The Motivations and Operations of Batch Normalization

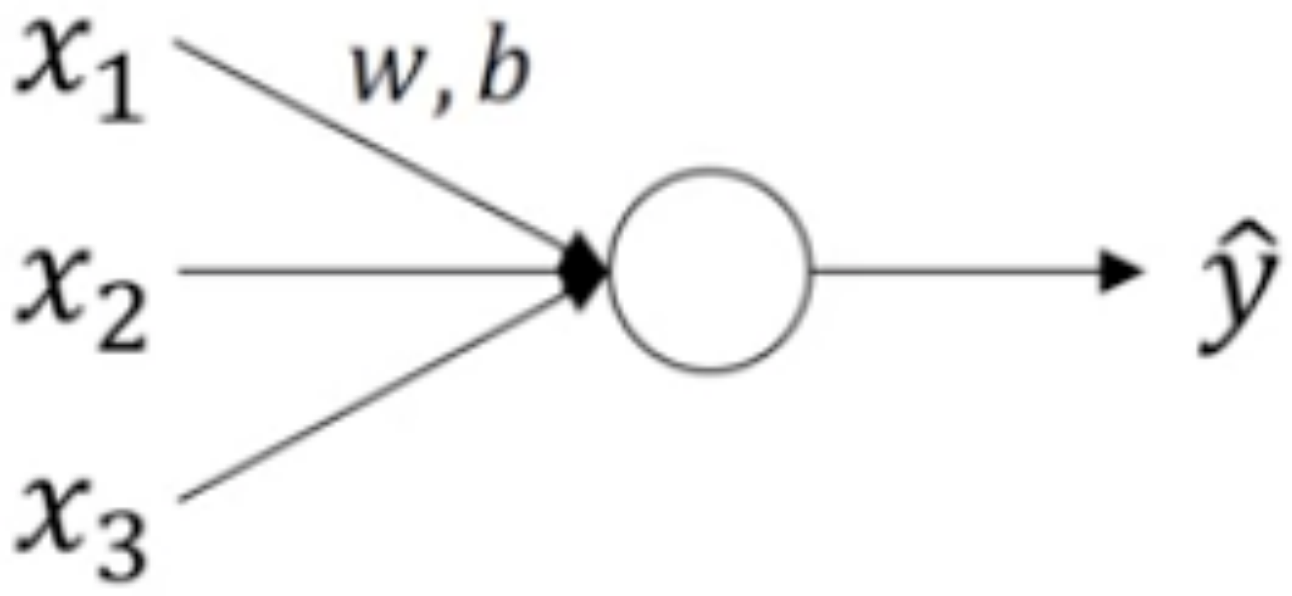
批量归一化操作的目的和实现

Hirah 8.8.2021

Table of Contents

Batch Normalization 操作

- Normalization for LR



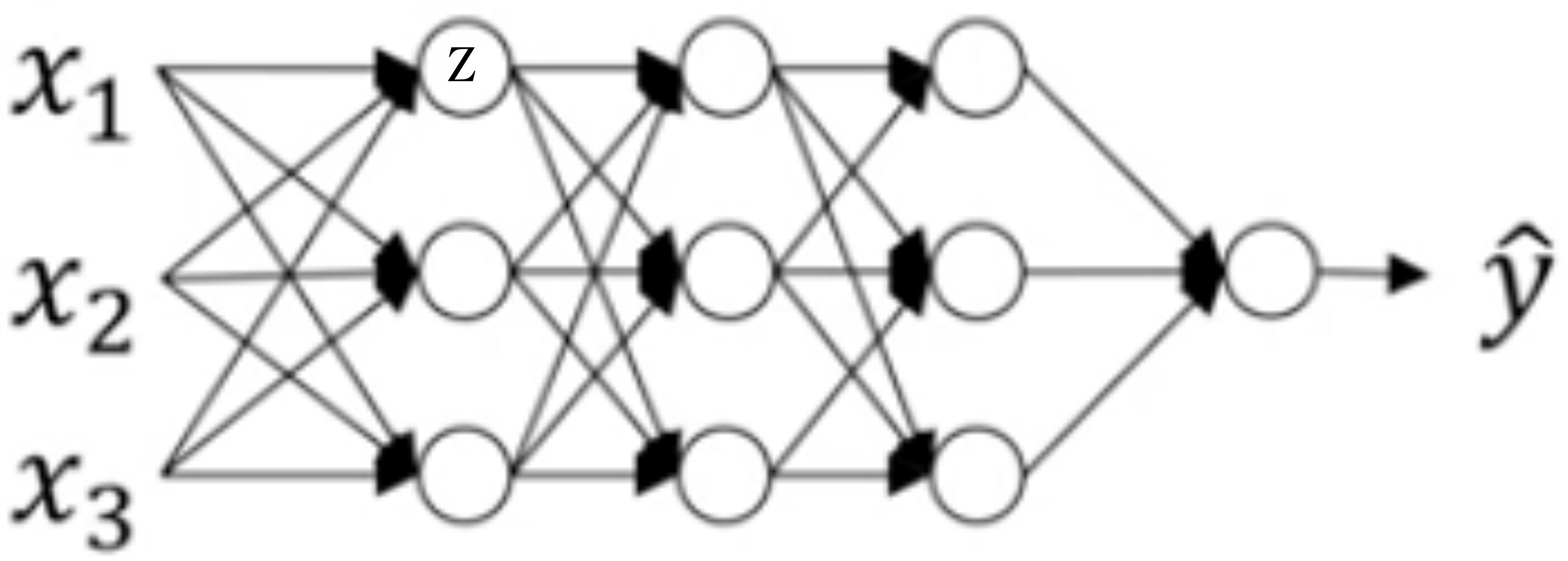
$$\mu_i = \frac{1}{M} \sum x_i$$

$$\sigma_i = \sqrt{\frac{1}{M} \sum \left(x_i - \mu_i\right)^2 + \epsilon}$$

$$x' = \frac{x - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

Batch Normalization 操作

-



$$Z = XW = w_1x_1 + w_2x_2 + w_3x_3$$

$$\tilde{\mathbf{Z}} = \mathbf{Z} - \frac{1}{m}\sum_{i=1}^{m} \mathbf{Z}_{i,:}$$

$$\hat{\mathbf{Z}} = \frac{\tilde{Z}}{\sqrt{\epsilon + \frac{1}{m}\sum_{i=1}^{m} \tilde{\mathbf{Z}}_{i,:}^2}}$$

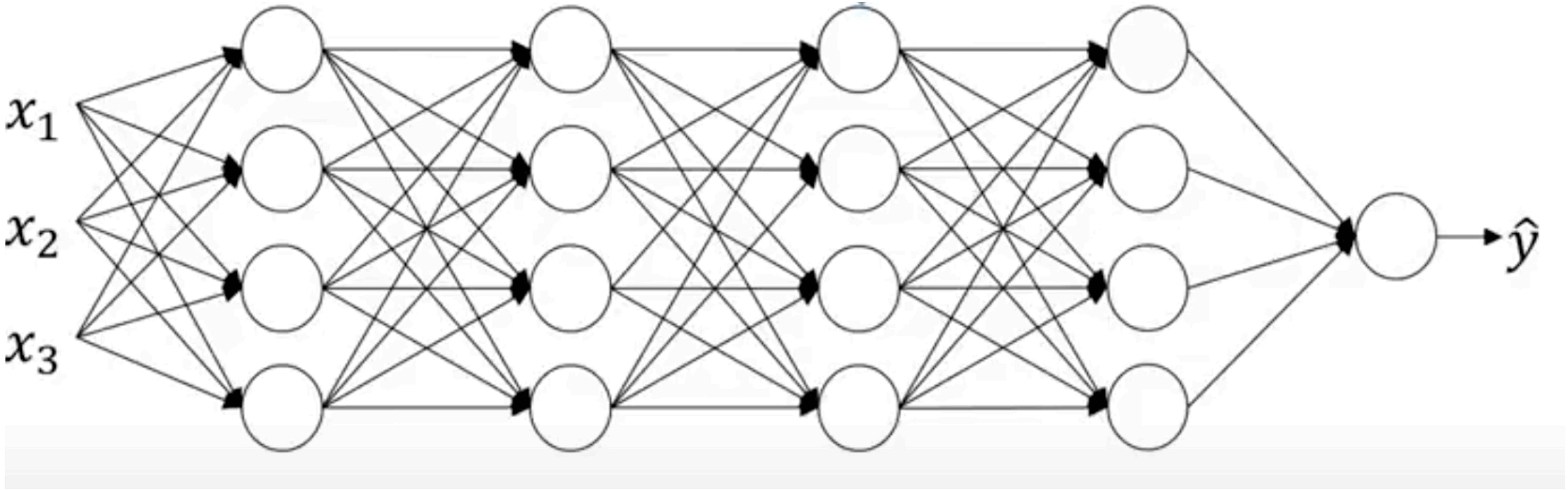
$$\boldsymbol{H} = \max\{0, \gamma\hat{Z} + \boldsymbol{\beta}\}$$

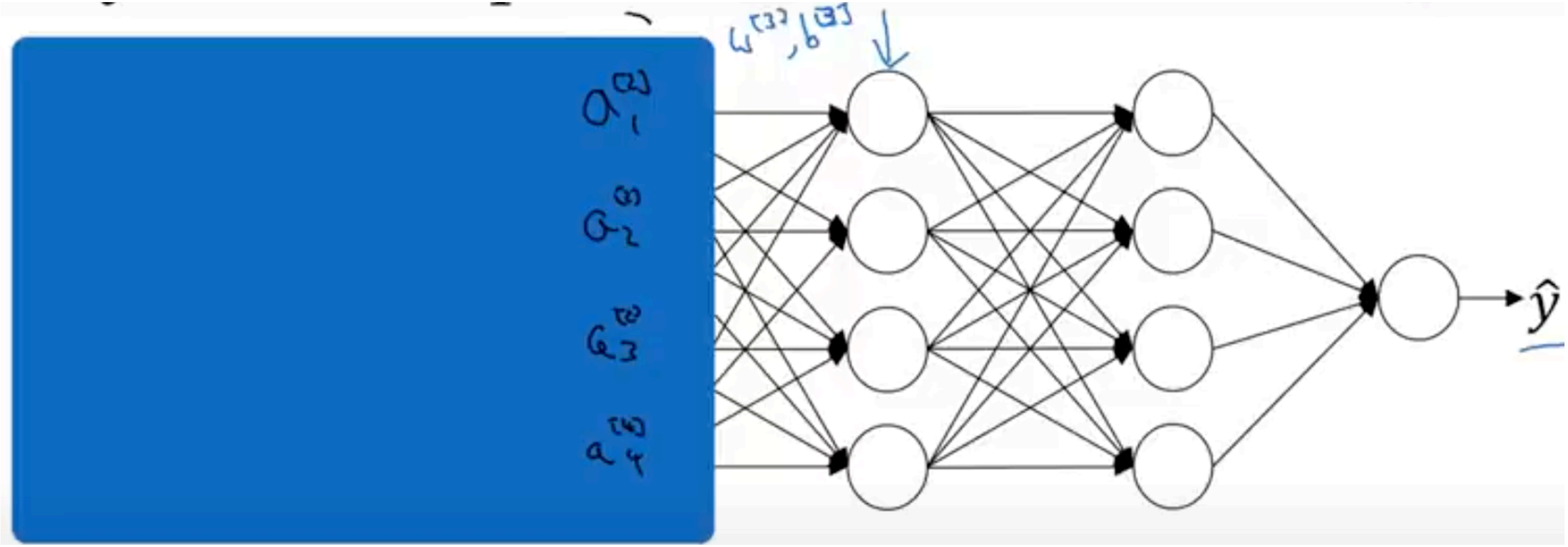
- 当加入BN后，我们不再需要$\phi(\boldsymbol{XW} + \boldsymbol{b})$ 中的Bias term b了
- https://arxiv.org/abs/1502.03167 Ioffe and Szegedy (2015) 推荐在使用激活函数之前做BN

Motivations 目的

Reparametrization

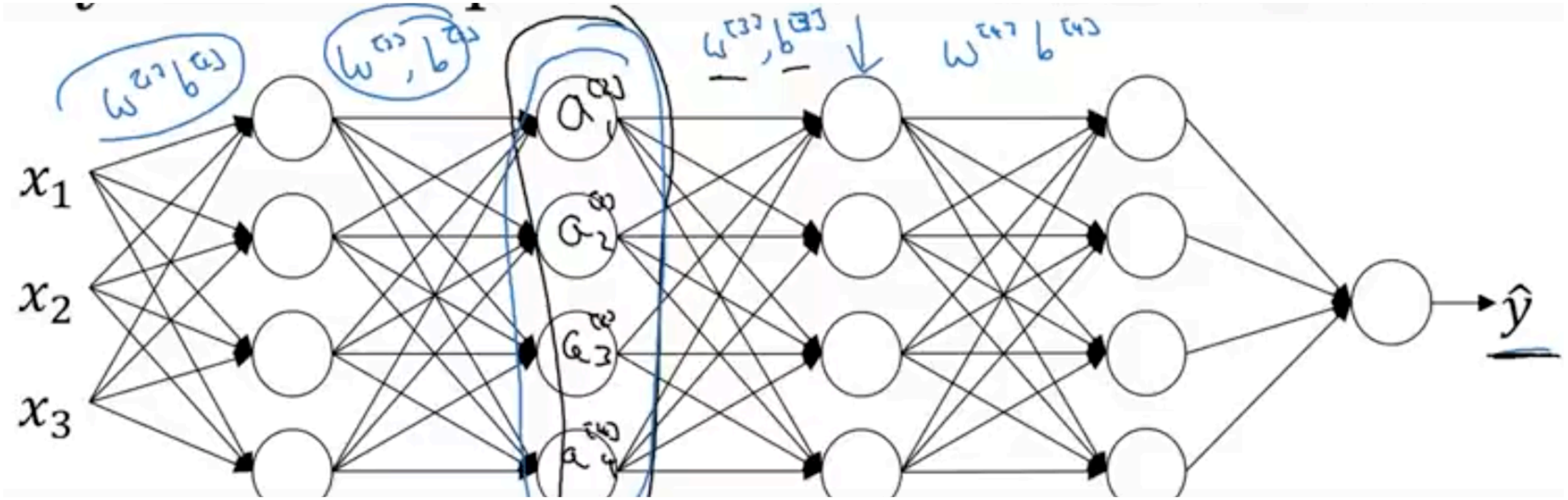
- 批量标准化的意义在于，它能够层数较大的模型稳定训练过程中隐藏层（特别是靠近输出部分的隐藏层）的参数

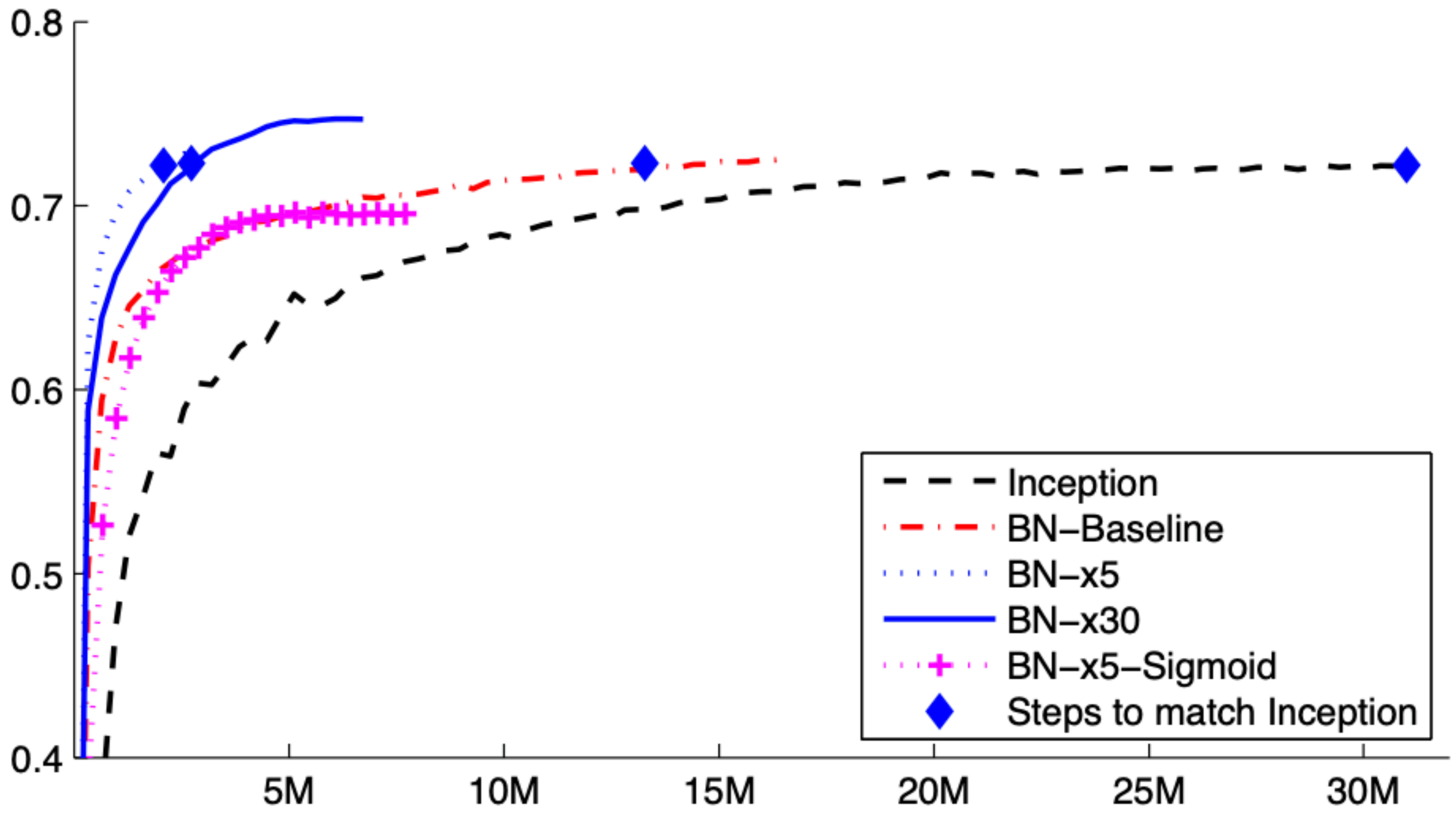



$a_1^{[2]}$
$a_2^{[2]}$
$a_3^{[2]}$
$a_4^{[2]}$
$\hat{y}$

Motivations 目的

- 即使模型训练数据的分布发生变化，导致隐藏节点之前的参数随之发生变化，批量标准化也可以保证隐藏层的输出有稳定的均值和标准差，这样就减小了输入变化时对整个模型参数的影响，提升了模型的稳定性。





"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015

Motivations 目的

批量标准化微弱的正则化效应

- SGD过程中，每一个mini-batch的均值和标准差是单独计算的。
- Mini-batch之间的均值和标准差存在差异（取决于mini-batch的大小）。
- 这种差异，从某种意义上说也是加入到隐藏层中的noise。
- 这和Dropout有相似的地方，Dropout也是在隐藏层中加入noise（不过通常为{0, 1}）
- 不过由于Batch Normalization正则化效应比较微弱，它并不能完全代替Dropout等其他模型正则化手段的作用。

Motivations 目的

- Reparametrization
- Easier hyperparameter tuning
- Feasible for sigmoid active function
- Regularization effect

Implementation

```python
import torch
from torch import nn
from d2l import torch as d2l

def batch_norm(X, gamma, beta, moving_mean, moving_var, eps, momentum):
    # Use `is_grad_enabled` to determine whether the current mode is training
    # mode or prediction mode
    if not torch.is_grad_enabled():
        # If it is prediction mode, directly use the mean and variance
        # obtained by moving average
        X_hat = (X - moving_mean) / torch.sqrt(moving_var + eps)
    else:
        assert len(X.shape) in (2, 4)
        if len(X.shape) == 2:
            # When using a fully-connected layer, calculate the mean and
            # variance on the feature dimension
            mean = X.mean(dim=0)
            var = ((X - mean)**2).mean(dim=0)
        else:
            # When using a two-dimensional convolutional layer, calculate the
            # mean and variance on the channel dimension (axis=1). Here we
            # need to maintain the shape of `X`, so that the broadcasting
            # operation can be carried out later
            mean = X.mean(dim=(0, 2, 3), keepdim=True)
            var = ((X - mean)**2).mean(dim=(0, 2, 3), keepdim=True)
        # In training mode, the current mean and variance are used for the
        # standardization
        X_hat = (X - mean) / torch.sqrt(var + eps)
        # Update the mean and variance using moving average
        moving_mean = momentum * moving_mean + (1.0 - momentum) * mean
        moving_var = momentum * moving_var + (1.0 - momentum) * var
    Y = gamma * X_hat + beta  # Scale and shift
    return Y, moving_mean.data, moving_var.data
```

加入的gamma（标准差）和beta（均值）可以作为参数通过优化器来优化，而不像未标准化时隐藏层的标准差，均值和标准差操控起来较为复杂