

RadioTherapy

Generated by Doxygen 1.13.2



<b>1 Namespace Index</b>	<b>1</b>
1.1 Package List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 GUI Namespace Reference	9
5.1.1 Variable Documentation	9
5.1.1.1 app	9
5.1.1.2 window	9
5.2 training Namespace Reference	9
5.2.1 Function Documentation	11
5.2.1.1 KL_loss()	11
5.2.1.2 select_channel()	11
5.2.2 Variable Documentation	11
5.2.2.1 adv_loss	11
5.2.2.2 adv_weight	11
5.2.2.3 autoencoder	12
5.2.2.4 autoencoder_warm_up_n_epochs	12
5.2.2.5 ax	12
5.2.2.6 axs	12
5.2.2.7 batch_size	12
5.2.2.8 channel	12
5.2.2.9 cmap	12
5.2.2.10 color	12
5.2.2.11 device	12
5.2.2.12 device_type	13
5.2.2.13 directory	13
5.2.2.14 disc_epoch_loss	13
5.2.2.15 discriminator	13
5.2.2.16 discriminator_loss	13
5.2.2.17 enabled	13
5.2.2.18 epoch_disc_loss_list	13
5.2.2.19 epoch_gen_loss_list	13
5.2.2.20 epoch_loss	13
5.2.2.21 epoch_loss_list	14
5.2.2.22 epoch_recon_loss_list	14

5.2.2.23 fig	14
5.2.2.24 first_batch	14
5.2.2.25 fontsize	14
5.2.2.26 force	14
5.2.2.27 gen_epoch_loss	14
5.2.2.28 generator_loss	14
5.2.2.29 idx	14
5.2.2.30 images	15
5.2.2.31 img	15
5.2.2.32 inferer	15
5.2.2.33 intermediary_images	15
5.2.2.34 kl_loss	15
5.2.2.35 kl_weight	15
5.2.2.36 l1_loss	15
5.2.2.37 label	15
5.2.2.38 linewidth	15
5.2.2.39 logits_fake	15
5.2.2.40 logits_real	16
5.2.2.41 loss	16
5.2.2.42 loss_d	16
5.2.2.43 loss_d_fake	16
5.2.2.44 loss_d_real	16
5.2.2.45 loss_g	16
5.2.2.46 loss_perceptual	16
5.2.2.47 n_epochs	16
5.2.2.48 n_example_images	17
5.2.2.49 ncols	17
5.2.2.50 noise	17
5.2.2.51 noise_pred	17
5.2.2.52 nrows	17
5.2.2.53 num_inference_steps	17
5.2.2.54 optimizer_d	17
5.2.2.55 optimizer_diff	17
5.2.2.56 optimizer_g	17
5.2.2.57 p_loss	18
5.2.2.58 perceptual_weight	18
5.2.2.59 progress_bar	18
5.2.2.60 prop	18
5.2.2.61 recons_loss	18
5.2.2.62 reconstruction	18
5.2.2.63 root_dir	18
5.2.2.64 scale_factor	18

5.2.2.65 scaler	18
5.2.2.66 scheduler	18
5.2.2.67 set_to_none	19
5.2.2.68 synthetic_images	19
5.2.2.69 timesteps	19
5.2.2.70 train_ds	19
5.2.2.71 train_loader	19
5.2.2.72 train_transforms	20
5.2.2.73 unet	20
5.2.2.74 val_interval	20
5.2.2.75 val_recon_epoch_loss_list	20
5.2.2.76 z	20
5.2.2.77 z_mu	20
5.2.2.78 z_sigma	20
<b>6 Class Documentation</b>	<b>21</b>
6.1 GUI.MainWindow Class Reference	21
6.1.1 Constructor & Destructor Documentation	21
6.1.1.1 __init__()	21
6.1.2 Member Function Documentation	21
6.1.2.1 open_file_dialog()	21
6.1.3 Member Data Documentation	22
6.1.3.1 close	22
6.1.3.2 open_file_dialog	22
<b>7 File Documentation</b>	<b>23</b>
7.1 GUI.py File Reference	23
7.2 training.py File Reference	23
<b>Index</b>	<b>27</b>



# Chapter 1

## Namespace Index

### 1.1 Package List

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

<a href="#">GUI</a> . . . . .	9
<a href="#">training</a> . . . . .	9





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

QMainWindow	
GUI.MainWindow . . . . .	<a href="#">21</a>



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">GUI.MainWindow</a> . . . . .	21
--	----



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">GUI.py</a>	.....	<a href="#">23</a>
<a href="#">training.py</a>	.....	<a href="#">23</a>



## Chapter 5

# Namespace Documentation

### 5.1 GUI Namespace Reference

#### Classes

- class [MainWindow](#)

#### Variables

- [app](#) = `QApplication(sys.argv)`
- [window](#) = `MainWindow()`

#### 5.1.1 Variable Documentation

##### 5.1.1.1 app

```
GUI.app = QApplication(sys.argv)
```

##### 5.1.1.2 window

```
GUI.window = MainWindow\(\)
```

### 5.2 training Namespace Reference

#### Functions

- [select\\_channel](#) (image, [channel](#)=0)
- [KL\\_loss](#) (z\_mu, [z\\_sigma](#))

## Variables

- `force`
- `directory = os.environ.get("MONAI_DATA_DIRECTORY")`
- `str root_dir = "/Users/giannigagliardi/Documents/Git/RadioTherapy/data"`
- `int batch_size = 1`
- `int channel = 0`
- `train_transforms`
- `train_ds`
- `train_loader`
- `device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`
- `autoencoder`
- `discriminator`
- `l1_loss = L1Loss()`
- `adv_loss = PatchAdversarialLoss(criterion="least_squares")`
- `loss_perceptual`
- `float adv_weight = 0.01`
- `float perceptual_weight = 0.001`
- `int kl_weight = 1e-6`
- `optimizer_g = torch.optim.Adam(params=autoencoder.parameters(), lr=1e-4)`
- `optimizer_d = torch.optim.Adam(params=discriminator.parameters(), lr=1e-4)`
- `int n_epochs = 10`
- `int autoencoder_warm_up_n_epochs = 5`
- `int val_interval = 10`
- `list epoch_recon_loss_list = []`
- `list epoch_gen_loss_list = []`
- `list epoch_disc_loss_list = []`
- `list val_recon_epoch_loss_list = []`
- `list intermediary_images = []`
- `int n_example_images = 4`
- `int epoch_loss = 0`
- `int gen_epoch_loss = 0`
- `int disc_epoch_loss = 0`
- `progress_bar = tqdm(enumerate(train_loader), total=len(train_loader), ncols=110)`
- `images = batch["image"].to(device)`
- `set_to_none`
- `reconstruction`
- `z_mu`
- `z_sigma`
- `kl_loss = KL_loss(z_mu, z_sigma)`
- `recons_loss = l1_loss(reconstruction.float(), images.float())`
- `p_loss = loss_perceptual(reconstruction.float(), images.float())`
- `int loss_g = recons_loss + kl_weight * kl_loss + perceptual_weight * p_loss`
- `logits_fake = discriminator(reconstruction.contiguous().float())[-1]`
- `generator_loss`
- `loss_d_fake`
- `logits_real = discriminator(images.contiguous().detach())[-1]`
- `loss_d_real`
- `tuple discriminator_loss = (loss_d_fake + loss_d_real) * 0.5`
- `float loss_d = adv_weight * discriminator_loss`
- `fontsize`
- `prop`
- `color`
- `linewidth`
- `label`



- `int idx = 0`
- `img = reconstruction[idx, channel].detach().cpu().numpy()`
- `fig`
- `axs`
- `nrows`
- `ncols`
- `ax = axs[0]`
- `cmap`
- `unet`
- `scheduler`
- `enabled`
- `first_batch = first(train_loader)`
- `z = autoencoder.encode_stage_2_inputs(first_batch["image"].to(device))`
- `int scale_factor = 1 / torch.std(z)`
- `inferer = LatentDiffusionInferer(scheduler, scale_factor=scale_factor)`
- `optimizer_diff = torch.optim.Adam(params=unet.parameters(), lr=1e-4)`
- `list epoch_loss_list = []`
- `scaler = GradScaler()`
- `device_type`
- `noise = torch.randn_like(z).to(device)`
- `timesteps`
- `noise_pred`
- `loss = F.mse_loss(noise_pred.float(), noise.float())`
- `num_inference_steps`
- `synthetic_images`

## 5.2.1 Function Documentation

### 5.2.1.1 KL\_loss()

```
training.KL_loss (
    z_mu,
    z_sigma)
```

### 5.2.1.2 select\_channel()

```
training.select_channel (
    image,
    channel = 0)
```

## 5.2.2 Variable Documentation

### 5.2.2.1 adv\_loss

```
training.adv_loss = PatchAdversarialLoss(criterion="least_squares")
```

### 5.2.2.2 adv\_weight

```
float training.adv_weight = 0.01
```

### 5.2.2.3 autoencoder

training.autoencoder

#### Initial value:

```
00001 = AutoencoderKL(  
00002     spatial_dims=3,  
00003     in_channels=1,  
00004     out_channels=1,  
00005     num_channels=(32, 32, 32),  
00006     latent_channels=2,  
00007     num_res_blocks=1,  
00008     norm_num_groups=8,  
00009     attention_levels=(False, False, True),  
00010 )
```

### 5.2.2.4 autoencoder\_warm\_up\_n\_epochs

```
int training.autoencoder_warm_up_n_epochs = 5
```

### 5.2.2.5 ax

```
training.ax = axis[0]
```

### 5.2.2.6 axs

training.axs

### 5.2.2.7 batch\_size

```
int training.batch_size = 1
```

### 5.2.2.8 channel

```
int training.channel = 0
```

### 5.2.2.9 cmap

training.cmap

### 5.2.2.10 color

training.color

### 5.2.2.11 device

```
training.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

#### 5.2.2.12 device\_type

```
training.device_type
```

#### 5.2.2.13 directory

```
training.directory = os.environ.get("MONAI_DATA_DIRECTORY")
```

#### 5.2.2.14 disc\_epoch\_loss

```
int training.disc_epoch_loss = 0
```

#### 5.2.2.15 discriminator

```
training.discriminator
```

##### Initial value:

```
00001 = PatchDiscriminator(  
00002     spatial_dims=3, num_layers_d=3, num_channels=32, in_channels=1, out_channels=1  
00003 )
```

#### 5.2.2.16 discriminator\_loss

```
tuple training.discriminator_loss = (loss_d_fake + loss_d_real) * 0.5
```

#### 5.2.2.17 enabled

```
training.enabled
```

#### 5.2.2.18 epoch\_disc\_loss\_list

```
training.epoch_disc_loss_list = []
```

#### 5.2.2.19 epoch\_gen\_loss\_list

```
training.epoch_gen_loss_list = []
```

#### 5.2.2.20 epoch\_loss

```
int training.epoch_loss = 0
```

**5.2.2.21 epoch\_loss\_list**

```
training.epoch_loss_list = []
```

**5.2.2.22 epoch\_recon\_loss\_list**

```
list training.epoch_recon_loss_list = []
```

**5.2.2.23 fig**

```
training.fig
```

**5.2.2.24 first\_batch**

```
training.first_batch = first(train\_loader)
```

**5.2.2.25 fontsize**

```
training.fontsize
```

**5.2.2.26 force**

```
training.force
```

**5.2.2.27 gen\_epoch\_loss**

```
int training.gen_epoch_loss = 0
```

**5.2.2.28 generator\_loss**

```
training.generator_loss
```

**Initial value:**

```
00001 = adv_loss(  
00002         logits_fake, target_is_real=True, for_discriminator=False  
00003     )
```

**5.2.2.29 idx**

```
int training.idx = 0
```

### 5.2.2.30 images

```
training.images = batch["image"].to(device)
```

### 5.2.2.31 img

```
training.img = reconstruction[idx, channel].detach().cpu().numpy()
```

### 5.2.2.32 inferer

```
training.inferer = LatentDiffusionInferer(scheduler, scale_factor=scale_factor)
```

### 5.2.2.33 intermediary\_images

```
list training.intermediary_images = []
```

### 5.2.2.34 kl\_loss

```
training.kl_loss = KL_loss(z_mu, z_sigma)
```

### 5.2.2.35 kl\_weight

```
int training.kl_weight = 1e-6
```

### 5.2.2.36 l1\_loss

```
training.l1_loss = L1Loss()
```

### 5.2.2.37 label

```
training.label
```

### 5.2.2.38 linewidth

```
training.linewidth
```

### 5.2.2.39 logits\_fake

```
training.logits_fake = discriminator(reconstruction.contiguous().float())[-1]
```

#### 5.2.2.40 logits\_real

```
training.logits_real = discriminator(images.contiguous().detach())[-1]
```

#### 5.2.2.41 loss

```
training.loss = F.mse_loss(noise_pred.float(), noise.float())
```

#### 5.2.2.42 loss\_d

```
float training.loss_d = adv_weight * discriminator_loss
```

#### 5.2.2.43 loss\_d\_fake

```
training.loss_d_fake
```

##### Initial value:

```
00001 = adv_loss(  
00002         logits_fake, target_is_real=False, for_discriminator=True  
00003     )
```

#### 5.2.2.44 loss\_d\_real

```
training.loss_d_real
```

##### Initial value:

```
00001 = adv_loss(  
00002         logits_real, target_is_real=True, for_discriminator=True  
00003     )
```

#### 5.2.2.45 loss\_g

```
int training.loss_g = recons_loss + kl_weight * kl_loss + perceptual_weight * p_loss
```

#### 5.2.2.46 loss\_perceptual

```
training.loss_perceptual
```

##### Initial value:

```
00001 = PerceptualLoss(  
00002     spatial_dims=3, network_type="squeeze", is_fake_3d=True, fake_3d_ratio=0.2  
00003 )
```

#### 5.2.2.47 n\_epochs

```
int training.n_epochs = 10
```

#### 5.2.2.48 n\_example\_images

```
int training.n_example_images = 4
```

#### 5.2.2.49 ncols

```
training.ncols
```

#### 5.2.2.50 noise

```
training.noise = torch.randn_like(z).to(device)
```

#### 5.2.2.51 noise\_pred

```
training.noise_pred
```

##### Initial value:

```
00001 = inferer(  
00002         inputs=images,  
00003         autoencoder_model=autoencoder,  
00004         diffusion_model=unet,  
00005         noise=noise,  
00006         timesteps=timesteps,  
00007     )
```

#### 5.2.2.52 n\_rows

```
training.nrows
```

#### 5.2.2.53 num\_inference\_steps

```
training.num_inference_steps
```

#### 5.2.2.54 optimizer\_d

```
training.optimizer_d = torch.optim.Adam(params=discriminator.parameters(), lr=1e-4)
```

#### 5.2.2.55 optimizer\_diff

```
training.optimizer_diff = torch.optim.Adam(params=unet.parameters(), lr=1e-4)
```

#### 5.2.2.56 optimizer\_g

```
training.optimizer_g = torch.optim.Adam(params=autoencoder.parameters(), lr=1e-4)
```

#### 5.2.2.57 p\_loss

```
training.p_loss = loss_perceptual(reconstruction.float(), images.float())
```

#### 5.2.2.58 perceptual\_weight

```
float training.perceptual_weight = 0.001
```

#### 5.2.2.59 progress\_bar

```
training.progress_bar = tqdm(enumerate(train_loader), total=len(train_loader), ncols=110)
```

#### 5.2.2.60 prop

```
training.prop
```

#### 5.2.2.61 recons\_loss

```
training.recons_loss = l1_loss(reconstruction.float(), images.float())
```

#### 5.2.2.62 reconstruction

```
training.reconstruction
```

#### 5.2.2.63 root\_dir

```
str training.root_dir = "/Users/giannigagliardi/Documents/Git/RadioTherapy/data"
```

#### 5.2.2.64 scale\_factor

```
int training.scale_factor = 1 / torch.std(z)
```

#### 5.2.2.65 scaler

```
training.scaler = GradScaler()
```

#### 5.2.2.66 scheduler

```
training.scheduler
```

##### Initial value:

```
00001 = DDPMScheduler(  
00002     num_train_timesteps=1000,  
00003     schedule="scaled_linear_beta",  
00004     beta_start=0.0015,  
00005     beta_end=0.0195,  
00006 )
```



### 5.2.2.67 set\_to\_none

```
training.set_to_none
```

### 5.2.2.68 synthetic\_images

```
training.synthetic_images
```

#### Initial value:

```
00001 = inferer.sample(  
00002     input_noise=noise,  
00003     autoencoder_model=autoencoder,  
00004     diffusion_model=unet,  
00005     scheduler=scheduler,  
00006 )
```

### 5.2.2.69 timesteps

```
training.timesteps
```

#### Initial value:

```
00001 = torch.randint(  
00002     0,  
00003     inferer.scheduler.num_train_timesteps,  
00004     (images.shape[0],),  
00005     device=images.device,  
00006 ).long()
```

### 5.2.2.70 train\_ds

```
training.train_ds
```

#### Initial value:

```
00001 = CustomDataset(  
00002     root_dir=root_dir,  
00003     section="training", # validation  
00004     cache_rate=0.0, # you may need a few Gb of RAM... Set to 0 otherwise  
00005     num_workers=0, # Set download to True if the dataset hasnt been downloaded yet  
00006     transform=train_transforms,  
00007     download=False,  
00008     seed=0,  
00009 )
```

### 5.2.2.71 train\_loader

```
training.train_loader
```

#### Initial value:

```
00001 = DataLoader(  
00002     train_ds,  
00003     batch_size=batch_size,  
00004     shuffle=True,  
00005     num_workers=0,  
00006     persistent_workers=False,  
00007 )
```

### 5.2.2.72 train\_transforms

```
training.train_transforms
```

#### Initial value:

```
00001 = transforms.Compose(
00002     [
00003         transforms.LoadImaged(keys=["image"], reader=NibabelReader),
00004         transforms.EnsureChannelFirstd(keys=["image"]),
00005         transforms.Lambdad(keys="image", func=select_channel),
00006         transforms.EnsureChannelFirstd(keys=["image"], channel_dim="no_channel"),
00007         transforms.EnsureTyped(keys=["image"]),
00008         transforms.Orientationd(keys=["image"], axcodes="RAS"),
00009         transforms.Spacingd(keys=["image"], pixdim=(2.4, 2.4, 2.2), mode="bilinear"),
00010         SpatialPadd(keys=["image"], spatial_size=(32, 32, 32), method='symmetric'),
00011         transforms.CenterSpatialCropd(keys=["image"], roi_size=(32, 32, 32)),
00012         transforms.ScaleIntensityRangePercentilesd(
00013             keys="image", lower=0, upper=99.5, b_min=0, b_max=1
00014         ),
00015     ]
00016 )
```

### 5.2.2.73 unet

```
training.unet
```

#### Initial value:

```
00001 = DiffusionModelUNet(
00002     spatial_dims=3,
00003     in_channels=2,
00004     out_channels=2,
00005     num_res_blocks=1,
00006     num_channels=(32, 64, 64),
00007     attention_levels=(False, True, True),
00008     num_head_channels=(0, 64, 64),
00009 )
```

### 5.2.2.74 val\_interval

```
int training.val_interval = 10
```

### 5.2.2.75 val\_recon\_epoch\_loss\_list

```
list training.val_recon_epoch_loss_list = []
```

### 5.2.2.76 z

```
training.z = autoencoder.encode_stage_2_inputs(first_batch["image"].to(device))
```

### 5.2.2.77 z\_mu

```
training.z_mu
```

### 5.2.2.78 z\_sigma

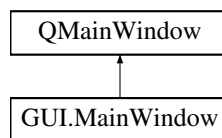
```
training.z_sigma
```

## Chapter 6

# Class Documentation

### 6.1 GUI.MainWindow Class Reference

Inheritance diagram for GUI.MainWindow:



#### Public Member Functions

- `__init__` (self)
- `open_file_dialog` (self)

#### Public Attributes

- `open_file_dialog` = `QPushButton("CT-Scan hochladen")`
- `close` = `QWidgetAction(self)`

#### 6.1.1 Constructor & Destructor Documentation

##### 6.1.1.1 `__init__()`

```
GUI.MainWindow.__init__ (  
    self)
```

#### 6.1.2 Member Function Documentation

##### 6.1.2.1 `open_file_dialog()`

```
GUI.MainWindow.open_file_dialog (  
    self)
```

## 6.1.3 Member Data Documentation

### 6.1.3.1 close

```
GUI.MainWindow.close = QWidgetAction(self)
```

### 6.1.3.2 open\_file\_dialog

```
GUI.MainWindow.open_file_dialog = QPushButton("CT-Scan hochladen")
```

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

- [GUI.py](#)

# Chapter 7

## File Documentation

### 7.1 GUI.py File Reference

#### Classes

- class [GUI.MainWindow](#)

#### Namespaces

- namespace [GUI](#)

#### Variables

- [GUI.app](#) = QApplication(sys.argv)
- [GUI.window](#) = [MainWindow\(\)](#)

### 7.2 training.py File Reference

#### Namespaces

- namespace [training](#)

#### Functions

- [training.select\\_channel](#) (image, [channel](#)=0)
- [training.KL\\_loss](#) ([z\\_mu](#), [z\\_sigma](#))

## Variables

- `training.force`
- `training.directory = os.environ.get("MONAI_DATA_DIRECTORY")`
- `str training.root_dir = "/Users/giannigagliardi/Documents/Git/RadioTherapy/data"`
- `int training.batch_size = 1`
- `int training.channel = 0`
- `training.train_transforms`
- `training.train_ds`
- `training.train_loader`
- `training.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")`
- `training.autoencoder`
- `training.discriminator`
- `training.l1_loss = L1Loss()`
- `training.adv_loss = PatchAdversarialLoss(criterion="least_squares")`
- `training.loss_perceptual`
- `float training.adv_weight = 0.01`
- `float training.perceptual_weight = 0.001`
- `int training.kl_weight = 1e-6`
- `training.optimizer_g = torch.optim.Adam(params=autoencoder.parameters(), lr=1e-4)`
- `training.optimizer_d = torch.optim.Adam(params=discriminator.parameters(), lr=1e-4)`
- `int training.n_epochs = 10`
- `int training.autoencoder_warm_up_n_epochs = 5`
- `int training.val_interval = 10`
- `list training.epoch_recon_loss_list = []`
- `list training.epoch_gen_loss_list = []`
- `list training.epoch_disc_loss_list = []`
- `list training.val_recon_epoch_loss_list = []`
- `list training.intermediary_images = []`
- `int training.n_example_images = 4`
- `int training.epoch_loss = 0`
- `int training.gen_epoch_loss = 0`
- `int training.disc_epoch_loss = 0`
- `training.progress_bar = tqdm(enumerate(train_loader), total=len(train_loader), ncols=110)`
- `training.images = batch["image"].to(device)`
- `training.set_to_none`
- `training.reconstruction`
- `training.z_mu`
- `training.z_sigma`
- `training.kl_loss = KL_loss(z_mu, z_sigma)`
- `training.recons_loss = l1_loss(reconstruction.float(), images.float())`
- `training.p_loss = loss_perceptual(reconstruction.float(), images.float())`
- `int training.loss_g = recons_loss + kl_weight * kl_loss + perceptual_weight * p_loss`
- `training.logits_fake = discriminator(reconstruction.contiguous().float())[-1]`
- `training.generator_loss`
- `training.loss_d_fake`
- `training.logits_real = discriminator(images.contiguous().detach())[-1]`
- `training.loss_d_real`
- `tuple training.discriminator_loss = (loss_d_fake + loss_d_real) * 0.5`
- `float training.loss_d = adv_weight * discriminator_loss`
- `training.fontsize`
- `training.prop`
- `training.color`
- `training.linewidth`
- `training.label`

- `int training.idx = 0`
- `training.img = reconstruction[idx, channel].detach().cpu().numpy()`
- `training.fig`
- `training.axs`
- `training.nrows`
- `training.ncols`
- `training.ax = axs[0]`
- `training.cmap`
- `training.unet`
- `training.scheduler`
- `training.enabled`
- `training.first_batch = first(train_loader)`
- `training.z = autoencoder.encode_stage_2_inputs(first_batch["image"].to(device))`
- `int training.scale_factor = 1 / torch.std(z)`
- `training.inferer = LatentDiffusionInferer(scheduler, scale_factor=scale_factor)`
- `training.optimizer_diff = torch.optim.Adam(params=unet.parameters(), lr=1e-4)`
- `list training.epoch_loss_list = []`
- `training.scaler = GradScaler()`
- `training.device_type`
- `training.noise = torch.randn_like(z).to(device)`
- `training.timesteps`
- `training.noise_pred`
- `training.loss = F.mse_loss(noise_pred.float(), noise.float())`
- `training.num_inference_steps`
- `training.synthetic_images`





# Index

- `__init__`
  - `GUI.MainWindow`, [21](#)
- `adv_loss`
  - `training`, [11](#)
- `adv_weight`
  - `training`, [11](#)
- `app`
  - `GUI`, [9](#)
- `autoencoder`
  - `training`, [11](#)
- `autoencoder_warm_up_n_epochs`
  - `training`, [12](#)
- `ax`
  - `training`, [12](#)
- `axs`
  - `training`, [12](#)
- `batch_size`
  - `training`, [12](#)
- `channel`
  - `training`, [12](#)
- `close`
  - `GUI.MainWindow`, [22](#)
- `cmap`
  - `training`, [12](#)
- `color`
  - `training`, [12](#)
- `device`
  - `training`, [12](#)
- `device_type`
  - `training`, [12](#)
- `directory`
  - `training`, [13](#)
- `disc_epoch_loss`
  - `training`, [13](#)
- `discriminator`
  - `training`, [13](#)
- `discriminator_loss`
  - `training`, [13](#)
- `enabled`
  - `training`, [13](#)
- `epoch_disc_loss_list`
  - `training`, [13](#)
- `epoch_gen_loss_list`
  - `training`, [13](#)
- `epoch_loss`
  - `training`, [13](#)
- `epoch_loss_list`
  - `training`, [13](#)
- `epoch_recon_loss_list`
  - `training`, [14](#)
- `fig`
  - `training`, [14](#)
- `first_batch`
  - `training`, [14](#)
- `fontsize`
  - `training`, [14](#)
- `force`
  - `training`, [14](#)
- `gen_epoch_loss`
  - `training`, [14](#)
- `generator_loss`
  - `training`, [14](#)
- `GUI`, [9](#)
  - `app`, [9](#)
  - `window`, [9](#)
- `GUI.MainWindow`, [21](#)
  - `__init__`, [21](#)
  - `close`, [22](#)
  - `open_file_dialog`, [21](#), [22](#)
- `GUI.py`, [23](#)
- `idx`
  - `training`, [14](#)
- `images`
  - `training`, [14](#)
- `img`
  - `training`, [15](#)
- `inferer`
  - `training`, [15](#)
- `intermediary_images`
  - `training`, [15](#)
- `KL_loss`
  - `training`, [11](#)
- `kl_loss`
  - `training`, [15](#)
- `kl_weight`
  - `training`, [15](#)
- `l1_loss`
  - `training`, [15](#)
- `label`
  - `training`, [15](#)
- `linewidth`
  - `training`, [15](#)

- logits\_fake
  - training, [15](#)
- logits\_real
  - training, [15](#)
- loss
  - training, [16](#)
- loss\_d
  - training, [16](#)
- loss\_d\_fake
  - training, [16](#)
- loss\_d\_real
  - training, [16](#)
- loss\_g
  - training, [16](#)
- loss\_perceptual
  - training, [16](#)
- n\_epochs
  - training, [16](#)
- n\_example\_images
  - training, [16](#)
- ncols
  - training, [17](#)
- noise
  - training, [17](#)
- noise\_pred
  - training, [17](#)
- nrows
  - training, [17](#)
- num\_inference\_steps
  - training, [17](#)
- open\_file\_dialog
  - GUI.MainWindow, [21](#), [22](#)
- optimizer\_d
  - training, [17](#)
- optimizer\_diff
  - training, [17](#)
- optimizer\_g
  - training, [17](#)
- p\_loss
  - training, [17](#)
- perceptual\_weight
  - training, [18](#)
- progress\_bar
  - training, [18](#)
- prop
  - training, [18](#)
- recons\_loss
  - training, [18](#)
- reconstruction
  - training, [18](#)
- root\_dir
  - training, [18](#)
- scale\_factor
  - training, [18](#)
- scaler
  - training, [18](#)
- scheduler
  - training, [18](#)
- select\_channel
  - training, [11](#)
- set\_to\_none
  - training, [18](#)
- synthetic\_images
  - training, [19](#)
- timesteps
  - training, [19](#)
- train\_ds
  - training, [19](#)
- train\_loader
  - training, [19](#)
- train\_transforms
  - training, [19](#)
- training, [9](#)
  - adv\_loss, [11](#)
  - adv\_weight, [11](#)
  - autoencoder, [11](#)
  - autoencoder\_warm\_up\_n\_epochs, [12](#)
  - ax, [12](#)
  - axs, [12](#)
  - batch\_size, [12](#)
  - channel, [12](#)
  - cmap, [12](#)
  - color, [12](#)
  - device, [12](#)
  - device\_type, [12](#)
  - directory, [13](#)
  - disc\_epoch\_loss, [13](#)
  - discriminator, [13](#)
  - discriminator\_loss, [13](#)
  - enabled, [13](#)
  - epoch\_disc\_loss\_list, [13](#)
  - epoch\_gen\_loss\_list, [13](#)
  - epoch\_loss, [13](#)
  - epoch\_loss\_list, [13](#)
  - epoch\_recon\_loss\_list, [14](#)
  - fig, [14](#)
  - first\_batch, [14](#)
  - fontsize, [14](#)
  - force, [14](#)
  - gen\_epoch\_loss, [14](#)
  - generator\_loss, [14](#)
  - idx, [14](#)
  - images, [14](#)
  - img, [15](#)
  - inferer, [15](#)
  - intermediary\_images, [15](#)
  - KL\_loss, [11](#)
  - kl\_loss, [15](#)
  - kl\_weight, [15](#)
  - l1\_loss, [15](#)
  - label, [15](#)
  - linewidth, [15](#)

- logits\_fake, 15
- logits\_real, 15
- loss, 16
- loss\_d, 16
- loss\_d\_fake, 16
- loss\_d\_real, 16
- loss\_g, 16
- loss\_perceptual, 16
- n\_epochs, 16
- n\_example\_images, 16
- ncols, 17
- noise, 17
- noise\_pred, 17
- nrows, 17
- num\_inference\_steps, 17
- optimizer\_d, 17
- optimizer\_diff, 17
- optimizer\_g, 17
- p\_loss, 17
- perceptual\_weight, 18
- progress\_bar, 18
- prop, 18
- recons\_loss, 18
- reconstruction, 18
- root\_dir, 18
- scale\_factor, 18
- scaler, 18
- scheduler, 18
- select\_channel, 11
- set\_to\_none, 18
- synthetic\_images, 19
- timesteps, 19
- train\_ds, 19
- train\_loader, 19
- train\_transforms, 19
- unet, 20
- val\_interval, 20
- val\_recon\_epoch\_loss\_list, 20
- z, 20
- z\_mu, 20
- z\_sigma, 20
- training.py, 23
- unet
  - training, 20
- val\_interval
  - training, 20
- val\_recon\_epoch\_loss\_list
  - training, 20
- window
  - GUI, 9
- z
  - training, 20
- z\_mu
  - training, 20
- z\_sigma
  - training, 20