Callbacks

Contents

Callbacks for Anomalib models.

class **anomalib.callbacks.GraphLogger**

Bases: `Callback`

Log model graph to respective logger.

Examples

Log model graph to Tensorboard

```
>>> from anomalib.callbacks import GraphLogger
>>> from anomalib.loggers import AnomalibTensorBoardLogger
>>> from anomalib.engine import Engine
...
>>> logger = AnomalibTensorBoardLogger()
>>> callbacks = [GraphLogger()]
>>> engine = Engine(logger=logger, callbacks=callbacks)
```

Log model graph to Comet

```
>>> from anomalib.loggers import AnomalibCometLogger
>>> from anomalib.engine import Engine
...
>>> logger = AnomalibCometLogger()
>>> callbacks = [GraphLogger()]
>>> engine = Engine(logger=logger, callbacks=callbacks)
```

`on_train_end(trainer, pl_module)`

Unwatch model if configured for wandb and log it model graph in Tensorboard if specified.

Parameters:

- **trainer** (`Trainer`) – Trainer object which contans reference to loggers.
- **pl_module** (`LightningModule`) – LightningModule object which is logged.

Return type:

`None`

`on_train_start(trainer, pl_module)`

Log model graph to respective logger.

Parameters:

- **trainer** (`Trainer`) – Trainer object which contans reference to loggers.
- **pl_module** (`LightningModule`) – LightningModule object which is logged.

Return type:

`None`

class `anomalib.callbacks.LoadModelCallback(weights_path)`

Bases: `Callback`

Callback that loads the model weights from the state dict.

Examples

```
>>> from anomalib.callbacks import LoadModelCallback
>>> from anomalib.engine import Engine
...
>>> callbacks = [LoadModelCallback(weights_path="path/to/weights.pt")]
>>> engine = Engine(callbacks=callbacks)
```

setup(*trainer*, *pl_module*, *stage=None*)

Call when inference begins.

Loads the model weights from `weights_path` into the PyTorch module.

Return type:

`None`

class **anomalib.callbacks.ModelCheckpoint**(*dirpath=None*, *filename=None*, *monitor=None*, *verbose=False*, *save_last=None*, *save_top_k=1*, *save_weights_only=False*, *mode='min'*, *auto_insert_metric_name=True*, *every_n_train_steps=None*, *train_time_interval=None*, *every_n_epochs=None*, *save_on_train_epoch_end=None*, *enable_version_counter=True*)

Bases: `ModelCheckpoint`

Anomalib Model Checkpoint Callback.

This class overrides the Lightning ModelCheckpoint callback to enable saving checkpoints without running any training steps. This is useful for zero-/few-shot models, where the fit sequence only consists of validation.

To enable saving checkpoints without running any training steps, we need to override two checks which are being called in the `on_validation_end` method of the parent class: - `_should_save_on_train_epoch_end`: This method checks whether the checkpoint should be saved at the end of a

> training epoch, or at the end of the validation sequence. We modify this method to default to saving at the end of the validation sequence when the model is of zero- or few-shot type, unless `save_on_train_epoch_end` is specifically set by the user.

- `_should_skip_saving_checkpoint`**: This method checks whether the checkpoint**

should be saved at all. We modify

this method to allow saving during both the `FITTING` and `VALIDATING` states. In addition, we allow saving if the global step has not changed since the last checkpoint, but only for zero- and few-shot models. This is needed because both the last global step and the last checkpoint remain unchanged during zero-/few-shot training, which would otherwise prevent saving checkpoints during validation.

class anomalib.callbacks.TimerCallback

Bases: `Callback`

Callback that measures the training and testing time of a PyTorch Lightning module.

Examples

```
>>> from anomalib.callbacks import TimerCallback
>>> from anomalib.engine import Engine
...
>>> callbacks = [TimerCallback()]
>>> engine = Engine(callbacks=callbacks)
```

on_fit_end(*trainer*, *pl_module*)

Call when fit ends.

Prints the time taken for training.

Parameters:

- **trainer** (*Trainer*) – PyTorch Lightning trainer.
- **pl_module** (*LightningModule*) – Current training module.

Return type:

`None`

Returns:

None

on_fit_start(*trainer*, *pl_module*)

Call when fit begins.

Sets the start time to the time training started.

Parameters:

- **trainer** (*Trainer*) – PyTorch Lightning trainer.
- **pl_module** (*LightningModule*) – Current training module.

Return type:

`None`

Returns:

None

`on_test_end(trainer, pl_module)`

Call when the test ends.

Prints the time taken for testing and the throughput in frames per second.

Parameters:

- **trainer** (*Trainer*) – PyTorch Lightning trainer.
- **pl_module** (*LightningModule*) – Current training module.

Return type:

`None`

Returns:

None

`on_test_start(trainer, pl_module)`

Call when the test begins.

Sets the start time to the time testing started. Goes over all the test dataloaders and adds the number of images in each.

Parameters:

- **trainer** (*Trainer*) – PyTorch Lightning trainer.
- **pl_module** (*LightningModule*) – Current training module.

Return type:

`None`

Returns:

None

‹ Previous [Loggers]

Next [CLI] ›