Efficient AD

Contents

EfficientAd: Accurate Visual Anomaly Detection at Millisecond-Level Latencies.

https://arxiv.org/pdf/2303.14535.pdf.

***class* anomalib.models.image.efficient_ad.lightning_model.EfficientAd**(*imagenet_dir='./datasets/imagenette'*, *teacher_out_channels=384*, *model_size=EfficientAdModelSize.S*, *lr=0.0001*, *weight_decay=1e-05*, *padding=False*, *pad_maps=True*, *batch_size=1*)

Bases: `AnomalyModule`

PL Lightning Module for the EfficientAd algorithm.

Parameters:

- **imagenet_dir** (*Path|str*) – directory path for the Imagenet dataset Defaults to `./datasets/imagenette`.
- **teacher_out_channels** (*int*) – number of convolution output channels Defaults to `384`.
- **model_size** (*str*) – size of student and teacher model Defaults to `EfficientAdModelSize.S`.
- **lr** (*float*) – learning rate Defaults to `0.0001`.
- **weight_decay** (*float*) – optimizer weight decay Defaults to `0.00001`.
- **padding** (*bool*) – use padding in convoluional layers Defaults to `False`.
- **pad_maps** (*bool*) – relevant if padding is set to False. In this case, pad_maps = True pads the output anomaly maps so that their size matches the size in the padding = True case. Defaults to `True`.
- **batch_size** (*int*) – batch size for imagenet dataloader Defaults to `1`.

configure_optimizers()

Configure optimizers.

Return type:

`Optimizer`

configure_transforms(*image_size=None*)

Default transform for Padim.

Return type:

`Transform`

property* `learning_type`*: LearningType

Return the learning type of the model.

Returns:

Learning type of the model.

Return type:

LearningType

`map_norm_quantiles`(*dataloader*)

Calculate 90% and 99.5% quantiles of the student(st) and autoencoder(ae).

Parameters:

dataloader (*DataLoader*) – Dataloader of the respective dataset.

Returns:

Dictionary of both the 90% and 99.5% quantiles of both the student and autoencoder feature maps.

Return type:

dict[str, torch.Tensor]

`on_train_start`()

Called before the first training epoch.

First sets up the pretrained teacher model, then prepares the imagenette data, and finally calculates or loads the channel-wise mean and std of the training dataset and push to the model.

Return type:

`None`

`on_validation_start`()

Calculate the feature map quantiles of the validation dataset and push to the model.

Return type:

`None`

`prepare_imagenette_data`(*image_size*)

Prepare ImageNette dataset transformations.

Parameters:

image_size (*tuple[int, int]* | *torch.Size*) – Image size.

Return type:

`None`

`prepare_pretrained_model`()

Prepare the pretrained teacher model.

Return type:

`None`

`teacher_channel_mean_std(dataloader)`

Calculate the mean and std of the teacher models activations.

Adapted from https://math.stackexchange.com/a/2148949

Parameters:

dataloader (*DataLoader*) – Dataloader of the respective dataset.

Returns:

Dictionary of channel-wise mean and std

Return type:

dict[str, torch.Tensor]

property `trainer_arguments`*: dict[str, Any]*

Return EfficientAD trainer arguments.

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

Perform the training step for EfficientAd returns the student, autoencoder and combined loss.

Parameters:

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

Return type:

`dict`[`str`, `Tensor`]

Returns:

Loss.

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

Perform the validation step of EfficientAd returns anomaly maps for the input image batch.

Parameters:

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

Return type:

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

Returns:

Dictionary containing anomaly maps.

Torch model for student, teacher and autoencoder model in EfficientAd.

class `anomalib.models.image.efficient_ad.torch_model.`**`EfficientAdModel`**`(`*`teacher_out_channels`*`, `*`model_size=EfficientAdModelSize.S`*`, `*`padding=False`*`, `*`pad_maps=True`*`)`

Bases: `Module`

EfficientAd model.

Parameters:

- **teacher_out_channels** (*int*) – number of convolution output channels of the pre-trained teacher model
- **model_size** (*str*) – size of student and teacher model
- **padding** (*bool*) – use padding in convoluional layers Defaults to `False`.
- **pad_maps** (*bool*) – relevant if padding is set to False. In this case, pad_maps = True pads the output anomaly maps so that their size matches the size in the padding = True case. Defaults to `True`.

`choose_random_aug_image``(`*`image`*`)`

Choose a random augmentation function and apply it to the input image.

Parameters:

image (*torch.Tensor*) – Input image.

Returns:

Augmented image.

Return type:

Tensor

`forward``(`*`batch`*`, `*`batch_imagenet=None`*`, `*`normalize=True`*`)`

Perform the forward-pass of the EfficientAd models.

Parameters:

- **batch** (*torch.Tensor*) – Input images.
- **batch_imagenet** (*torch.Tensor*) – ImageNet batch. Defaults to None.
- **normalize** (*bool*) – Normalize anomaly maps or not

Returns:

Predictions

Return type:

Tensor

`is_set``(`*`p_dic`*`)`

Check if any of the parameters in the parameter dictionary is set.

Parameters:

p_dic (*nn.ParameterDict*) – Parameter dictionary.

Returns:

Boolean indicating whether any of the parameters in the parameter dictionary is set.

Return type:

bool

Previous
DSR

Next
FastFlow