Normalization

Junkun Yuan
Zhejiang University
yuanjk@zju.edu.cn

1. Introduction

Normalization techniques is proposed to speed up the convergence of the optimization of deep neural networks by processing the input tensors (e.g., BN [2], LN [1] , IN [4], GN [5]) or the optimized parameters (e.g., WN [3]).

We summarize normalization methods with key phrases:

- **Batch Normalization (BN):** per channel at training, reduce internal covariate shift, accelerate training, reduce dependence of gradients on scale and initial values of parameters, can use higher learning rates, reduce need for Dropout, can use saturating nonlinearities.
- **Layer Normalization (LN):** per sample per layer at training and test, no batchsize constraint, online, RNN.
- **Instance Normalization (IN):** per sample per channel at training and test, preserve instance characteristics.
- **Group Normalization (GN):** per sample per group, divide channels into groups, small batch size.
- **Weight Normalization (WN):** decompose weight into magnitude and direction and optimize them instead of the original weight to speed up the convergence.

2. Batch Normalization (BN)

The change in the distributions of internal nodes of a deep network is called *internal covariate shift*. **BN** [2] reduces internal covariate shift and accelerates the training. It fixes means and variances of layer inputs x:

$$y = \frac{x - E(x)}{Var(x) + \epsilon} * \gamma + \beta \tag{1}$$

BN also reduces the dependence of gradients on the scale of the parameters or their initial values. It allows us to use much higher learning rates without the risk of divergence. Furthermore, BN regularizes the model and reduces the need for Dropout, because a training example is affected by the random selection of the examples in the same mini-batch. Finally, BN makes it possible to use saturating nonlinearities by preventing the network from getting stuck in the saturated modes. (From Sec. 1 of [2])

Code example of BN.

```
# Input X in (batchsize, channel, height, width)
# Parameters gamma, beta, momentum
mean = X.mean(dim=(0, 2, 3), keepdim=True)
var = ((X - mean) ** 2).mean(dim=(0, 2, 3),
                              keepdim=True)
X_hat = (X - mean) / torch.sqrt(var + eps)
# moving averages of mean and var are for test
moving_mean = momentum * moving_mean +
               (1 - momentum) * mean
moving_var = momentum * moving_var +
               (1 - momentum) * var
Y = gamma * X_hat + beta
```

3. Layer Normalization (LN)

The effect of batch normalization is dependent on the mini-batch size and it is not obvious how to apply it to recurrent neural networks. **LN** [1] computes the mean and variance used for normalization from all of the summed inputs to the neurons in a layer on a single training case. All the hidden units in a layer share the same normalization terms, but different training cases have different normalization terms. Unlike BN, LN does not impose any constraint on the size of a mini-batch and it can be used online. It performs at training and test time. (From Sec. 3 of [1])

4. Instance Normalization (IN)

Let input $x \in \mathbb{R}^{T\times C\times W\times H}$ with batchsize T, channel C, width W, and height H. Then, BN can be expressed by:

$$\begin{aligned} y_{tijk} &= \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \\ \mu_i &= \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm} \\ \sigma_i &= \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H} (x_{tilm} - \mu_i)^2 \end{aligned} \tag{2}$$

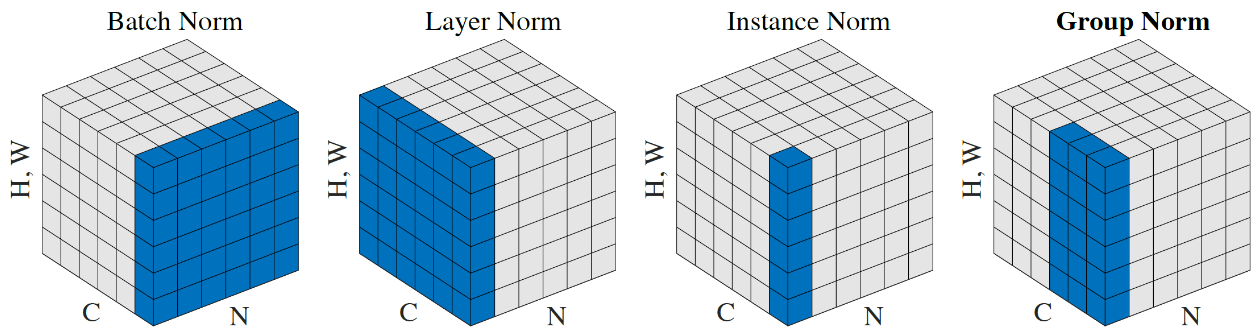

Figure 1. **Normalization methods.** N, C, H, and W represent batch size, channel, height, and width of a feature map tensor, respectively. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels. (From Fig. 2 of [5])

IN [4] combines the effects of instance-specific normalization and batch normalization, that is,

$$\begin{aligned} y_{tijk} &= \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}} \\ \mu_{ti} &= \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} x_{tilm} \\ \sigma_{ti} &= \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} (x_{tilm} - \mu_i)^2 \end{aligned} \tag{3}$$

It preserves instance characteristics for image generation. It is applied at test time as well. (From Sec. 2 of [4])

5. Group Normalization (GN)

BN's error increases rapidly when the batch size becomes smaller, caused by inaccurate batch statistics estimation. It limits BN's usage for training larger models and transferring features to computer vision tasks including detection, segmentation, and video, which require small batches constrained by memory consumption. **GN** [5] divides the channels into groups and computes within each group the mean and the variance for normalization. GN's computation is independent of batch sizes, and its accuracy is stable in a wide range of batch sizes. (From Sec. 3 of [5])

6. Weight Normalization (WN)

WN [3] speeds up the convergence of optimization by reparameterized each weight vector $\mathbf{w}$ in terms of a parameter vector $\mathbf{v}$ and a scalar vector g and performs optimization with respect to $\mathbf{v}$ and g instead. (From Sec. 2 of [3])

$$\mathbf{w} = \frac{g}{||\mathbf{v}||}\mathbf{v} \tag{4}$$

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 1

[3] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016. 1, 2

[4] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 1, 2

[5] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, volume 11217, pages 3–19. Springer, 2018. 1, 2