Normalization in Deep Neural Networks

Xin Li

June 2023

1 Introduction: Input-Level Normalization

One issue of deep neural nets is that the learning capability of the network deteriorates as the network goes deeper due to vanishing or exploding gradients. Adding normalization layers provides a common remedy to this kind of problem by *standardizing the statistics of the hidden units* (zero mean & unit variance, also known as *whitening*). Specifically, for input feature set $\{\mathbf{x}_i\}_{i=1}^N$ with dimension D, i.e., $\mathbf{x}_i \in \mathbb{R}^D$, we have the "standardization" of the input features on dimension j (j-th feature) can be represented as (the scaled feature will have zero mean and unit variance),

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}, \tag{1}$$

where,

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{j,i} \tag{2}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2 \tag{3}$$

Normalization also helps reduce the impact from features of different scales, as we are applying the same learning rate for all weight parameters, so large parameters will dominate the weight updates while using stochastic gradient descent algorithms.

2 Batch Normalization: Extending Input Normalization to the Hidden Layers

However, normalizing the inputs only affects the first hidden layer, to standardize deeper hidden layer activations, one approach is called the Batch Normalization [1], which ensures the distribution of the hidden layer activations for each layer has a zero mean and unit variance. Some key properties of the batch normalization,

- Normalizes hidden layer inputs.
- Helps with exploding/vanishing gradient problems.
- Can increase training stability and convergence rate.
- Can be understood as additional (normalization) layers (with additional parameters).

When the input distribution to a learning system changes, it is said to experience *covariate shift*, furthermore, the idea of covariate shift can be extended to part of the learning system, such as a layer of the network. Consider the computation on the second layer of a neural network with input uu,

$$\boldsymbol{\ell} = F_2(F_1(u, \Theta_1), \Theta_2), \tag{4}$$

where F_1 and F_2 are arbitrary network activation functions, and the parameters Θ_1, Θ_2 are to be learned to minimize the loss $\boldsymbol{\ell}$. While using Stochastic Gradient Descent (SGD) methods, the training proceeds in mini-batches, i.e., for the training set $\{\mathbf{x}_i\}_{i=1}^N$, at each training step we consider a mini-batch $\{\mathbf{x}_i\}_{i=1}^m$, denote the learning rate as α, we then have for a gradient descent step, the parameter update rule for Θ_2 can be expressed as,

$$\Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial F_2(\mathbf{x}_i, \Theta_2)}{\partial \Theta_2} \tag{5}$$

2.1 Chain Rule Primer

Suppose we have a function $u(x, y)$ where $x(r, t)$ and $y(r, t)$ are also two functions. Then to compute $\partial u/\partial r$ and $\partial u/\partial t$ we apply the chain rule,

$$\frac{\partial u}{\partial r} = \frac{\partial u}{\partial x} \cdot \frac{\partial x}{\partial r} + \frac{\partial u}{\partial y} \cdot \frac{\partial y}{\partial r} \tag{6}$$

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x} \cdot \frac{\partial x}{\partial t} + \frac{\partial u}{\partial y} \cdot \frac{\partial y}{\partial t} \tag{7}$$

2.2 Batch Normalization: Normalization via Mini-Batch Statistics

Consider a mini-batch $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^m$, denote the normalized values as $\{\hat{\mathbf{x}}_i\}_{i=1}^m$ and the corresponding linear transformation as $\{\mathbf{y}_i\}_{i=1}^m$, we refer to the transform,

$$\mathtt{BN}_{\boldsymbol{\gamma},\boldsymbol{\beta}} : \{\mathbf{x}_i\}_{i=1}^m \to \{\mathbf{y}_i\}_{i=1}^m \tag{8}$$

as the *Batch Normalizing Transform*, which can be performed in the following steps,

Algorithm 1 Batch Normalizing Transform

```
Input: values of $\mathbf{x}$ over a mini-batch: $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^m$
Parameters to be learned: $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$
Output: $\{\mathbf{y}_i = \mathrm{BN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{x}_i)\}$
1: procedure $\mathrm{BN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{x}_i)$
2:     $\boldsymbol{\mu}_{\mathcal{B}} \leftarrow (1/m) \sum_{i=1}^m \mathbf{x}_i$                                  ▷ mini-batch mean
3:     $\sigma_{\mathcal{B}}^2 \leftarrow (1/m) \sum_{i=1}^m \|\mathbf{x}_i - \mu_{\mathcal{B}}\|^2$                    ▷ mini-batch variance
4:     $\hat{\mathbf{x}}_i \leftarrow (\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}) / \sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}$                          ▷ normalize
5:     $\mathbf{y}_i \leftarrow \boldsymbol{\gamma} \odot \hat{\mathbf{x}}_i + \boldsymbol{\beta} \equiv \mathtt{BN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{x}_i)$                    ▷ scale and shift
```

where ϵ is a small constant for numerical stability, say $1e-5$ and $\odot$ denotes the Hadamard product. Note that here the parameter $\boldsymbol{\gamma}$ controls the spread or scale and the parameter $\boldsymbol{\beta}$ controls the mean, which also makes the original bias terms in the network redundant. Since the transformation is differentiable, we can easily pass the gradients back to the input of the layer and to the batch normalization parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. Specifically, we have during the backpropagation (not simplified),

$$\frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} = \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \odot \boldsymbol{\gamma} \tag{9}$$

$$\frac{\partial \boldsymbol{\ell}}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot (\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}) \cdot \left(-\frac{1}{2}\right) (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2} \tag{10}$$

$$\frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \boldsymbol{\ell}}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}})}{m} \tag{11}$$

$$\frac{\partial \boldsymbol{\ell}}{\partial \mathbf{x}_i} = \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \boldsymbol{\ell}}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}^2)}{m} + \frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} \cdot \frac{1}{m} \tag{12}$$

$$\frac{\partial \boldsymbol{\ell}}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \cdot \hat{\mathbf{x}}_i \tag{13}$$

$$\frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \tag{14}$$

2.3 Derivations

We first express the forward pass in the diagram below, for a mini-batch $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^{m}$, we have,

$$\mathbf{x} \to \hat{\mathbf{x}}(\boldsymbol{\mu}_{\mathcal{B}}, \sigma_{\mathcal{B}}^2, \mathbf{x}) \to \mathbf{y}(\hat{\mathbf{x}}, \boldsymbol{\gamma}, \boldsymbol{\beta}) \to \boldsymbol{\ell}(\mathbf{y}), \tag{15}$$

hence the backward pass can be expressed as,

$$\boldsymbol{\ell}(\mathbf{y}) \to \mathbf{y}(\hat{\mathbf{x}}, \boldsymbol{\gamma}, \boldsymbol{\beta}) \to \hat{\mathbf{x}}(\boldsymbol{\mu}_{\mathcal{B}}, \sigma_{\mathcal{B}}^2, \mathbf{x}). \tag{16}$$

First we have $\frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i}$ available as the upstream derivative, which is given in the function parameter.

Then to compute $\mathbf{y}(\hat{\mathbf{x}}, \boldsymbol{\gamma}, \boldsymbol{\beta})$, which consists of three variables, we first compute the gradients with respect to each variable,

$$\begin{aligned} \frac{\partial \boldsymbol{\ell}}{\partial \gamma} &= \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \cdot \frac{\partial \mathbf{y}_i}{\partial \boldsymbol{\gamma}} \qquad (17) \\ &= \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \cdot \hat{\mathbf{x}}_i, \qquad (18) \end{aligned}$$

the summation indicates the computation is for the batches. As for $\boldsymbol{\beta}$ we have,

$$\begin{aligned} \frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\beta}} &= \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \cdot \frac{\partial \mathbf{y}_i}{\partial \boldsymbol{\beta}} \qquad (19) \\ &= \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i}, \qquad (20) \end{aligned}$$

and finally $\hat{\mathbf{x}}_i$,

$$\frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} = \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \cdot \frac{\partial \mathbf{y}_i}{\partial \hat{\mathbf{x}}_i} \tag{21}$$

$$= \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \odot \boldsymbol{\gamma}. \tag{22}$$

To compute the gradient with respect to $\mathbf{x}_i$, we have since $\boldsymbol{\mu}$ and σ^2 is a function of $\boldsymbol{\mu}$,

$$\frac{\partial \boldsymbol{\ell}}{\partial \mu_{\mathcal{B}}} = \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \frac{\partial \hat{\mathbf{x}}_i}{\partial \boldsymbol{\mu}_{\mathcal{B}}} + \frac{\partial \boldsymbol{\ell}}{\partial \sigma^2_{\mathcal{B}}} \cdot \frac{\partial \sigma^2_{\mathcal{B}}}{\partial \boldsymbol{\mu}_{\mathcal{B}}}, \tag{23}$$

we have,

$$\frac{\partial \hat{\mathbf{x}}_i}{\partial \boldsymbol{\mu}_{\mathcal{B}}} = -\frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}, \tag{24}$$

and,

$$\frac{\partial \sigma^2_{\mathcal{B}}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} = -\frac{1}{m}\sum_{i=1}^{m} 2 \cdot (\boldsymbol{x}_i - \boldsymbol{\mu}_{\mathcal{B}}). \tag{25}$$

Next we compute the partial,

$$\begin{aligned}
\frac{\partial \boldsymbol{\ell}}{\partial \sigma^2_{\mathcal{B}}} &= \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}} \cdot \frac{\partial \hat{\mathbf{x}}}{\partial \sigma^2_{\mathcal{B}}} &(26)\\
&= \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}} \cdot \left(-\frac{1}{2}\sum_{i=1}^{m} \frac{\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}}{(\sigma^2_{\mathcal{B}} + \epsilon)^{3/2}}\right). &(27)
\end{aligned}$$

Thus we have,

$$\begin{aligned}
\frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} &= \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}} \cdot \frac{\partial \hat{\mathbf{x}}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} + \frac{\partial \boldsymbol{\ell}}{\partial \sigma^2_{\mathcal{B}}} \cdot \frac{\partial \sigma^2_{\mathcal{B}}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} &(28)\\
&= \left(\sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}\right)\right) + &(29)\\
&\quad \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}} \cdot \left(-\frac{1}{2}\sum_{i=1}^{m} \frac{\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}}{(\sigma^2_{\mathcal{B}} + \epsilon)^{3/2}}\right) \cdot \left(-\frac{1}{m}\sum_{i=1}^{m} 2 \cdot (\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}})\right) &(30)\\
&= \left(\sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}\right)\right) + \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}} \cdot \left(-\frac{1}{2}\sum_{i=1}^{m} \frac{\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}}{(\sigma^2_{\mathcal{B}} + \epsilon)^{3/2}}\right) \cdot \mathbf{0} &(31)\\
&= \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}\right). &(32)
\end{aligned}$$

Finally we derive the partial for $\mathbf{x}$, which is a function of variables $\hat{\mathbf{x}}$, $\boldsymbol{\mu}_{\mathcal{B}}$, and $\sigma^2_{\mathcal{B}}$,

$$\frac{\partial \boldsymbol{\ell}}{\partial \mathbf{x}_i} = \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \frac{\partial \hat{\mathbf{x}}_i}{\partial \mathbf{x}_i} + \frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} \cdot \frac{\partial \boldsymbol{\mu}_{\mathcal{B}}}{\partial \mathbf{x}_i} + \frac{\partial \boldsymbol{\ell}}{\partial \sigma^2_{\mathcal{B}}} \cdot \frac{\partial \sigma^2_{\mathcal{B}}}{\partial \mathbf{x}_i} \tag{33}$$

$$= \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} + \frac{\partial \boldsymbol{\ell}}{\partial \sigma^2_{\mathcal{B}}} \cdot \frac{2(\mathbf{x}_i - \boldsymbol{\mu}^2_{\mathcal{B}})}{m} + \frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} \cdot \frac{1}{m} \tag{34}$$

$$= \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \cdot \boldsymbol{\gamma} \cdot \frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} + \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \right) \cdot \frac{1}{m} + \tag{35}$$

$$\frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{2} \sum_{i=1}^{m} \frac{\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}}{(\sigma^2_{\mathcal{B}} + \epsilon)^{3/2}} \right) \cdot \frac{2(\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}})}{m} \tag{36}$$

$$= \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \cdot \boldsymbol{\gamma} \cdot \frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} + \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \right) \cdot \frac{1}{m} + \tag{37}$$

$$\frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{2} \sum_{i=1}^{m} \frac{\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \right) \cdot \frac{2(\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}})}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \cdot \frac{1}{m \cdot \sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}, \tag{38}$$

also note that,

$$\hat{\mathbf{x}}_i = \frac{\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}, \tag{39}$$

we have,

$$\frac{\partial \boldsymbol{\ell}}{\partial \mathbf{x}_i} = \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \frac{\partial \hat{\mathbf{x}}_i}{\partial \mathbf{x}_i} + \frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} \cdot \frac{\partial \boldsymbol{\mu}_{\mathcal{B}}}{\partial \mathbf{x}_i} + \frac{\partial \boldsymbol{\ell}}{\partial \sigma^2_{\mathcal{B}}} \cdot \frac{\partial \sigma^2_{\mathcal{B}}}{\partial \mathbf{x}_i} \tag{40}$$

$$= \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} + \frac{\partial \boldsymbol{\ell}}{\partial \sigma^2_{\mathcal{B}}} \cdot \frac{2(\mathbf{x}_i - \boldsymbol{\mu}^2_{\mathcal{B}})}{m} + \frac{\partial \boldsymbol{\ell}}{\partial \boldsymbol{\mu}_{\mathcal{B}}} \cdot \frac{1}{m} \tag{41}$$

$$= \frac{\partial \boldsymbol{\ell}}{\partial \mathbf{y}_i} \cdot \boldsymbol{\gamma} \cdot \frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} + \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \right) \cdot \frac{1}{m} + \tag{42}$$

$$\frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} \cdot \left(-\frac{1}{2} \sum_{i=1}^{m} \hat{\mathbf{x}}_i \right) \cdot \frac{2\hat{\mathbf{x}}_i}{m \cdot \sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \tag{43}$$

$$= \frac{1}{m\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \cdot \left[m \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_i} - \sum_{j=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_j} - \hat{\mathbf{x}}_i \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{x}}_j} \cdot \hat{\mathbf{x}}_j \right] \tag{44}$$

$$= \frac{m \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{y}}_i} - \sum_{j=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{y}}_j} - \hat{\mathbf{x}}_i \sum_{i=1}^{m} \frac{\partial \boldsymbol{\ell}}{\partial \hat{\mathbf{y}}_j} \cdot \hat{\mathbf{x}}_j}{m\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \odot \boldsymbol{\gamma}, \tag{45}$$

where $\{\partial \boldsymbol{\ell} / \partial \mathbf{y}_i\}_{i=1}^{m}$ are available as the function parameter.

Note that after the "whitening" process, we have the input to the next activation layer,

$$\mathbf{y}_i \leftarrow \boldsymbol{\gamma}\hat{\mathbf{x}}_i + \boldsymbol{\beta}, \tag{46}$$

where $\boldsymbol{\beta}$ makes the requirement for the bias term in the activation function redundant as the learning procedes. However, by whitening we lose the ability to perform deterministic test-time inference as we have quantities ($\boldsymbol{\mu}_{\mathcal{B}}$, $\sigma^2_{\mathcal{B}}$) that depend on the mini-batch instead of on individual samples alone, where the rest of the samples in the mini-batch are randomly selected. So to have some deterministic and reproducible results at test time, people use running averages of the quantities dependent on the mini-batch during training. Specifically, for multiple mini-batches of training, compute the average (over mini-batch means and variances), by taking the running averages, those values are not dependent on the mini-batches anymore, so we have deterministic inference during test-time. The below procedure summarizes the training of batch-normalized networks,

Algorithm 2 Training Batch-Normalized Networks

```
Input: network $N$ with trainable parameters $\Theta$; subset of layer activations $\{\mathbf{x}^{(k)}\}_{k=1}^{K}$.
Parameters to be learned: $\Theta$, $\boldsymbol{\beta}$
Output: batch-normalized network for inference $N^{\text{inf}}_{\text{BN}}$
 1: $N^{\text{tr}}_{\text{BN}} \leftarrow N$                                  ▷ training a batch-normalized network
 2: for $k = 1 \cdots K$ do
 3:     add transformation $\mathbf{y}^{(k)} = \text{BN}_{\boldsymbol{\gamma}^{(k)},\boldsymbol{\beta}^{(k)}}(\mathbf{x}^{(k)})$ to the network $N^{t}_{\text{BN}}r \leftarrow N$ (Algorithm 1)
 4:     modify each layer in $N^{\text{tr}}_{\text{BN}}$ with input $\mathbf{x}^{(k)}$ to take $\mathbf{y}^{(k)}$ instead
 5: train $N^{\text{tr}}_{\text{BN}}$ to optimize the model parameters $\Theta \cup \{\boldsymbol{\gamma}^{(k)}, \boldsymbol{\beta}^{(k)}\}_{k=1}^{K}$.
 6: $N^{\text{inf}}_{\text{BN}} \leftarrow N^{\text{tr}}_{\text{BN}}$                  ▷ freeze batch-normalized network parameters for inference
 7: for $k = 1, \cdots, K$ do
 8:                        ▷ for simplicity, denote $\mathbf{x} \equiv \mathbf{x}^{(k)}$, $\boldsymbol{\gamma} \equiv \boldsymbol{\gamma}^{(k)}$, $\boldsymbol{\mu}_{\mathcal{B}} \equiv \boldsymbol{\mu}^{(k)}_{\mathcal{B}}$, etc.
 9:     process multiple mini-batches $\mathcal{B}$, each of size $m$, and compute the running average,
            $\mathbb{E}[\mathbf{x}] \leftarrow \mathbb{E}_{\mathcal{B}}[\boldsymbol{\mu}_{\mathcal{B}}]$                        (47)
            $\text{Var}[\mathbf{x}] \leftarrow \frac{m}{m-1}\mathbb{E}_{\mathcal{B}}[\sigma^2_{\mathcal{B}}]$                 (48)
10:     in the batch-normalized network $N^{\text{inf}}_{\text{BN}}$, replace the transform $\mathbf{y} = \text{BN}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\mathbf{x})$ with,
            $\mathbf{y} = \frac{\boldsymbol{\gamma}}{\sqrt{\text{Var}[\mathbf{x}] + \epsilon}} \cdot \mathbf{x} + \left(\boldsymbol{\beta} - \frac{\boldsymbol{\gamma}\mathbb{E}[\mathbf{x}]}{\sqrt{\text{Var}[\mathbf{x}] + \epsilon}}\right)$   (49)
```

Batch normalization reduces the internal covariate shift across the hidden layers and thus enables much higher learning rates and faster convergence.

2.4 Why Batch Normalization Works?

The original batch normalization paper [1] proposes that the batch normalization helps training by reducing the internal covariate shift (feature shift in hidden layers) where each layer of a deep neural net can be thought of as solving an *empirical risk minimization problem* with layer inputs and some loss function to optimize and the layer inputs exhibit some *distributional changes*, but there's no guarantee or strong evidence showing that the batch normalization helps with the internal covariate shift. However, in a later work by Shibani *et al.* [2] claims that the distributional stability of layer inputs has little to do with the success of batch normalization. Specifically, we define the internal covariate shift for the underlying optimization task with the first-order gradient method as the optimizer in mind,

Definition 2.1 (Internal Covariate Shift)**.** *Let $\boldsymbol{\ell}$ be the loss function, $\{\mathbf{W}_t^{(k)}\}$ be the weight parameters of layer k, and input-label pair $(\mathbf{x}_t, \mathbf{y}_t)$ be the batch feeding to the network at training time-step t. Then we define the* ***Internal Covariate Shift*** *(ICS) of the layer activation i at time t with the gradients of the layer parameters,*

$$\mathbf{G}_{t,i} = \nabla_{\mathbf{W}_t^{(i)}} \boldsymbol{\ell}\Big(\mathbf{W}_t^{(1)}, \cdots, \mathbf{W}_t^{(k)}; \mathbf{x}_t, \mathbf{y}_t\Big) \tag{50}$$

$$\mathbf{G}'_{t,i} = \nabla_{\mathbf{W}_t^{(i)}} \boldsymbol{\ell}\Big(\mathbf{W}_{t+1}^{(1)}, \cdots, \mathbf{W}_{t+1}^{(i-1)}, \mathbf{W}_t^{(i)}, \mathbf{W}_t^{(t+1)}, \cdots, \mathbf{W}_t^{(k)}; \mathbf{x}_t, \mathbf{y}_t\Big), \tag{51}$$

where $\mathbf{G}_{t,i}$ denotes the gradient of the layer parameters during layer update, and $\mathbf{G}'_{t,i}$ denotes the same gradient after all the previous layers have been updated with the new values.

With this definition in mind, the experiments as proposed in [2] show that batch normalization might **not** even reduce the internal covariate shift.

Shibani *et al.* [2] further shows that what batch normalization really helps is offering a significantly smoother optimization landscape (i.e., Lipschitzness of the loss landscape), which eventually induces a more predictive and stable behavior of the network gradients, and thus speeds up the training. To see that, we first review a common concept called *Lipschitzness* in optimization,

Definition 2.2 (Lipschitz-Continuity)**.** *A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is Lipschitz-continuous with constant L over a set $\mathcal{X} \subset \mathbb{R}^n$, i.e.,*

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \le L\|\mathbf{x}_1 - \mathbf{x}_2\|, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}. \tag{52}$$

We call the function f is L-Lipschitz for all $\mathbf{x}_1$ and $\mathbf{x}_2$.

First the following theorem shows that batch normalization induces better Lipschitzness of the loss function in terms of the magnitude of the gradient $\|\nabla_{\mathbf{x}^{(k)}}\boldsymbol{\ell}\|$ at network layer $k = 1, 2, \cdots, K$.

Theorem 2.1 (Batch Normalization Helps with Lipschitz-Continuity)**.** *Consider network layer activations $\mathbf{x}^{(k)}$ for layer k, denote the loss function of the m-batch normalized network as $\hat{\boldsymbol{\ell}}$ and $\boldsymbol{\ell}$ for the corresponding non-BN network with identical loss function, we have,*

$$\|\nabla_{\mathbf{x}^{(k)}}\hat{\boldsymbol{\ell}}\|^2 \le \frac{\gamma^2}{\left(\sigma_{\mathcal{B}}^{(k)}\right)^2}\left(\|\nabla_{\mathbf{x}^{(k)}}\boldsymbol{\ell}\|^2 - \frac{1}{m}\langle \mathbf{1}, \nabla_{\mathbf{x}^{(k)}}\boldsymbol{\ell}\rangle^2 - \frac{1}{m}\left\langle \nabla_{\mathbf{x}^{(k)}}\boldsymbol{\ell}, \hat{\mathbf{x}}^{(k)}\right\rangle^2\right) \tag{53}$$

Proof. First we have the gradient of the m-batch normalized network w.r.t. to activations for batch $\mathcal{B}$,

$$\nabla_{\mathbf{x}^{(k)}}\hat{\boldsymbol{\ell}} = \frac{\partial\hat{\boldsymbol{\ell}}}{\partial\mathbf{x}^{(k)}} = \left(\frac{\gamma}{m\sigma_{\mathcal{B}}^{(k)}}\right)\left(m\frac{\partial\hat{\boldsymbol{\ell}}}{\partial\mathbf{y}^{(k)}} - \mathbf{1}\left\langle \mathbf{1}, \frac{\partial\hat{\boldsymbol{\ell}}}{\partial\mathbf{y}^{(k)}}\right\rangle - \hat{\mathbf{x}}^{(k)}\left\langle \frac{\partial\hat{\boldsymbol{\ell}}}{\partial\mathbf{y}^{(k)}}, \hat{\mathbf{x}}^{(k)}\right\rangle\right), \tag{54}$$

also note that after normalization $\hat{\mathbf{x}}^{(k)}$ is zero-mean and $\sqrt{m}$-norm, thus we have,

$$\left\|\nabla_{\mathbf{x}^{(k)}}\hat{\boldsymbol{\ell}}\right\|^2 = \left(\frac{\gamma}{\sigma_{\mathcal{B}}^{(k)}}\right)^2\left\|\frac{\partial\hat{\boldsymbol{\ell}}}{\partial\mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\left\langle \mathbf{1}, \frac{\partial\hat{\boldsymbol{\ell}}}{\partial\mathbf{y}^{(k)}}\right\rangle - \frac{\hat{\mathbf{x}}^{(k)}}{m}\left\langle \frac{\partial\hat{\boldsymbol{\ell}}}{\partial\mathbf{y}^{(k)}}, \hat{\mathbf{x}}^{(k)}\right\rangle\right\|^2 \tag{55}$$

$$= \left(\frac{\gamma}{\sigma_{\mathcal{B}}^{(k)}}\right)^2 \left\| \left(\frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle\right) - \frac{\hat{\mathbf{x}}^{(k)}}{\|\hat{\mathbf{x}}^{(k)}\|}\left\langle \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle, \frac{\hat{\mathbf{x}}^{(k)}}{\|\hat{\mathbf{x}}^{(k)}\|} \right\rangle \right\|^2 \tag{56}$$

$$= \left(\frac{\gamma}{\sigma_{\mathcal{B}}^{(k)}}\right)^2 \left(\left\| \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle \right\|^2 + \left\langle \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle, \frac{\hat{\mathbf{x}}^{(k)}}{\|\hat{\mathbf{x}}^{(k)}\|} \right\rangle^2 - \right.$$
$$\left. 2 \cdot \left(\frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle\right) \cdot \frac{\hat{\mathbf{x}}^{(k)}}{\|\hat{\mathbf{x}}^{(k)}\|}\left\langle \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle, \frac{\hat{\mathbf{x}}^{(k)}}{\|\hat{\mathbf{x}}^{(k)}\|} \right\rangle \right) \tag{57}$$

$$= \left(\frac{\gamma}{\sigma_{\mathcal{B}}^{(k)}}\right)^2 \left(\left\| \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle \right\|^2 - \left\langle \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} - \frac{\mathbf{1}}{m}\underbrace{\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle}_{\text{scalar}}, \frac{\hat{\mathbf{x}}^{(k)}}{\|\hat{\mathbf{x}}^{(k)}\|} \right\rangle^2 \right) \tag{58}$$

$$= \left(\frac{\gamma}{\sigma_{\mathcal{B}}^{(k)}}\right)^2 \left(\left\| \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\|^2 + \frac{1}{m^2}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle^2 \cdot m - \frac{2}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle^2 - \right.$$
$$\left. \left\| \left\langle \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}}, \frac{\hat{\mathbf{x}}^{(k)}}{\|\hat{\mathbf{x}}^{(k)}\|} \right\rangle - \underbrace{\left\langle \frac{\mathbf{1}}{m}, \frac{\hat{\mathbf{x}}^{(k)}}{\|\hat{\mathbf{x}}^{(k)}\|} \right\rangle}_{0} \right\|^2 \right) \tag{59}$$

$$= \left(\frac{\gamma}{\sigma_{\mathcal{B}}^{(k)}}\right)^2 \left(\left\| \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\|^2 - \frac{1}{m}\left\langle \mathbf{1}, \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}} \right\rangle^2 - \frac{1}{m}\left\langle \frac{\partial \hat{\boldsymbol{\ell}}}{\partial \mathbf{y}^{(k)}}, \hat{\mathbf{x}}^{(k)} \right\rangle^2 \right), \tag{60}$$

also note that $\partial\hat{\boldsymbol{\ell}}/\partial\mathbf{y}^{(k)} = \partial\boldsymbol{\ell}/\partial\mathbf{x}$, therefore we have,

$$\left\|\nabla_{\mathbf{x}^{(k)}}\hat{\boldsymbol{\ell}}\right\|^2 \leq \frac{\gamma^2}{\sigma_{\mathcal{B}}^2}\left(\|\nabla_{\mathbf{x}^{(k)}}\boldsymbol{\ell}\|^2 - \frac{1}{m}\langle \mathbf{1}, \nabla_{\mathbf{x}^{(k)}}\boldsymbol{\ell}\rangle^2 - \frac{1}{m}\left\langle \nabla_{\mathbf{x}^{(k)}}\boldsymbol{\ell}, \hat{\mathbf{x}}^{(k)} \right\rangle^2 \right). \tag{61}$$

□

We then proceed to show the minimax bound on the weight-space Lipschitzness,

Theorem 2.2 (Minimax Bound on Weight-Space Lipschitzness). *For a batch-normalized neural network with loss function $\hat{\boldsymbol{\ell}}$ and an identical non-batch-normalized network with identical loss function $\boldsymbol{\ell}$, denote,*

$$g^{(k)} = \max_{\|\mathbf{X}\|\leq\lambda}\|\nabla_{\mathbf{W}}\boldsymbol{\ell}\|^2, \quad \hat{g}^{(k)} = \max_{\|\mathbf{X}\|\leq\lambda}\left\|\nabla_{\mathbf{W}}\hat{\boldsymbol{\ell}}\right\|^2, \tag{62}$$

we then have,

$$\hat{g}^{(k)} \leq \left(\frac{\gamma}{\sigma^{(k)}}\right)^2\left(g_j^2 - m\mu_{g_j}^2 - \lambda^2\left\langle \nabla_{\mathbf{x}^{(k)}}\boldsymbol{\ell}, \hat{\mathbf{x}}^{(k)} \right\rangle^2\right). \tag{63}$$

3 Research Ideas

Consider the multi-camera system, the data collection is over wireless (we don't have unlimited bandwidth), component images, do we want to actually compute the super-resolution images or make decision based on current images, is it worth doing the work to get better data? Say we have a raw input x, $f(x)$ is for generating the labels efficiently, $g(x)$ is good data storage and labeling offline, like the super-resolution version, you don't want to collect the data you don't want to use. You can act the data now, or save it later to spend more time working on it. Understand when should you ask for help for more data? Asking for help in classification is like asking for help in another sensor.

References

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

[2] Shibani Santurkar et al. "How Does Batch Normalization Help Optimization?" In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf.