DSR

Contents

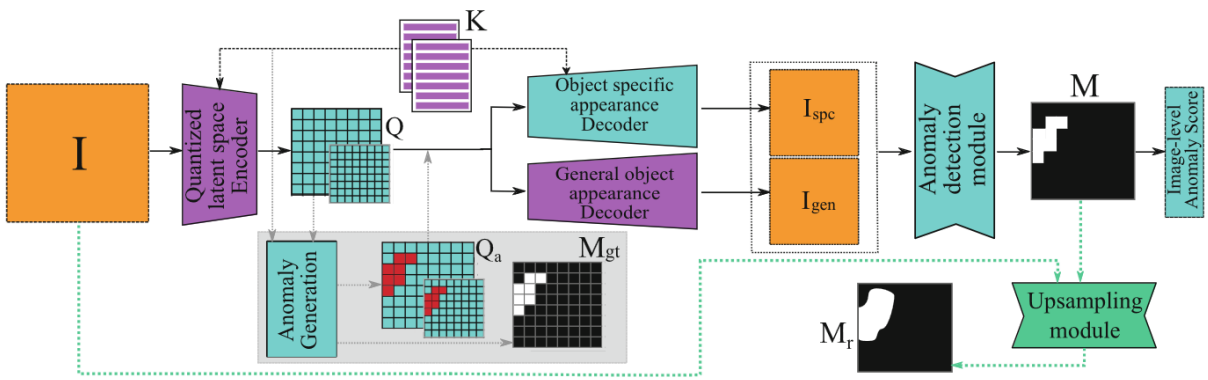
This is the implementation of the DSR paper.

Model Type: Segmentation

Description

DSR is a quantized-feature based algorithm that consists of an autoencoder with one encoder and two decoders, coupled with an anomaly detection module. DSR learns a codebook of quantized representations on ImageNet, which are then used to encode input images. These quantized representations also serve to sample near-in-distribution anomalies, since they do not rely on external datasets. Training takes place in three phases. The encoder and "general object decoder", as well as the codebook, are pretrained on ImageNet. Defects are then generated at the feature level using the codebook on the quantized representations, and are used to train the object-specific decoder as well as the anomaly detection module. In the final phase of training, the upsampling module is trained on simulated image-level smudges in order to output more robust anomaly maps.

Architecture



PyTorch model for the DSR model implementation.

class anomalib.models.image.dsr.torch_model.**AnomalyDetectionModule**(*in_channels*, *out_channels*, *base_width*)

Bases: `Module`

Anomaly detection module.

Module that detects the preseßnce of an anomaly by comparing two images reconstructed by the object specific decoder and the general object decoder.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **out_channels** (*int*) – Number of output channels.
- **base_width** (*int*) – Base dimensionality of the layers of the autoencoder.

forward(*batch_real*, *batch_anomaly*)

Computes the anomaly map over corresponding real and anomalous images.

Parameters:

- **batch_real** (*torch.Tensor*) – Batch of real, non defective images.
- **batch_anomaly** (*torch.Tensor*) – Batch of potentially anomalous images.

Return type:

`Tensor`

Returns:

The anomaly segmentation map.

class anomalib.models.image.dsr.torch_model.**DecoderBot**(*in_channels*, *num_hiddens*, *num_residual_layers*, *num_residual_hiddens*)

Bases: `Module`

General appearance decoder module to reconstruct images while keeping possible anomalies.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **num_hiddens** (*int*) – Number of hidden channels.
- **num_residual_layers** (*int*) – Number of residual layers in residual stack.
- **num_residual_hiddens** (*int*) – Number of channels in residual layers.

forward(*inputs*)

Decode quantized feature maps into an image.

Parameters:

inputs (*torch.Tensor*) – Quantized feature maps.

Return type:

`Tensor`

Returns:

Decoded image.

class anomalib.models.image.dsr.torch_model.DiscreteLatentModel(*num_hiddens*, *num_residual_layers*, *num_residual_hiddens*, *num_embeddings*, *embedding_dim*)

Bases: `Module`

Discrete Latent Model.

Autoencoder quantized model that encodes the input images into quantized feature maps and generates a reconstructed image using the general appearance decoder.

Parameters:

- **num_hiddens** (*int*) – Number of hidden channels.
- **num_residual_layers** (*int*) – Number of residual layers in residual stacks.
- **num_residual_hiddens** (*int*) – Number of channels in residual layers.
- **num_embeddings** (*int*) – Size of embedding dictionary.
- **embedding_dim** (*int*) – Dimension of embeddings.

forward(*batch*, *anomaly_mask=None*, *anom_str_lo=None*, *anom_str_hi=None*)

Generate quantized feature maps.

Generates quantized feature maps of batch of input images as well as their reconstruction based on the general appearance decoder.

Parameters:

- **batch** (*Tensor*) – Batch of input images.
- **anomaly_mask** (*Tensor* | *None*) – Anomaly mask to be used to generate anomalies on the quantized feature maps.
- **anom_str_lo** (*torch.Tensor* | *None*) – Strength of generated anomaly lo.
- **anom_str_hi** (*torch.Tensor* | *None*) – Strength of generated anomaly hi.

Returns:

If generating an anomaly mask:

- General object decoder-decoded anomalous image
- Reshaped ground truth anomaly map
- Non defective quantized lo feature
- Non defective quantized hi feature
- Non quantized subspace encoded defective lo feature
- Non quantized subspace encoded defective hi feature

Else:

- General object decoder-decoded image
- Quantized lo feature
- Quantized hi feature

Return type:

dict[str, torch.Tensor]

`generate_fake_anomalies_joined(features, embeddings, memory_torch_original, mask, strength)`

Generate quantized anomalies.

Parameters:

- **features** (*torch.Tensor*) – Features on which the anomalies will be generated.
- **embeddings** (*torch.Tensor*) – Embeddings to use to generate the anomalies.
- **memory_torch_original** (*torch.Tensor*) – Weight of embeddings.
- **mask** (*torch.Tensor*) – Original anomaly mask.
- **strength** (*float*) – Strength of generated anomaly.

Returns:

Anomalous embedding.

Return type:

torch.Tensor

property **vq_vae_bot**: *VectorQuantizer*

Return `self._vq_vae_bot`.

property **vq_vae_top**: *VectorQuantizer*

Return `self._vq_vae_top`.

class **anomalib.models.image.dsr.torch_model.DsrModel**(*latent_anomaly_strength=0.2*, *embedding_dim=128*, *num_embeddings=4096*, *num_hiddens=128*, *num_residual_layers=2*, *num_residual_hiddens=64*)

Bases: `Module`

DSR PyTorch model.

Consists of the discrete latent model, image reconstruction network, subspace restriction modules, anomaly detection module and upsampling module.

Parameters:

- **embedding_dim** (*int*) – Dimension of codebook embeddings.
- **num_embeddings** (*int*) – Number of embeddings.
- **latent_anomaly_strength** (*float*) – Strength of the generated anomalies in the latent space.
- **num_hiddens** (*int*) – Number of output channels in residual layers.
- **num_residual_layers** (*int*) – Number of residual layers.
- **num_residual_hiddens** (*int*) – Number of intermediate channels.

forward(*batch*, *anomaly_map_to_generate=None*)

Compute the anomaly mask from an input image.

Parameters:

- **batch** (*torch.Tensor*) – Batch of input images.
- **anomaly_map_to_generate** (*torch.Tensor* | *None*) – anomaly map to use to generate quantized defects.
- **2** (*If not training phase*) –
- **None.** (*should be*) –

Returns:

If testing:

- "anomaly_map": Upsampled anomaly map
- "pred_score": Image score

If training phase 2:

- "recon_feat_hi": Reconstructed non-quantized hi features of defect (F~_hi)
- "recon_feat_lo": Reconstructed non-quantized lo features of defect (F~_lo)
- "embedding_bot": Quantized features of non defective img (Q_hi)
- "embedding_top": Quantized features of non defective img (Q_lo)
- "obj_spec_image": Object-specific-decoded image (I_spc)
- "anomaly_map": Predicted segmentation mask (M)
- "true_mask": Resized ground-truth anomaly map (M_gt)

If training phase 3:

- "anomaly_map": Reconstructed anomaly map

Return type:

dict[str, torch.Tensor]

load_pretrained_discrete_model_weights(*ckpt*)

Load pre-trained model weights.

Return type:

`None`

class anomalib.models.image.dsr.torch_model.EncoderBot(*in_channels*, *num_hiddens*, *num_residual_layers*, *num_residual_hiddens*)

Bases: `Module`

Encoder module for bottom quantized feature maps.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **num_hiddens** (*int*) – Number of hidden channels.
- **num_residual_layers** (*int*) – Number of residual layers in residual stacks.
- **num_residual_hiddens** (*int*) – Number of channels in residual layers.

forward(*batch*)

Encode inputs to be quantized into the bottom feature map.

Parameters:

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

Return type:

`Tensor`

Returns:

Encoded feature maps.

class anomalib.models.image.dsr.torch_model.EncoderTop(*in_channels*, *num_hiddens*, *num_residual_layers*, *num_residual_hiddens*)

Bases: `Module`

Encoder module for top quantized feature maps.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **num_hiddens** (*int*) – Number of hidden channels.
- **num_residual_layers** (*int*) – Number of residual layers in residual stacks.
- **num_residual_hiddens** (*int*) – Number of channels in residual layers.

forward(*batch*)

Encode inputs to be quantized into the top feature map.

Parameters:

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

Return type:

`Tensor`

Returns:

Encoded feature maps.

class anomalib.models.image.dsr.torch_model.FeatureDecoder(*base_width*, *out_channels=1*)

Bases: `Module`

Feature decoder for the subspace restriction network.

Parameters:

- **base_width** (*int*) – Base dimensionality of the layers of the autoencoder.
- **out_channels** (*int*) – Number of output channels.

`forward(_, __, b3)`

Decode a batch of latent features to a non-quantized representation.

Parameters:

- _ (*torch.Tensor*) – Top latent feature layer.
- __ (*torch.Tensor*) – Middle latent feature layer.
- **b3** (*torch.Tensor*) – Bottom latent feature layer.

Return type:

`Tensor`

Returns:

Decoded non-quantized representation.

class anomalib.models.image.dsr.torch_model.FeatureEncoder(*in_channels*, *base_width*)

Bases: `Module`

Feature encoder for the subspace restriction network.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **base_width** (*int*) – Base dimensionality of the layers of the autoencoder.

`forward(batch)`

Encode a batch of input features to the latent space.

Parameters:

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

Return type:

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

Returns: Encoded feature maps.

class anomalib.models.image.dsr.torch_model.**ImageReconstructionNetwork**(*in_channels*, *num_hiddens*, *num_residual_layers*, *num_residual_hiddens*)

Bases: `Module`

Image Reconstruction Network.

Image reconstruction network that reconstructs the image from a quantized representation.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **num_hiddens** (*int*) – Number of output channels in residual layers.
- **num_residual_layers** (*int*) – Number of residual layers.
- **num_residual_hiddens** (*int*) – Number of intermediate channels.

forward(*inputs*)

Reconstructs an image from a quantized representation.

Parameters:

inputs (*torch.Tensor*) – Quantized features.

Return type:

`Tensor`

Returns:

Reconstructed image.

class anomalib.models.image.dsr.torch_model.**Residual**(*in_channels*, *out_channels*, *num_residual_hiddens*)

Bases: `Module`

Residual layer.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **out_channels** (*int*) – Number of output channels.
- **num_residual_hiddens** (*int*) – Number of intermediate channels.

forward(*batch*)

Compute residual layer.

Parameters:

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

Return type:

`Tensor`

Returns:

Computed feature maps.

class anomalib.models.image.dsr.torch_model.ResidualStack(*in_channels*, *num_hiddens*, *num_residual_layers*, *num_residual_hiddens*)

Bases: `Module`

Stack of residual layers.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **num_hiddens** (*int*) – Number of output channels in residual layers.
- **num_residual_layers** (*int*) – Number of residual layers.
- **num_residual_hiddens** (*int*) – Number of intermediate channels.

forward(*batch*)

Compute residual stack.

Parameters:

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

Return type:

`Tensor`

Returns:

Computed feature maps.

class anomalib.models.image.dsr.torch_model.SubspaceRestrictionModule(*base_width*)

Bases: `Module`

Subspace Restriction Module.

Subspace restriction module that restricts the appearance subspace into configurations that agree with normal appearances and applies quantization.

Parameters:

base_width (*int*) – Base dimensionality of the layers of the autoencoder.

forward(*batch*, *quantization*)

Generate the quantized anomaly-free representation of an anomalous image.

Parameters:

- **batch** (*torch.Tensor*) – Batch of input images.
- **quantization** (*function* | *object*) – Quantization function.

Return type:

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

Returns:

Reconstructed batch of non-quantized features and corresponding quantized features.

class anomalib.models.image.dsr.torch_model.SubspaceRestrictionNetwork(*in_channels=64*, *out_channels=64*, *base_width=64*)

Bases: `Module`

Subspace Restriction Network.

Subspace restriction network that reconstructs the input image into a non-quantized configuration that agrees with normal appearances.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **out_channels** (*int*) – Number of output channels.
- **base_width** (*int*) – Base dimensionality of the layers of the autoencoder.

forward(*batch*)

Reconstruct non-quantized representation from batch.

Generate non-quantized feature maps from potentially anomalous images, to be quantized into non-anomalous quantized representations.

Parameters:

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

Return type:

`Tensor`

Returns:

Reconstructed non-quantized representation.

class anomalib.models.image.dsr.torch_model.UnetDecoder(*base_width*, *out_channels=1*)

Bases: `Module`

Decoder of the Unet network.

Parameters:

- **base_width** (*int*) – Base dimensionality of the layers of the autoencoder.
- **out_channels** (*int*) – Number of output channels.

forward(*b1*, *b2*, *b3*, *b4*)

Decodes latent represnetations into an image.

Parameters:

- **b1** (*torch.Tensor*) – First (top level) quantized feature map.
- **b2** (*torch.Tensor*) – Second quantized feature map.
- **b3** (*torch.Tensor*) – Third quantized feature map.
- **b4** (*torch.Tensor*) – Fourth (bottom level) quantized feature map.

Return type:

`Tensor`

Returns:

Reconstructed image.

class anomalib.models.image.dsr.torch_model.UnetEncoder(*in_channels*, *base_width*)

Bases: `Module`

Encoder of the Unet network.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **base_width** (*int*) – Base dimensionality of the layers of the autoencoder.

forward(*batch*)

Encodes batch of images into a latent representation.

Parameters:

batch (*torch.Tensor*) – Quantized features.

Return type:

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

Returns:

Latent representations of the input batch.

class anomalib.models.image.dsr.torch_model.UnetModel(*in_channels=64*, *out_channels=64*, *base_width=64*)

Bases: `Module`

Autoencoder model that reconstructs the input image.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **out_channels** (*int*) – Number of output channels.
- **base_width** (*int*) – Base dimensionality of the layers of the autoencoder.

forward(*batch*)

Reconstructs an input batch of images.

Parameters:

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

Return type:

`Tensor`

Returns:

Reconstructed images.

class anomalib.models.image.dsr.torch_model.UpsamplingModule(*in_channels=8*,

out_channels=2, base_width=64)

Bases: `Module`

Module that upsamples the generated anomaly mask to full resolution.

Parameters:

- **in_channels** (*int*) – Number of input channels.
- **out_channels** (*int*) – Number of output channels.
- **base_width** (*int*) – Base dimensionality of the layers of the autoencoder.

forward(*batch_real, batch_anomaly, batch_segmentation_map*)

Computes upsampled segmentation maps.

Parameters:

- **batch_real** (*torch.Tensor*) – Batch of real, non defective images.
- **batch_anomaly** (*torch.Tensor*) – Batch of potentially anomalous images.
- **batch_segmentation_map** (*torch.Tensor*) – Batch of anomaly segmentation maps.

Return type:

`Tensor`

Returns:

Upsampled anomaly segmentation maps.

class anomalib.models.image.dsr.torch_model.**VectorQuantizer**(*num_embeddings, embedding_dim*)

Bases: `Module`

Module that quantizes a given feature map using learned quantization codebooks.

Parameters:

- **num_embeddings** (*int*) – Size of embedding codebook.
- **embedding_dim** (*int*) – Dimension of embeddings.

property **embedding**: *Tensor*

Return embedding.

forward(*inputs*)

Calculates quantized feature map.

Parameters:

inputs (*torch.Tensor*) – Non-quantized feature maps.

Return type:

`Tensor`

Returns:

Quantized feature maps.

DSR - A Dual Subspace Re-Projection Network for Surface Anomaly Detection.

Paper https://link.springer.com/chapter/10.1007/978-3-031-19821-2_31

class **anomalib.models.image.dsr.lightning_model.Dsr(*latent_anomaly_strength=0.2*, *upsampling_train_ratio=0.7*)**

Bases: `AnomalyModule`

DSR: A Dual Subspace Re-Projection Network for Surface Anomaly Detection.

Parameters:

- **latent_anomaly_strength** (*float*) – Strength of the generated anomalies in the latent space. Defaults to 0.2
- **upsampling_train_ratio** (*float*) – Ratio of training steps for the upsampling module. Defaults to 0.7

configure_optimizers()

Configure the Adam optimizer for training phases 2 and 3.

Does not train the discrete model (phase 1)

Returns:

Dictionary of optimizers

Return type:

dict[str, torch.optim.Optimizer | torch.optim.lr_scheduler.LRScheduler]

property **learning_type: *LearningType***

Return the learning type of the model.

Returns:

Learning type of the model.

Return type:

LearningType

on_train_epoch_start()

Display a message when starting to train the upsampling module.

Return type:

`None`

on_train_start()

Load pretrained weights of the discrete model when starting training.

Return type:

`None`

prepare_pretrained_model()

Download pre-trained models if they don't exist.

Return type:

`Path`

property trainer_arguments: *dict[str, Any]*

Required trainer arguments.

training_step(*batch*)

Training Step of DSR.

Feeds the original image and the simulated anomaly mask during first phase. During second phase, feeds a generated anomalous image to train the upsampling module.

Parameters:

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

Returns:

Loss dictionary

Return type:

STEP_OUTPUT

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

Validation step of DSR.

The Softmax predictions of the anomalous class are used as anomaly map.

Parameters:

- **batch** (*dict[str, str | Tensor]*) – Batch of input images
- ***args** – unused
- ****kwargs** – unused

Returns:

Dictionary to which predicted anomaly maps have been added.

Return type:

STEP_OUTPUT

Anomaly generator for the DSR model implementation.

class

anomalib.models.image.dsr.anomaly_generator.DsrAnomalyGenerator(*p_anomalous=0.5*)

Bases: `Module`

Anomaly generator of the DSR model.

The anomaly is generated using a Perlin noise generator on the two quantized representations of an image. This generator is only used during the second phase of training! The third phase requires generating smudges over the input images.

Parameters:

p_anomalous (*float, optional*) – Probability to generate an anomalous image.

augment_batch(*batch*)

Generate anomalous augmentations for a batch of input images.

Parameters:

batch (*Tensor*) – Batch of input images

Returns:

Ground truth masks corresponding to the anomalous perturbations.

Return type:

Tensor

generate_anomaly(*height*, *width*)

Generate an anomalous mask.

Parameters:

- **height** (*int*) – Height of generated mask.
- **width** (*int*) – Width of generated mask.

Returns:

Generated mask.

Return type:

Tensor

Loss function for the DSR model implementation.

class anomalib.models.image.dsr.loss.DsrSecondStageLoss

Bases: `Module`

Overall loss function of the second training phase of the DSR model.

The total loss consists of:

- MSE loss between non-anomalous quantized input image and anomalous subspace-reconstructed non-quantized input (hi and lo)
- MSE loss between input image and reconstructed image through object-specific decoder,
- Focal loss between computed segmentation mask and ground truth mask.

forward(*recon_nq_hi*, *recon_nq_lo*, *qu_hi*, *qu_lo*, *input_image*, *gen_img*, *seg*, *anomaly_mask*)

Compute the loss over a batch for the DSR model.

Parameters:

- **recon_nq_hi** (*Tensor*) – Reconstructed non-quantized hi feature
- **recon_nq_lo** (*Tensor*) – Reconstructed non-quantized lo feature
- **qu_hi** (*Tensor*) – Non-defective quantized hi feature
- **qu_lo** (*Tensor*) – Non-defective quantized lo feature
- **input_image** (*Tensor*) – Original image
- **gen_img** (*Tensor*) – Object-specific decoded image
- **seg** (*Tensor*) – Computed anomaly map
- **anomaly_mask** (*Tensor*) – Ground truth anomaly map

Returns:

Total loss

Return type:

Tensor

class anomalib.models.image.dsr.loss.DsrThirdStageLoss

Bases: `Module`

Overall loss function of the third training phase of the DSR model.

The loss consists of a focal loss between the computed segmentation mask and the ground truth mask.

forward(*pred_mask*, *true_mask*)

Compute the loss over a batch for the DSR model.

Parameters:

- **pred_mask** (*Tensor*) – Computed anomaly map
- **true_mask** (*Tensor*) – Ground truth anomaly map

Returns:

Total loss

Return type:

Tensor

Previous DRAEM

Next Efficient AD