STFPM

Contents

STFPM: Student-Teacher Feature Pyramid Matching for Unsupervised Anomaly Detection.

https://arxiv.org/abs/2103.04257

class **anomalib.models.image.stfpm.lightning_model.Stfpm(*backbone='resnet18', layers=('layer1', 'layer2', 'layer3')*)**

Bases: `AnomalyModule`

PL Lightning Module for the STFPM algorithm.

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"]`.

configure_optimizers()

Configure optimizers.

Returns:

SGD optimizer

Return type:

Optimizer

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, Any]*

Required trainer arguments.

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

Perform a training step of STFPM.

For each batch, teacher and student and teacher features are extracted from the CNN.

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:

Loss value

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

Perform a validation Step of STFPM.

Similar to the training step, student/teacher features are extracted from the CNN for each batch, and anomaly map is computed.

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, anomaly maps, true labels and masks. These are required in *validation_epoch_end* for feature concatenation.

PyTorch model for the STFPM model implementation.

class anomalib.models.image.stfpm.torch_model.STFPMModel(*layers*, *backbone='resnet18'*)

Bases: `Module`

STFPM: Student-Teacher Feature Pyramid Matching for Unsupervised Anomaly Detection.

Parameters:

- **layers** (*list[str]*) – Layers used for feature extraction.
- **backbone** (*str, optional*) – Pre-trained model backbone. Defaults to `resnet18`.

forward(*images*)

Forward-pass images into the network.

During the training mode the model extracts the features from the teacher and student networks. During the evaluation mode, it returns the predicted anomaly map.

Parameters:

images (*torch.Tensor*) – Batch of images.

Return type:

`Tensor` | `dict`[`str`, `Tensor`] | `tuple`[`dict`[`str`, `Tensor`]]

Returns:

Teacher and student features when in training mode, otherwise the predicted anomaly maps.

Loss function for the STFPM Model Implementation.

class anomalib.models.image.stfpm.loss.STFPMLoss

Bases: `Module`

Feature Pyramid Loss This class implmenents the feature pyramid loss function proposed in STFPM paper.

Example

```
>>> from anomalib.models.components.feature_extractors import TimmFeatureExtrac
>>> from anomalib.models.stfpm.loss import STFPMLoss
>>> from torchvision.models import resnet18
```

```
>>> layers = ['layer1', 'layer2', 'layer3']
>>> teacher_model = TimmFeatureExtractor(model=resnet18(pretrained=True), layer
>>> student_model = TimmFeatureExtractor(model=resnet18(pretrained=False), laye
>>> loss = Loss()
```

```
>>> inp = torch.rand((4, 3, 256, 256))
>>> teacher_features = teacher_model(inp)
>>> student_features = student_model(inp)
>>> loss(student_features, teacher_features)
    tensor(51.2015, grad_fn=<SumBackward0>)
```

compute_layer_loss(*teacher_feats*, *student_feats*)

Compute layer loss based on Equation (1) in Section 3.2 of the paper.

Parameters:

- **teacher_feats** (*torch.Tensor*) – Teacher features
- **student_feats** (*torch.Tensor*) – Student features

Return type:

`Tensor`

Returns:

L2 distance between teacher and student features.

forward(*teacher_features*, *student_features*)

Compute the overall loss via the weighted average of the layer losses computed by the cosine similarity.

Parameters:

- **teacher_features** (*dict[str, torch.Tensor]*) – Teacher features
- **student_features** (*dict[str, torch.Tensor]*) – Student features

Return type:

`Tensor`

Returns:

Total loss, which is the weighted average of the layer losses.

Anomaly Map Generator for the STFPM model implementation.

class anomalib.models.image.stfpm.anomaly_map.AnomalyMapGenerator

Bases: `Module`

Generate Anomaly Heatmap.

compute_anomaly_map(*teacher_features*, *student_features*, *image_size*)

Compute the overall anomaly map via element-wise production the interpolated anomaly maps.

Parameters:

- **teacher_features** (*dict[str, torch.Tensor]*) – Teacher features
- **student_features** (*dict[str, torch.Tensor]*) – Student features
- **image_size** (*tuple[int, int]*) – Image size to which the anomaly map should be resized.

Return type:

`Tensor`

Returns:

Final anomaly map

compute_layer_map(*teacher_features*, *student_features*, *image_size*)

Compute the layer map based on cosine similarity.

Parameters:

- **teacher_features** (*torch.Tensor*) – Teacher features
- **student_features** (*torch.Tensor*) – Student features
- **image_size** (*tuple[int, int]*) – Image size to which the anomaly map should be resized.

Return type:

`Tensor`

Returns:

Anomaly score based on cosine similarity.

forward(****kwargs**)

Return anomaly map.

Expects *teach_features* and *student_features* keywords to be passed explicitly.

Parameters:

kwargs (*dict[str, torch.Tensor]*) – Keyword arguments

Example

```
>>> anomaly_map_generator = AnomalyMapGenerator(image_size=tuple(hparams.mc
>>> output = self.anomaly_map_generator(
        teacher_features=teacher_features,
        student_features=student_features
    )
```

Raises:

ValueError – *teach_features* and *student_features* keys are not found

Returns:

anomaly map

Return type:

torch.Tensor

Previous
R-KDE

Next
U-Flow