BATCHNORM2D

CLASS `torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True, device=None, dtype=None)` [SOURCE]

Applies Batch Normalization over a 4D input (a mini-batch of 2D inputs with additional channel dimension) as described in the paper Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift .

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

The mean and standard-deviation are calculated per-dimension over the mini-batches and γ and β are learnable parameter vectors of size *C* (where *C* is the input size). By default, the elements of γ are set to 1 and the elements of β are set to 0. The standard-deviation is calculated via the biased estimator, equivalent to *torch.var(input, unbiased=False)*.

Also by default, during training this layer keeps running estimates of its computed mean and variance, which are then used for normalization during evaluation. The running estimates are kept with a default `momentum` of 0.1.

If `track_running_stats` is set to `False`, this layer then does not keep running estimates, and batch statistics are instead used during evaluation time as well.

NOTE

This `momentum` argument is different from one used in optimizer classes and the conventional notion of momentum. Mathematically, the update rule for running statistics here is $\hat{x}_{\text{new}} = (1 - \text{momentum}) \times \hat{x} + \text{momentum} \times x_t$, where $\hat{x}$ is the estimated statistic and x_t is the new observed value.

Because the Batch Normalization is done over the *C* dimension, computing statistics on (*N, H, W*) slices, it's common terminology to call this Spatial Batch Normalization.

Parameters

- **num_features** – C from an expected input of size (N, C, H, W)
- **eps** – a value added to the denominator for numerical stability. Default: 1e-5
- **momentum** – the value used for the running_mean and running_var computation. Can be set to `None` for cumulative moving average (i.e. simple average). Default: 0.1
- **affine** – a boolean value that when set to `True`, this module has learnable affine parameters. Default: `True`
- **track_running_stats** – a boolean value that when set to `True`, this module tracks the running mean and variance, and when set to `False`, this module does not track such statistics, and initializes statistics buffers `running_mean` and `running_var` as `None`. When these buffers are `None`, this module always uses batch statistics. in both training and eval modes. Default: `True`

Shape:

- Input: (N, C, H, W)
- Output: (N, C, H, W) (same shape as input)

Examples:

```
>>> # With Learnable Parameters
>>> m = nn.BatchNorm2d(100)
>>> # Without Learnable Parameters
>>> m = nn.BatchNorm2d(100, affine=False)
>>> input = torch.randn(20, 100, 35, 45)
>>> output = m(input)
```

‹ Previous Next ›