AI VAD

Contents

- `AiVad`
- `AiVadModel`
- `DeepExtractor`
- `FeatureExtractor`
- `FeatureType`
- `PoseExtractor`
- `VelocityExtractor`
- `RegionExtractor`
- `FlowExtractor`
- `BaseDensityEstimator`
- `CombinedDensityEstimator`
- `GMMEstimator`
- `GroupedKNNEstimator`

Attribute-based Representations for Accurate and Interpretable Video Anomaly Detection.

Paper https://arxiv.org/pdf/2212.00789.pdf

class **anomalib.models.video.ai_vad.lightning_model.AiVad**(*box_score_thresh=0.7*, *persons_only=False*, *min_bbox_area=100*, *max_bbox_overlap=0.65*, *enable_foreground_detections=True*, *foreground_kernel_size=3*, *foreground_binary_threshold=18*, *n_velocity_bins=1*, *use_velocity_features=True*, *use_pose_features=True*, *use_deep_features=True*, *n_components_velocity=2*, *n_neighbors_pose=1*, *n_neighbors_deep=1*)

Bases: `MemoryBankMixin`, `AnomalyModule`

AI-VAD: Attribute-based Representations for Accurate and Interpretable Video Anomaly Detection.

Parameters:

- **box_score_thresh** (*float*) – Confidence threshold for bounding box predictions. Defaults to `0.7`.
- **persons_only** (*bool*) – When enabled, only regions labeled as person are included. Defaults to `False`.
- **min_bbox_area** (*int*) – Minimum bounding box area. Regions with a surface area lower than this value are excluded. Defaults to `100`.
- **max_bbox_overlap** (*float*) – Maximum allowed overlap between bounding boxes. Defaults to `0.65`.
- **enable_foreground_detections** (*bool*) – Add additional foreground detections based on pixel difference between consecutive frames. Defaults to `True`.
- **foreground_kernel_size** (*int*) – Gaussian kernel size used in foreground detection. Defaults to `3`.
- **foreground_binary_threshold** (*int*) – Value between 0 and 255 which acts as binary threshold in foreground detection. Defaults to `18`.
- **n_velocity_bins** (*int*) – Number of discrete bins used for velocity histogram features. Defaults to `1`.
- **use_velocity_features** (*bool*) – Flag indicating if velocity features should be used. Defaults to `True`.
- **use_pose_features** (*bool*) – Flag indicating if pose features should be used. Defaults to `True`.
- **use_deep_features** (*bool*) – Flag indicating if deep features should be used. Defaults to `True`.
- **n_components_velocity** (*int*) – Number of components used by GMM density estimation for velocity features. Defaults to `2`.
- **n_neighbors_pose** (*int*) – Number of neighbors used in KNN density estimation for pose features. Defaults to `1`.
- **n_neighbors_deep** (*int*) – Number of neighbors used in KNN density estimation for deep features. Defaults to `1`.

static `configure_optimizers()`

AI-VAD training does not involve fine-tuning of NN weights, no optimizers needed.

Return type:

`None`

`configure_transforms(image_size=None)`

AI-VAD does not need a transform, as the region- and feature-extractors apply their own transforms.

Return type:

`Transform` | `None`

`fit()`

Fit the density estimators to the extracted features 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]*

AI-VAD specific trainer arguments.

`training_step`(*batch*)

Training Step of AI-VAD.

Extract features from the batch of clips and update the density estimators.

Parameters:

batch (*dict[str, str* | *torch.Tensor]*) – Batch containing image filename, image, label and mask

Return type:

`None`

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

Perform the validation step of AI-VAD.

Extract boxes and box scores..

Parameters:

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

Return type:

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

Returns:

Batch dictionary with added boxes and box scores.

PyTorch model for AI-VAD model implementation.

Paper https://arxiv.org/pdf/2212.00789.pdf

```
class anomalib.models.video.ai_vad.torch_model.AiVadModel(box_score_thresh=0.8,
persons_only=False, min_bbox_area=100, max_bbox_overlap=0.65,
enable_foreground_detections=True, foreground_kernel_size=3,
foreground_binary_threshold=18, n_velocity_bins=8, use_velocity_features=True,
use_pose_features=True, use_deep_features=True, n_components_velocity=5,
n_neighbors_pose=1, n_neighbors_deep=1)
```

Bases: `Module`

AI-VAD model.

Parameters:

- **box_score_thresh** (*float*) – Confidence threshold for region extraction stage. Defaults to `0.8`.
- **persons_only** (*bool*) – When enabled, only regions labeled as person are included. Defaults to `False`.
- **min_bbox_area** (*int*) – Minimum bounding box area. Regions with a surface area lower than this value are excluded. Defaults to `100`.
- **max_bbox_overlap** (*float*) – Maximum allowed overlap between bounding boxes. Defaults to `0.65`.
- **enable_foreground_detections** (*bool*) – Add additional foreground detections based on pixel difference between consecutive frames. Defaults to `True`.
- **foreground_kernel_size** (*int*) – Gaussian kernel size used in foreground detection. Defaults to `3`.
- **foreground_binary_threshold** (*int*) – Value between 0 and 255 which acts as binary threshold in foreground detection. Defaults to `18`.
- **n_velocity_bins** (*int*) – Number of discrete bins used for velocity histogram features. Defaults to `8`.
- **use_velocity_features** (*bool*) – Flag indicating if velocity features should be used. Defaults to `True`.
- **use_pose_features** (*bool*) – Flag indicating if pose features should be used. Defaults to `True`.
- **use_deep_features** (*bool*) – Flag indicating if deep features should be used. Defaults to `True`.
- **n_components_velocity** (*int*) – Number of components used by GMM density estimation for velocity features. Defaults to `5`.
- **n_neighbors_pose** (*int*) – Number of neighbors used in KNN density estimation for pose features. Defaults to `1`.
- **n_neighbors_deep** (*int*) – Number of neighbors used in KNN density estimation for deep features. Defaults to `1`.

`forward(batch)`

Forward pass through AI-VAD model.

Parameters:

batch (*torch.Tensor*) – Input image of shape (N, L, C, H, W)

Returns:

List of bbox locations for each image. list[torch.Tensor]: List of per-bbox anomaly scores for each image. list[torch.Tensor]: List of per-image anomaly scores.

Return type:

list[torch.Tensor]

Feature extraction module for AI-VAD model implementation.

class `anomalib.models.video.ai_vad.features.DeepExtractor`

Bases: `Module`

Deep feature extractor.

Extracts the deep (appearance) features from the input regions.

`forward`(*batch*, *boxes*, *batch_size*)

Extract deep features using CLIP encoder.

Parameters:

- **batch** (*torch.Tensor*) – Batch of RGB input images of shape (N, 3, H, W)
- **boxes** (*torch.Tensor*) – Bounding box coordinates of shaspe (M, 5). First column indicates batch index of the bbox.
- **batch_size** (*int*) – Number of images in the batch.

Returns:

Deep feature tensor of shape (M, 512)

Return type:

Tensor

class `anomalib.models.video.ai_vad.features.FeatureExtractor`(*n_velocity_bins=8*, *use_velocity_features=True*, *use_pose_features=True*, *use_deep_features=True*)

Bases: `Module`

Feature extractor for AI-VAD.

Parameters:

- **n_velocity_bins** (*int*) – Number of discrete bins used for velocity histogram features. Defaults to `8`.
- **use_velocity_features** (*bool*) – Flag indicating if velocity features should be used. Defaults to `True`.
- **use_pose_features** (*bool*) – Flag indicating if pose features should be used. Defaults to `True`.
- **use_deep_features** (*bool*) – Flag indicating if deep features should be used. Defaults to `True`.

`forward`(*rgb_batch*, *flow_batch*, *regions*)

Forward pass through the feature extractor.

Extract any combination of velocity, pose and deep features depending on configuration.

Parameters:

- **rgb_batch** (*torch.Tensor*) – Batch of RGB images of shape (N, 3, H, W)
- **flow_batch** (*torch.Tensor*) – Batch of optical flow images of shape (N, 2, H, W)
- **regions** (*list[dict]*) – Region information per image in batch.

Returns:

Feature dictionary per image in batch.

Return type:

list[dict]

class **anomalib.models.video.ai_vad.features.FeatureType**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: `str`, `Enum`

Names of the different feature streams used in AI-VAD.

class **anomalib.models.video.ai_vad.features.PoseExtractor**(**args*, ***kwargs*)

Bases: `Module`

Pose feature extractor.

Extracts pose features based on estimated body landmark keypoints.

forward(*batch*, *boxes*)

Extract pose features using a human keypoint estimation model.

Parameters:

- **batch** (*torch.Tensor*) – Batch of RGB input images of shape (N, 3, H, W)
- **boxes** (*torch.Tensor*) – Bounding box coordinates of shaspe (M, 5). First column indicates batch index of the bbox.

Returns:

list of pose feature tensors for each image.

Return type:

list[torch.Tensor]

class **anomalib.models.video.ai_vad.features.VelocityExtractor**(*n_bins=8*)

Bases: `Module`

Velocity feature extractor.

Extracts histograms of optical flow magnitude and direction.

Parameters:

n_bins (*int*) – Number of direction bins used for the feature histograms.

`forward(flows, boxes)`

Extract velocioty features by filling a histogram.

Parameters:

- **flows** (*torch.Tensor*) – Batch of optical flow images of shape (N, 2, H, W)
- **boxes** (*torch.Tensor*) – Bounding box coordinates of shaspe (M, 5). First column indicates batch index of the bbox.

Returns:

Velocity feature tensor of shape (M, n_bins)

Return type:

Tensor

Regions extraction module of AI-VAD model implementation.

`class anomalib.models.video.ai_vad.regions.RegionExtractor(box_score_thresh=0.8, persons_only=False, min_bbox_area=100, max_bbox_overlap=0.65, enable_foreground_detections=True, foreground_kernel_size=3, foreground_binary_threshold=18)`

Bases: `Module`

Region extractor for AI-VAD.

Parameters:

- **box_score_thresh** (*float*) – Confidence threshold for bounding box predictions. Defaults to `0.8`.
- **persons_only** (*bool*) – When enabled, only regions labeled as person are included. Defaults to `False`.
- **min_bbox_area** (*int*) – Minimum bounding box area. Regions with a surface area lower than this value are excluded. Defaults to `100`.
- **max_bbox_overlap** (*float*) – Maximum allowed overlap between bounding boxes. Defaults to `0.65`.
- **enable_foreground_detections** (*bool*) – Add additional foreground detections based on pixel difference between consecutive frames. Defaults to `True`.
- **foreground_kernel_size** (*int*) – Gaussian kernel size used in foreground detection. Defaults to `3`.
- **foreground_binary_threshold** (*int*) – Value between 0 and 255 which acts as binary threshold in foreground detection. Defaults to `18`.

`add_foreground_boxes`(*regions*, *first_frame*, *last_frame*, *kernel_size*, *binary_threshold*)

Add any foreground regions that were not detected by the region extractor.

This method adds regions that likely belong to the foreground of the video scene, but were not detected by the region extractor module. The foreground pixels are determined by taking the pixel difference between two consecutive video frames and applying a binary threshold. The final detections consist of all connected components in the foreground that do not fall in one of the bounding boxes predicted by the region extractor.

Parameters:

- **regions** (*list[dict[str, torch.Tensor]]*) – Region detections for a batch of images, generated by the region extraction module.
- **first_frame** (*torch.Tensor*) – video frame at time t-1
- **last_frame** (*torch.Tensor*) – Video frame time t
- **kernel_size** (*int*) – Kernel size for Gaussian smoothing applied to input frames
- **binary_threshold** (*int*) – Binary threshold used in foreground detection, should be in range [0, 255]

Returns:

region detections with foreground regions appended

Return type:

list[dict[str, torch.Tensor]]

`forward(first_frame, last_frame)`

Perform forward-pass through region extractor.

Parameters:

- **first_frame** (*torch.Tensor*) – Batch of input images of shape (N, C, H, W) forming the first frames in the clip.
- **last_frame** (*torch.Tensor*) – Batch of input images of shape (N, C, H, W) forming the last frame in the clip.

Returns:

List of Mask RCNN predictions for each image in the batch.

Return type:

list[dict]

`post_process_bbox_detections(regions)`

Post-process the region detections.

The region detections are filtered based on class label, bbox area and overlap with other regions.

Parameters:

regions (*list[dict[str, torch.Tensor]]*) – Region detections for a batch of images, generated by the region extraction module.

Returns:

Filtered regions

Return type:

list[dict[str, torch.Tensor]]

`static subsample_regions(regions, indices)`

Subsample the items in a region dictionary based on a Tensor of indices.

Parameters:

- **regions** (*dict[str, torch.Tensor]*) – Region detections for a single image in the batch.
- **indices** (*torch.Tensor*) – Indices of region detections that should be kept.

Returns:

Subsampled region detections.

Return type:

dict[str, torch.Tensor]

Optical Flow extraction module for AI-VAD implementation.

class `anomalib.models.video.ai_vad.flow.FlowExtractor`(**args*, ***kwargs*)

Bases: `Module`

Optical Flow extractor.

Computes the pixel displacement between 2 consecutive frames from a video clip.

`forward`(*first_frame*, *last_frame*)

Forward pass through the flow extractor.

Parameters:

- **first_frame** (*torch.Tensor*) – Batch of starting frames of shape (N, 3, H, W).
- **last_frame** (*torch.Tensor*) – Batch of last frames of shape (N, 3, H, W).

Returns:

Estimated optical flow map of shape (N, 2, H, W).

Return type:

Tensor

`pre_process`(*first_frame*, *last_frame*)

Resize inputs to dimensions required by backbone.

Parameters:

- **first_frame** (*torch.Tensor*) – Starting frame of optical flow computation.
- **last_frame** (*torch.Tensor*) – Last frame of optical flow computation.

Returns:

Preprocessed first and last frame.

Return type:

tuple[torch.Tensor, torch.Tensor]

Density estimation module for AI-VAD model implementation.

class `anomalib.models.video.ai_vad.density.BaseDensityEstimator`(**args*, ***kwargs*)

Bases: `Module`, `ABC`

Base density estimator.

abstract `fit`()

Compose model using collected features.

Return type:

`None`

`forward(features)`

Update or predict depending on training status.

Return type:

`Tensor` | `tuple`[`Tensor`, `Tensor`] | `None`

abstract `predict(features)`

Predict the density of a set of features.

Return type:

`Tensor` | `tuple`[`Tensor`, `Tensor`]

abstract `update(features, group=None)`

Update the density model with a new set of features.

Return type:

`None`

class `anomalib.models.video.ai_vad.density.CombinedDensityEstimator(use_pose_features=True, use_deep_features=True, use_velocity_features=False, n_neighbors_pose=1, n_neighbors_deep=1, n_components_velocity=5)`

Bases: `BaseDensityEstimator`

Density estimator for AI-VAD.

Combines density estimators for the different feature types included in the model.

Parameters:

- **use_pose_features** (*bool*) – Flag indicating if pose features should be used. Defaults to `True`.
- **use_deep_features** (*bool*) – Flag indicating if deep features should be used. Defaults to `True`.
- **use_velocity_features** (*bool*) – Flag indicating if velocity features should be used. Defaults to `False`.
- **n_neighbors_pose** (*int*) – Number of neighbors used in KNN density estimation for pose features. Defaults to `1`.
- **n_neighbors_deep** (*int*) – Number of neighbors used in KNN density estimation for deep features. Defaults to `1`.
- **n_components_velocity** (*int*) – Number of components used by GMM density estimation for velocity features. Defaults to `5`.

`fit()`

Fit the density estimation models on the collected features.

Return type:

`None`

`predict(features)`

Predict the region- and image-level anomaly scores for an image based on a set of features.

Parameters:

features (*dict[Tensor]*) – Dictionary containing extracted features for a single frame.

Returns:

Region-level anomaly scores for all regions withing the frame. Tensor: Frame-level anomaly score for the frame.

Return type:

Tensor

`update(features, group=None)`

Update the density estimators for the different feature types.

Parameters:

- **features** (*dict[FeatureType, torch.Tensor]*) – Dictionary containing extracted features for a single frame.
- **group** (*str*) – Identifier of the video from which the frame was sampled. Used for grouped density estimation.

Return type:

`None`

class `anomalib.models.video.ai_vad.density.GMMEstimator(n_components=2)`

Bases: `BaseDensityEstimator`

Density estimation based on Gaussian Mixture Model.

Parameters:

n_components (*int*) – Number of components used in the GMM. Defaults to `2`.

`fit()`

Fit the GMM and compute normalization statistics.

Return type:

`None`

`predict(features, normalize=True)`

Predict the density of a set of feature vectors.

Parameters:

- **features** (*torch.Tensor*) – Input feature vectors.
- **normalize** (*bool*) – Flag indicating if the density should be normalized to min-max stats of the feature bank. Defaults to `True`.

Returns:

Density scores of the input feature vectors.

Return type:

Tensor

`update(features, group=None)`

Update the feature bank.

Return type:

`None`

class `anomalib.models.video.ai_vad.density.GroupedKNNEstimator(n_neighbors)`

Bases: `DynamicBufferMixin`, `BaseDensityEstimator`

Grouped KNN density estimator.

Keeps track of the group (e.g. video id) from which the features were sampled for normalization purposes.

Parameters:
n_neighbors (*int*) – Number of neighbors used in KNN search.

`fit()`

Fit the KNN model by stacking the feature vectors and computing the normalization statistics.

Return type:
`None`

`predict(features, group=None, n_neighbors=1, normalize=True)`

Predict the (normalized) density for a set of features.

Parameters:

- **features** (*torch.Tensor*) – Input features that will be compared to the density model.
- **group** (*str, optional*) – Group (video id) from which the features originate. If passed, all features of the same group in the memory bank will be excluded from the density estimation. Defaults to `None`.
- **n_neighbors** (*int*) – Number of neighbors used in the KNN search. Defaults to `1`.
- **normalize** (*bool*) – Flag indicating if the density should be normalized to min-max stats of the feature bank. Defatuls to `True`.

Returns:
Mean (normalized) distances of input feature vectors to k nearest neighbors in feature bank.

Return type:
Tensor

`update(features, group=None)`

Update the internal feature bank while keeping track of the group.

Parameters:

- **features** (*torch.Tensor*) – Feature vectors extracted from a video frame.
- **group** (*str*) – Identifier of the group (video) from which the frame was sampled.

Return type:
`None`

Previous
Video Models

Next
Engine