Normalizing Flows

Contents

All In One Block Layer.

class **anomalib.models.components.flow.AllInOneBlock(*dims_in*, *dims_c=None*, *subnet_constructor=None*, *affine_clamping=2.0*, *gin_block=False*, *global_affine_init=1.0*, *global_affine_type='SOFTPLUS'*, *permute_soft=False*, *learned_householder_permutation=0*, *reverse_permutation=False*)**

Bases: `InvertibleModule`

Module combining the most common operations in a normalizing flow or similar model.

It combines affine coupling, permutation, and global affine transformation ('ActNorm'). It can also be used as GIN coupling block, perform learned householder permutations, and use an inverted pre-permutation. The affine transformation includes a soft clamping mechanism, first used in Real-NVP. The block as a whole performs the following computation:

$$y = VR\ \Psi(s_{\text{global}}) \odot \text{Coupling}\Big(R^{-1}V^{-1}x\Big) + t_{\text{global}}$$

- The inverse pre-permutation of x (i.e. $R^{-1}V^{-1}$) is optional (see `reverse_permutation` below).
- The learned householder reflection matrix V is also optional all together (see `learned_householder_permutation` below).
- For the coupling, the input is split into x_1, x_2 along the channel dimension. Then the output of the coupling operation is the two halves $u = \text{concat}(u_1, u_2)$.

$$u_1 = x_1 \odot \exp\left(\alpha \tanh\left(s(x_2)\right)\right) + t(x_2)$$
$$u_2 = x_2$$

Because $\tanh(s) \in [-1, 1]$, this clamping mechanism prevents exploding values in the exponential. The hyperparameter α can be adjusted.

Parameters:

- **subnet_constructor** (`Callable` | `None`) – class or callable `f`, called as `f(channels_in, channels_out)` and should return a torch.nn.Module. Predicts coupling coefficients s, t.
- **affine_clamping** (`float`) – clamp the output of the multiplicative coefficients before exponentiation to +/- `affine_clamping` (see α above).
- **gin_block** (`bool`) – Turn the block into a GIN block from Sorrenson et al, 2019. Makes it so that the coupling operations as a whole is volume preserving.
- **global_affine_init** (`float`) – Initial value for the global affine scaling s_{global}.
- **global_affine_init** – `'SIGMOID'`, `'SOFTPLUS'`, or `'EXP'`. Defines the activation to be used on the beta for the global affine scaling (Ψ above).
- **permute_soft** (`bool`) – bool, whether to sample the permutation matrix R from $SO(N)$, or to use hard permutations instead. Note, `permute_soft=True` is very slow when working with >512 dimensions.
- **learned_householder_permutation** (`int`) – Int, if >0, turn on the matrix V above, that represents multiple learned householder reflections. Slow if large number. Dubious whether it actually helps network performance.
- **reverse_permutation** (`bool`) – Reverse the permutation before the block, as introduced by Putzky et al, 2019. Turns on the $R^{-1}V^{-1}$ pre-multiplication above.

`forward(x, c=None, rev=False, jac=True)`

See base class docstring.

Return type:

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

`output_dims(input_dims)`

Output dimensions of the layer.

Parameters:

input_dims (*list[tuple[int]]*) – Input dimensions.

Returns:

Output dimensions.

Return type:

list[tuple[int]]

Previous
Dimensionality Reduction

Next
Sampling Components