
def diffusiontools.models.CNNBlock.forward (
              self,
              x )
```

7.2.3 Member Data Documentation

7.2.3.1 net

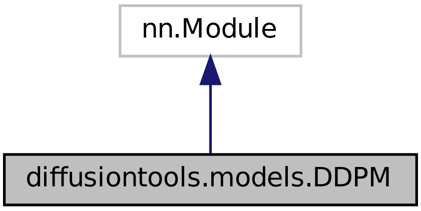
```
diffusiontools.models.CNNBlock.net
```

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

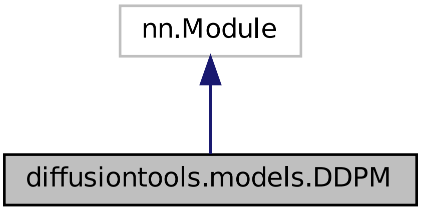
- /home/jhughes2712/projects/m2_assessment/jh2284/src/diffusiontools/models.py

7.3 diffusiontools.models.DDPM Class Reference

Inheritance diagram for diffusiontools.models.DDPM:



Collaboration diagram for diffusiontools.models.DDPM:



Public Member Functions

- None __init__ (self, gt, Tuple[float, float] betas, int n_T, nn.Module criterion=nn.MSELoss())
- torch.Tensor forward (self, torch.Tensor x)

  Algorithm 18.1 in Prince.
- torch.Tensor sample (self, int n_sample, size, device)

  Algorithm 18.2 in Prince.

Public Attributes

- gt
- n_T
- criterion

7.3.1 Constructor & Destructor Documentation

7.3.1.1 __init__()

```
None diffusiontools.models.DDPM.__init__ (
              self,
              gt,
             Tuple[float, float] betas,
             int n_T,
             nn.Module  criterion = nn.MSELoss() )
```

7.3.2 Member Function Documentation

7.3.2.1 forward()

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

Algorithm 18.1 in Prince.

7.3.2.2 sample()

```
torch.Tensor diffusiontools.models.DDPM.sample (
              self,
             int n_sample,
              size,
              device )
```

Algorithm 18.2 in Prince.

7.3.3 Member Data Documentation

7.3.3.1 criterion

```
diffusiontools.models.DDPM.criterion
```

7.3.3.2 gt

```
diffusiontools.models.DDPM.gt
```

7.3.3.3 n_T

```
diffusiontools.models.DDPM.n_T
```

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

- /home/jhughes2712/projects/m2_assessment/jh2284/src/diffusiontools/models.py

Chapter 8

File Documentation

8.1 /home/jhughes2712/projects/m2_↩ assessment/jh2284/src/diffusiontools/__init__.py File Reference

Namespaces

- diffusiontools

8.2 /home/jhughes2712/projects/m2_↩ assessment/jh2284/src/diffusiontools/models.py File Reference

Classes

- class diffusiontools.models.CNNBlock
- class diffusiontools.models.CNN
- class diffusiontools.models.DDPM

Namespaces

- diffusiontools.models

Functions

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

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

8.3 /home/jhughes2712/projects/m2_↩ assessment/jh2284/src/diffusiontools/train.py File Reference

Namespaces

- diffusiontools.train

Functions

- def diffusiontools.train.train_ddpm (nn.Module ddpm, torch.optim.Optimizer optim, DataLoader dataloader, Accelerator accelerator, int n_epoch, int save_interval, str sample_path, str checkpoint_path, str config_id)

8.4 /home/jhughes2712/projects/m2_assessment/jh2284/src/main.py File Reference

Used to create and train DDPM .

Namespaces

- main

Variables

- main.config = cfg.ConfigParser()
- main.input_file = sys.argv[1]
- main.beta1 = config.getfloat("model", "beta1", fallback=1e-4)
- main.beta2 = config.getfloat("model", "beta2", fallback=0.02)
- main.n_T = config.getint("model", "n_T", fallback=1000)
- main.n_hidden = config.get("model", "n_hidden", fallback="16 32 32 16")
- main.n_epoch = config.getint("training", "n_epoch", fallback=100)
- main.batch_size = config.getint("training", "batch_size", fallback=128)
- main.lr_initial = config.getfloat("training", "lr_initial", fallback=2e-4)
- main.checkpoint_path
- main.sample_path
- main.save_interval = config.getint("output", "save_interval", fallback=10)
- main.config_id = config.getint("output", "config_id", fallback=1234)
- main.tf
- main.dataset = MNIST("./data", train=True, download=True, transform=tf)
- main.dataloader
- main.gt
- main.ddpm = DDPM(gt=gt, betas=(beta1, beta2), n_T=n_T)
- main.optim = torch.optim.Adam(ddpm.parameters(), lr=lr_initial)
- main.accelerator = Accelerator()

8.4.1 Detailed Description

Used to create and train DDPM .

Author

Created by J. Hughes on 18/03/2024.

Index