Batch Normalization https://arxiv.org/abs/1502.03167

Training Time

Let $\{f_1, f_2, f_3, \dots, f_n\}$ be the batched feature vector of some layer (a.k.a. that layer's output, either pre-activation or post-activation). Let's say each $f_i \in \mathbb{R}^d$. The Batch Normalization first computes

$$\tilde{f}_i = (f_i - \mu)/\sigma$$
$$\mu = \frac{1}{n}\sum_{i=1}^{n} f_i$$
$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(f_i - \mu)^2}{n}}$$

Second, computes the affine transform

$$\hat{f}_i = \gamma \cdot \tilde{f}_i + \beta$$

where all arithmetic are element-wise, and γ, β are learnable.

We'll use each $\hat{f}_i$ as next layer's input. Each new batch features $\{\hat{f}_i\}_{i=1}^{n}$ has mean 0 and standard deviation 1.

We can regard BN as a another layer takes the whole batch as input, with weight γ and bias β.

Testing Time

During the testing time, mostly we don't have a bachted data. Hence no on-line μ, σ to compute. Instead, we use the training data help us get them.

For each i-th batch, we have μ_i, σ_i. We can then use Exponential Moving Average to compute the $\bar{\mu}, \bar{\sigma}$ for testing

$$\bar{\mu} = \alpha\bar{\mu} + (1 - \alpha)\mu_i$$
$$\bar{\sigma} = \alpha\bar{\sigma} + (1 - \alpha)\sigma_i$$

where α is a pre-defined hyper parameter. In pytorch, α is called *momentum* and the default value is 0.1.

Batch ReNormalization

https://arxiv.org/abs/1702.03275

Layer Normalization

https://arxiv.org/abs/1607.06450

Instance Normalization

https://arxiv.org/abs/1607.08022

Group Normalization

https://arxiv.org/abs/1803.08494

Weight Normalization

https://arxiv.org/abs/1602.07868

Weight Normalization

https://arxiv.org/abs/1705.10941