Classification

Contents

Classification modules.

class **anomalib.models.components.classification.FeatureScalingMethod(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)**

Bases: `str`, `Enum`

Determines how the feature embeddings are scaled.

class **anomalib.models.components.classification.KDEClassifier(*n_pca_components=16*, *feature_scaling_method=FeatureScalingMethod.SCALE*, *max_training_points=40000*)**

Bases: `Module`

Classification module for KDE-based anomaly detection.

Parameters:

- **n_pca_components** (*int, optional*) – Number of PCA components. Defaults to 16.
- **feature_scaling_method** (*FeatureScalingMethod, optional*) – Scaling method applied to features before passing to KDE. Options are *norm* (normalize to unit vector length) and *scale* (scale to max length observed in training).
- **max_training_points** (*int, optional*) – Maximum number of training points to fit the KDE model. Defaults to 40000.

compute_kde_scores(features, as_log_likelihood=False)

Compute the KDE scores.

The scores calculated from the KDE model are converted to densities. If *as_log_likelihood* is set to true then

the log of the scores are calculated.

Parameters:

- **features** (*torch.Tensor*) – Features to which the PCA model is fit.
- **as_log_likelihood** (*bool | None, optional*) – If true, gets log likelihood scores. Defaults to False.

Returns:

Score

Return type:

(torch.Tensor)

static compute_probabilities(scores)

Convert density scores to anomaly probabilities (see https://www.desmos.com/calculator/ifju7eesg7).

Parameters:

scores (*torch.Tensor*) – density of an image.

Return type:

`Tensor`

Returns:

probability that image with {density} is anomalous

fit(embeddings)

Fit a kde model to embeddings.

Parameters:

embeddings (*torch.Tensor*) – Input embeddings to fit the model.

Return type:

`bool`

Returns:

Boolean confirming whether the training is successful.

`forward(features)`

Make predictions on extracted features.

Return type:

`Tensor`

`pre_process(feature_stack, max_length=None)`

Pre-process the CNN features.

Parameters:

- **feature_stack** (*torch.Tensor*) – Features extracted from CNN
- **max_length** (*Tensor | None*) – Used to unit normalize the feature_stack vector. If `max_len` is not provided, the length is calculated from the `feature_stack`. Defaults to None.

Returns:

Stacked features and length

Return type:

(Tuple)

`predict(features)`

Predicts the probability that the features belong to the anomalous class.

Parameters:

features (*torch.Tensor*) – Feature from which the output probabilities are detected.

Return type:

`Tensor`

Returns:

Detection probabilities

Previous
Filters

Next
Cluster