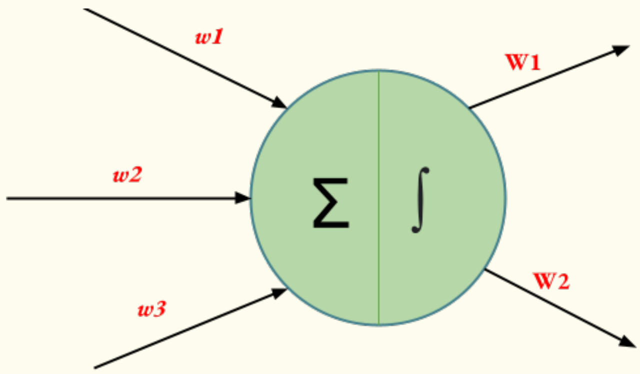
Weight Initialization:

→ to tackle with vanishing and Exploding gradient problem.

→ faster convergence



- fan_in = 3
- fan_out = 2

1. zero initialization:

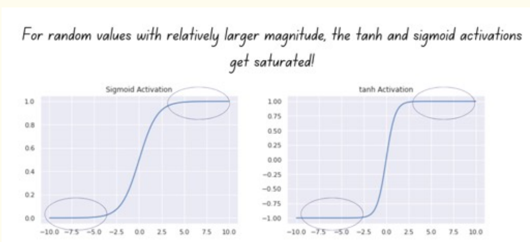
→ highly ineffective as neurons learn the same feature during first few iterations

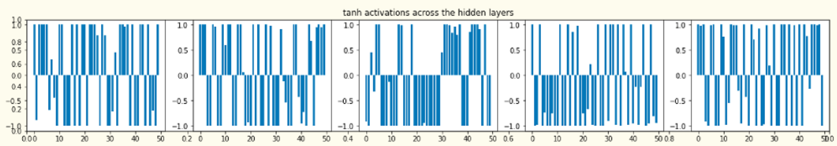
→ slow convergence

2. Random Initialization:

→ better than zero initialization or const initialization.

→ very high or low value may lead to vanishing or Exploding Gradient problem.





→ saturating tanh activations for large random weights.

a) random uniform $\Rightarrow w_i \sim U(0,1)$

b) random normal $\Rightarrow w_i \sim N(0,1)$

3. Xavier / Glorot Initialization:

→ $w_i \sim U\left[-\sqrt{\frac{6}{f_{in}+f_{out}}}, \sqrt{\frac{6}{f_{in}+f_{out}}}\right]$

→ uniform distribution

→ takes account of input and output connections

→ mostly used for sigmoid.

→ weights do not saturate or vanish during forward pass.

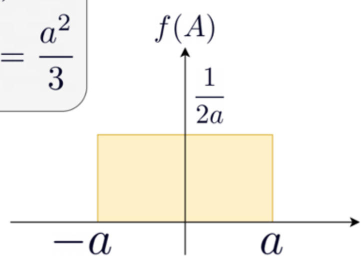
a) Glorot Normal:

$$W \sim N(0, \sigma) \quad \sigma = \sqrt{\frac{6}{f_{in}+f_{out}}}$$

b) Glorot uniform:

$$\mathbb{E}(A) = 0$$

$$Var(A) = \frac{(2a)^2}{12} = \frac{a^2}{3}$$



$$Var(A) = \frac{(2a)^2}{12} = \frac{a^2}{3}$$

We know that the variance should be equal to $\frac{2}{fan_{in}+fan_{out}}$; we can work backward to find the endpoints of the interval.

$$Var(w) = \frac{(2a)^2}{12} = \frac{a^2}{3}$$

We have,

$$Var(w) = \frac{2}{fan_{in} + fan_{out}}$$

$$\implies \frac{a^2}{3} = \frac{2}{fan_{in} + fan_{out}}$$

$$a^2 = \frac{6}{fan_{in} + fan_{out}}$$

$$\implies a = \sqrt{\frac{6}{fan_{in} + fan_{out}}}$$

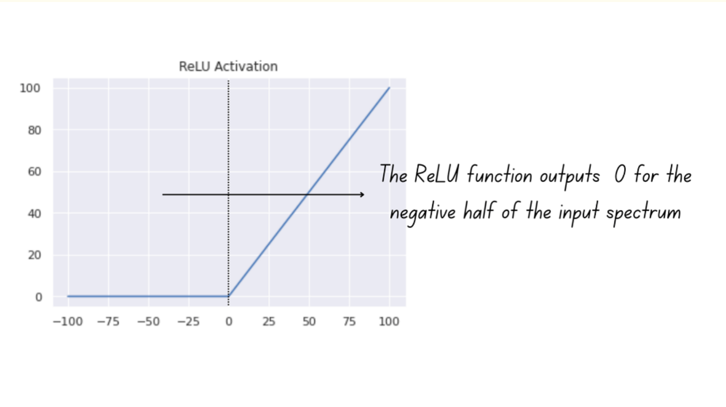
$$w \in \mathcal{U}\left[-\sqrt{\frac{6}{fan_{in} + fan_{out}}}, \sqrt{\frac{6}{fan_{in} + fan_{out}}}\right]$$

4. He Weight Initialization: (Kaiming)

→ mostly used for ReLU and Leaky-ReLU.

→ helps avoiding slow convergence

→ avoid oscillations while reaching minima.



a) Normal:

$$W_i \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{f_{in}}}$$

b) Uniform:

$$W_i \sim U\left[-\sqrt{\frac{6}{f_{in}}}, \sqrt{\frac{6}{f_{in}}}\right]$$

* Another method to prevent Exploding gradient problem is – Gradient Clipping

Batch Normalization

→ faster convergence
  ↳ increase in LR
  ↳ less oscillation
  ↳ reduce training time.

→ remove necessity dropout

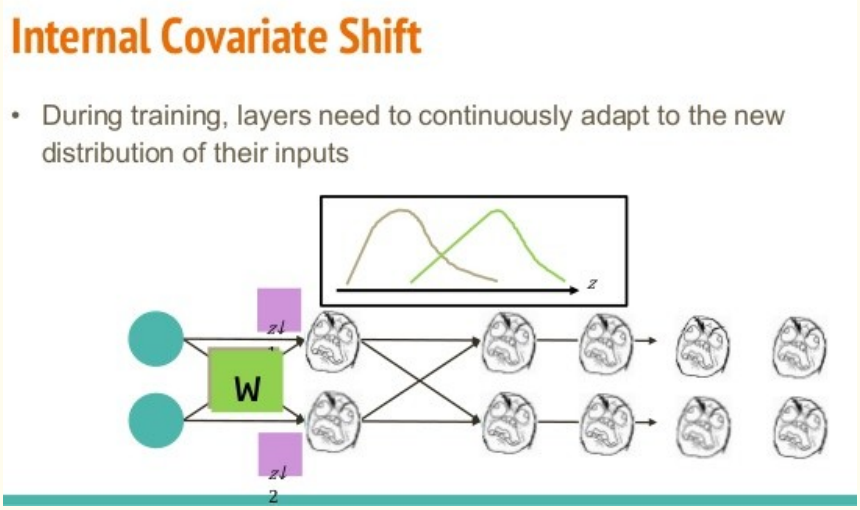
→ prevents from vanishing or exploding gradient problem.

→ saves from dead activation for sigmoid & tanh.

→ reduce dependency on Hyper-parameters.

→ saves NN from Internal covariate shift

→ smoothens the loss function curve.



ICS: change in the distribution of the network activations output or feature map due to change in n/w parameters during training back propagation.

solution:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

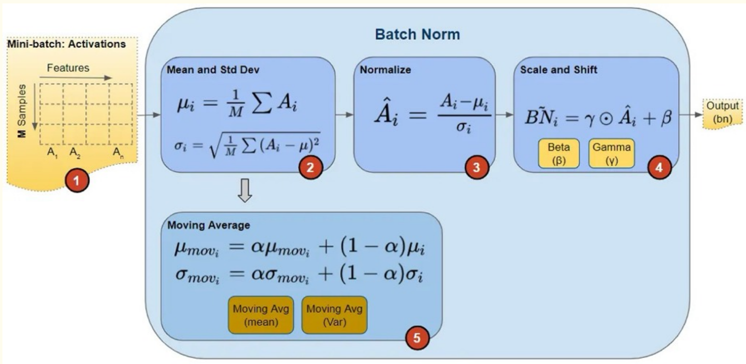
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

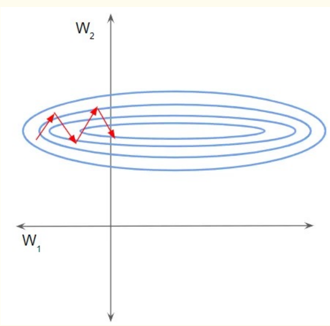
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

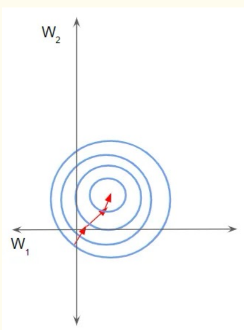
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

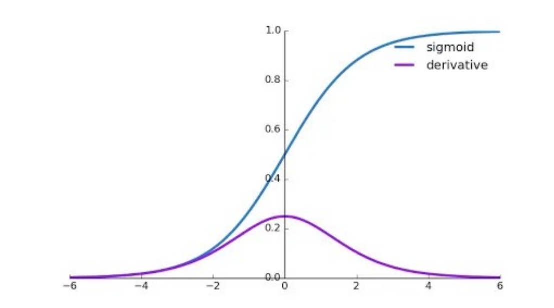
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.



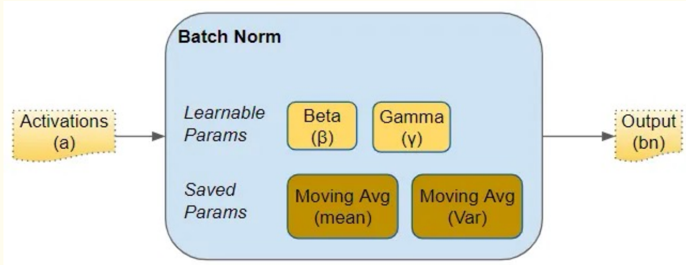




→ smoothens the loss surface that leads to faster convergence.



→ prevent from saturation



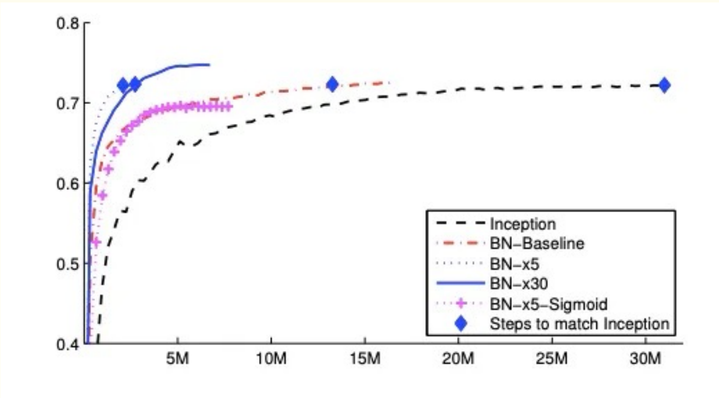


Figure 2. Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.