DFKDE

Contents

DFKDE: Deep Feature Kernel Density Estimation.

class **anomalib.models.image.dfkde.lightning_model.Dfkde(*backbone='resnet18', layers=('layer4',), pre_trained=True, n_pca_components=16, feature_scaling_method=FeatureScalingMethod.SCALE, max_training_points=40000*)**

Bases: `MemoryBankMixin`, `AnomalyModule`

DFKDE: Deep Feature Kernel Density Estimation.

Parameters:

- **backbone** (*str*) – Pre-trained model backbone. Defaults to `"resnet18"`.
- **layers** (*Sequence[str], optional*) – Layers to extract features from. Defaults to `("layer4",)`.
- **pre_trained** (*bool, optional*) – Boolean to check whether to use a pre_trained backbone. Defaults to `True`.
- **n_pca_components** (*int, optional*) – Number of PCA components. Defaults to `16`.
- **feature_scaling_method** (*FeatureScalingMethod, optional*) – Feature scaling method. Defaults to `FeatureScalingMethod.SCALE`.
- **max_training_points** (*int, optional*) – Number of training points to fit the KDE model. Defaults to `40000`.

static **configure_optimizers()**

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

Return type:

None

fit()

Fit a KDE Model 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, Any]*

Return DFKDE-specific trainer arguments.

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

Perform the training step of DFKDE. For each batch, features are extracted from the CNN.

Parameters:

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

Return type:

`None`

Returns:

Deep CNN features.

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

Perform the validation step of DFKDE.

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

Parameters:

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

Return type:

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

Returns:

Dictionary containing probability, prediction and ground truth values.

Normality model of DFKDE.

class anomalib.models.image.dfkde.torch_model.DfkdeModel(*backbone*, *layers*, *pre_trained=True*, *n_pca_components=16*, *feature_scaling_method=FeatureScalingMethod.SCALE*, *max_training_points=40000*)

Bases: `Module`

Normality Model for the DFKDE algorithm.

Parameters:

- **backbone** (*str*) – Pre-trained model backbone.
- **layers** (*Sequence[str]*) – Layers to extract features from.
- **pre_trained** (*bool, optional*) – Boolean to check whether to use a pre_trained backbone. Defaults to `True`.
- **n_pca_components** (*int, optional*) – Number of PCA components. Defaults to `16`.
- **feature_scaling_method** (*FeatureScalingMethod, optional*) – Feature scaling method. Defaults to `FeatureScalingMethod.SCALE`.
- **max_training_points** (*int, optional*) – Number of training points to fit the KDE model. Defaults to `40000`.

`forward(batch)`

Prediction by normality model.

Parameters:

batch (*torch.Tensor*) – Input images.

Returns:

Predictions

Return type:

Tensor

`get_features(batch)`

Extract features from the pretrained network.

Parameters:

batch (*torch.Tensor*) – Image batch.

Returns:

torch.Tensor containing extracted features.

Return type:

Tensor

Previous
CS-Flow

Next
DFM