Padim

Contents

PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization.

Paper https://arxiv.org/abs/2011.08785

class **anomalib.models.image.padim.lightning_model.Padim(*backbone='resnet18', layers=['layer1', 'layer2', 'layer3'], pre_trained=True, n_features=None*)**

Bases: `MemoryBankMixin`, `AnomalyModule`

PaDiM: a Patch Distribution Modeling Framework for Anomaly Detection and Localization.

Parameters:

- **backbone** (*str*) – Backbone CNN network Defaults to `resnet18`.
- **layers** (*list[str]*) – Layers to extract features from the backbone CNN Defaults to `["layer1", "layer2", "layer3"]`.
- **pre_trained** (*bool, optional*) – Boolean to check whether to use a pre_trained backbone. Defaults to `True`.
- **n_features** (*int, optional*) – Number of features to retain in the dimension reduction step. Default values from the paper are available for: resnet18 (100), wide_resnet50_2 (550). Defaults to `None`.

static **configure_optimizers()**

PADIM doesn't require optimization, therefore returns no optimizers.

Return type:

`None`

configure_transforms(*image_size=None*)

Default transform for Padim.

Return type:

`Transform`

fit()

Fit a Gaussian to the embedding collected from the training set.

Return type:

`None`

property learning_type: *LearningType*

Return the learning type of the model.

Returns:

Learning type of the model.

Return type:

LearningType

property trainer_arguments: *dict[str, int | float]*

Return PADIM trainer arguments.

Since the model does not require training, we limit the max_epochs to 1. Since we need to run training epoch before validation, we also set the sanity steps to 0

training_step(*batch*, **args*, ***kwargs*)

Perform the training step of PADIM. For each batch, hierarchical features are

extracted from the CNN.

Parameters:

- **batch** (*dict[str, str | torch.Tensor]*) – Batch containing image filename, image, label and mask
- **args** – Additional arguments.
- **kwargs** – Additional keyword arguments.

Return type:

`None`

Returns:

Hierarchical feature map

`validation_step(batch, *args, **kwargs)`

Perform a validation step of PADIM.

Similar to the training step, hierarchical features are extracted from the CNN for each batch.

Parameters:

- **batch** (*dict[str, str | torch.Tensor]*) – Input batch
- **args** – Additional arguments.
- **kwargs** – Additional keyword arguments.

Return type:

`Union`[`Tensor`, `Mapping`[`str`, `Any`], `None`]

Returns:

Dictionary containing images, features, true labels and masks. These are required in *validation_epoch_end* for feature concatenation.

PyTorch model for the PaDiM model implementation.

`class anomalib.models.image.padim.torch_model.PadimModel(layers, backbone='resnet18', pre_trained=True, n_features=None)`

Bases: `Module`

Padim Module.

Parameters:

- **layers** (*list[str]*) – Layers used for feature extraction
- **backbone** (*str, optional*) – Pre-trained model backbone. Defaults to "resnet18". Defaults to `resnet18`.
- **pre_trained** (*bool, optional*) – Boolean to check whether to use a pre_trained backbone. Defaults to `True`.
- **n_features** (*int, optional*) – Number of features to retain in the dimension reduction step. Default values from the paper are available for: resnet18 (100), wide_resnet50_2 (550). Defaults to `None`.

`forward(input_tensor)`

Forward-pass image-batch (N, C, H, W) into model to extract features.

Parameters:

- **input_tensor** (`Tensor`) – Image-batch (N, C, H, W)
- **input_tensor** – torch.Tensor:

Return type:

`Tensor`

Returns:

Features from single/multiple layers.

Example

```
>>> x = torch.randn(32, 3, 224, 224)
>>> features = self.extract_features(input_tensor)
>>> features.keys()
dict_keys(['layer1', 'layer2', 'layer3'])
```

```
>>> [v.shape for v in features.values()]
[torch.Size([32, 64, 56, 56]),
torch.Size([32, 128, 28, 28]),
torch.Size([32, 256, 14, 14])]
```

`generate_embedding(features)`

Generate embedding from hierarchical feature map.

Parameters:

features (*dict[str, torch.Tensor]*) – Hierarchical feature map from a CNN (ResNet18 or WideResnet)

Return type:

`Tensor`

Returns:

Embedding vector

Previous
GANomaly

Next
PatchCore