# CS-Flow

## Contents

Fully Convolutional Cross-Scale-Flows for Image-based Defect Detection.

https://arxiv.org/pdf/2110.02855.pdf

*class* **anomalib.models.image.csflow.lightning_model.Csflow**(*cross_conv_hidden_channels=1024*, *n_coupling_blocks=4*, *clamp=3*, *num_channels=3*)

Bases: `AnomalyModule`

Fully Convolutional Cross-Scale-Flows for Image-based Defect Detection.

**Parameters:**

- **n_coupling_blocks** (*int*) – Number of coupling blocks in the model. Defaults to `4`.
- **cross_conv_hidden_channels** (*int*) – Number of hidden channels in the cross convolution. Defaults to `1024`.
- **clamp** (*int*) – Clamp value for glow layer. Defaults to `3`.
- **num_channels** (*int*) – Number of channels in the model. Defaults to `3`.

**configure_optimizers**()

Configure optimizers.

**Returns:**

Adam optimizer

**Return type:**

Optimizer

### *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]*

CS-Flow-specific trainer arguments.

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

Perform the training step of CS-Flow.

**Parameters:**

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

**Return type:**

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

**Returns:**

Loss value

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

Perform the validation step for CS Flow.

**Parameters:**

- **batch** (*torch.Tensor*) – Input batch
- **args** – Arguments.
- **kwargs** – Keyword arguments.

**Returns:**

Dictionary containing the anomaly map, scores, etc.

**Return type:**

dict[str, torch.Tensor]

PyTorch model for CS-Flow implementation.

*class* `anomalib.models.image.csflow.torch_model.`**`CsFlowModel`**`(`*`input_size`*`, `*`cross_conv_hidden_channels`*`, `*`n_coupling_blocks`*`=4, `*`clamp`*`=3, `*`num_channels`*`=3)`

Bases: `Module`

CS Flow Module.

**Parameters:**

- **input_size** (*tuple[int, int]*) – Input image size.
- **cross_conv_hidden_channels** (*int*) – Number of hidden channels in the cross convolution.
- **n_coupling_blocks** (*int*) – Number of coupling blocks. Defaults to `4`.
- **clamp** (*float*) – Clamp value for the coupling blocks. Defaults to `3`.
- **num_channels** (*int*) – Number of channels in the input image. Defaults to `3`.

**`forward`**`(`*`images`*`)`

Forward method of the model.

**Parameters:**

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

**Returns:**

**During training: tuple containing the z_distribution for three scales**

and the sum of log determinant of the Jacobian. During evaluation: tuple containing anomaly maps and anomaly scores

**Return type:**

tuple[torch.Tensor, torch.Tensor]

Loss function for the CS-Flow Model Implementation.

*class* `anomalib.models.image.csflow.loss.`**`CsFlowLoss`**`(*`*`args`*`, **`*`kwargs`*`)`

Bases: `Module`

Loss function for the CS-Flow Model Implementation.

**`forward`**`(`*`z_dist`*`, `*`jacobians`*`)`

Compute the loss CS-Flow.

**Parameters:**

- **z_dist** (*torch.Tensor*) – Latent space image mappings from NF.
- **jacobians** (*torch.Tensor*) – Jacobians of the distribution

**Return type:**

`Tensor`

**Returns:**

Loss value

Anomaly Map Generator for CS-Flow model.

*class* **anomalib.models.image.csflow.anomaly_map.AnomalyMapGenerator**(*input_dims*, *mode=AnomalyMapMode.ALL*)

Bases: `Module`

Anomaly Map Generator for CS-Flow model.

**Parameters:**

- **input_dims** (*tuple[int, int, int]*) – Input dimensions.
- **mode** (*AnomalyMapMode*) – Anomaly map mode. Defaults to `AnomalyMapMode.ALL`.

**forward**(*inputs*)

Get anomaly maps by taking mean of the z-distributions across channels.

By default it computes anomaly maps for all the scales as it gave better performance on initial tests. Use `AnomalyMapMode.MAX` for the largest scale as mentioned in the paper.

**Parameters:**

- **inputs** (*torch.Tensor*) – z-distributions for the three scales.
- **mode** (*AnomalyMapMode*) – Anomaly map mode.

**Returns:**

Anomaly maps.

**Return type:**

Tensor

*class* **anomalib.models.image.csflow.anomaly_map.AnomalyMapMode**(*value*, *names=None*, **, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: `str`, `Enum`

Generate anomaly map from all the scales or the max.

Previous
C-Flow

Next
DFKDE