CFA

Print to PDF

Contents

Lightning Implementatation of the CFA Model.

CFA: Coupled-hypersphere-based Feature Adaptation for Target-Oriented Anomaly Localization

Paper https://arxiv.org/abs/2206.04325

class **anomalib.models.image.cfa.lightning_model.Cfa**(***backbone='wide_resnet50_2'***, ***gamma_c=1***, ***gamma_d=1***, ***num_nearest_neighbors=3***, ***num_hard_negative_features=3***, ***radius=1e-05***)

Bases: `AnomalyModule`

CFA: Coupled-hypersphere-based Feature Adaptation for Target-Oriented Anomaly Localization.

Parameters:

- **backbone** (*str*) – Backbone CNN network Defaults to `"wide_resnet50_2"`.
- **gamma_c** (*int, optional*) – gamma_c value from the paper. Defaults to `1`.
- **gamma_d** (*int, optional*) – gamma_d value from the paper. Defaults to `1`.
- **num_nearest_neighbors** (*int*) – Number of nearest neighbors. Defaults to `3`.
- **num_hard_negative_features** (*int*) – Number of hard negative features. Defaults to `3`.
- **radius** (*float*) – Radius of the hypersphere to search the soft boundary. Defaults to `1e-5`.

backward(***loss***, ****args***, *****kwargs***)

Perform backward-pass for the CFA model.

Parameters:

- **loss** (*torch.Tensor*) – Loss value.
- ***args** – Arguments.
- ****kwargs** – Keyword arguments.

Return type:

None

configure_optimizers()

Configure optimizers for the CFA Model.

Returns:

Adam optimizer for each decoder

Return type:

Optimizer

property learning_type: *LearningType*

Return the learning type of the model.

Returns:

Learning type of the model.

Return type:

LearningType

on_train_start()

Initialize the centroid for the memory bank computation.

Return type:

None

property trainer_arguments: *dict[str, Any]*

CFA specific trainer arguments.

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

Perform the training step for the CFA model.

Parameters:

- **batch** (*dict[str, str | torch.Tensor]*) – Batch input.
- ***args** – Arguments.
- ****kwargs** – Keyword arguments.

Returns:

Loss value.

Return type:

STEP_OUTPUT

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

Perform the validation step for the CFA model.

Parameters:

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

Returns:

Anomaly map computed by the model.

Return type:

dict

Torch Implementatation of the CFA Model.

CFA: Coupled-hypersphere-based Feature Adaptation for Target-Oriented Anomaly Localization

Paper https://arxiv.org/abs/2206.04325

class anomalib.models.image.cfa.torch_model.CfaModel(*backbone*, *gamma_c*, *gamma_d*, *num_nearest_neighbors*, *num_hard_negative_features*, *radius*)

Bases: `DynamicBufferMixin`

Torch implementation of the CFA Model.

Parameters:

- **backbone** (*str*) – Backbone CNN network.
- **gamma_c** (*int*) – gamma_c parameter from the paper.
- **gamma_d** (*int*) – gamma_d parameter from the paper.
- **num_nearest_neighbors** (*int*) – Number of nearest neighbors.
- **num_hard_negative_features** (*int*) – Number of hard negative features.
- **radius** (*float*) – Radius of the hypersphere to search the soft boundary.

`compute_distance(target_oriented_features)`

Compute distance using target oriented features.

Parameters:

target_oriented_features (*torch.Tensor*) – Target oriented features computed using the descriptor.

Returns:

Distance tensor.

Return type:

Tensor

`forward(input_tensor)`

Forward pass.

Parameters:

input_tensor (*torch.Tensor*) – Input tensor.

Raises:

ValueError – When the memory bank is not initialized.

Returns:

Loss or anomaly map depending on the train/eval mode.

Return type:

Tensor

`get_scale(input_size)`

Get the scale of the feature map.

Parameters:

input_size (*tuple[int, int]*) – Input size of the image tensor.

Return type:

`Size`

initialize_centroid(*data_loader*)

Initialize the Centroid of the Memory Bank.

Parameters:

data_loader (*DataLoader*) – Train Dataloader.

Returns:

Memory Bank.

Return type:

Tensor

Loss function for the Cfa Model Implementation.

class anomalib.models.image.cfa.loss.CfaLoss(*num_nearest_neighbors*, *num_hard_negative_features*, *radius*)

Bases: `Module`

Cfa Loss.

Parameters:

- **num_nearest_neighbors** (*int*) – Number of nearest neighbors.
- **num_hard_negative_features** (*int*) – Number of hard negative features.
- **radius** (*float*) – Radius of the hypersphere to search the soft boundary.

forward(*distance*)

Compute the CFA loss.

Parameters:

distance (*torch.Tensor*) – Distance computed using target oriented features.

Returns:

CFA loss.

Return type:

Tensor

Anomaly Map Generator for the CFA model implementation.

class **anomalib.models.image.cfa.anomaly_map.AnomalyMapGenerator(*num_nearest_neighbors*, *sigma=4*)**

Bases: `Module`

Generate Anomaly Heatmap.

compute_anomaly_map(*score*, *image_size=None*)

Compute anomaly map based on the score.

Parameters:

- **score** (*torch.Tensor*) – Score tensor.
- **image_size** (*tuple[int, int] | torch.Size | None, optional*) – Size of the input image.

Returns:

Anomaly map.

Return type:

Tensor

compute_score(*distance*, *scale*)

Compute score based on the distance.

Parameters:

- **distance** (*torch.Tensor*) – Distance tensor computed using target oriented features.
- **scale** (*tuple[int, int]*) – Height and width of the largest feature map.

Returns:

Score value.

Return type:

Tensor

forward(*kwargs*)**

Return anomaly map.

Raises:

distance` and scale keys are not found –

Returns:

Anomaly heatmap.

Return type:

Tensor

Previous
Image Models

Next
C-Flow