DEEP LEARNING (CS436)

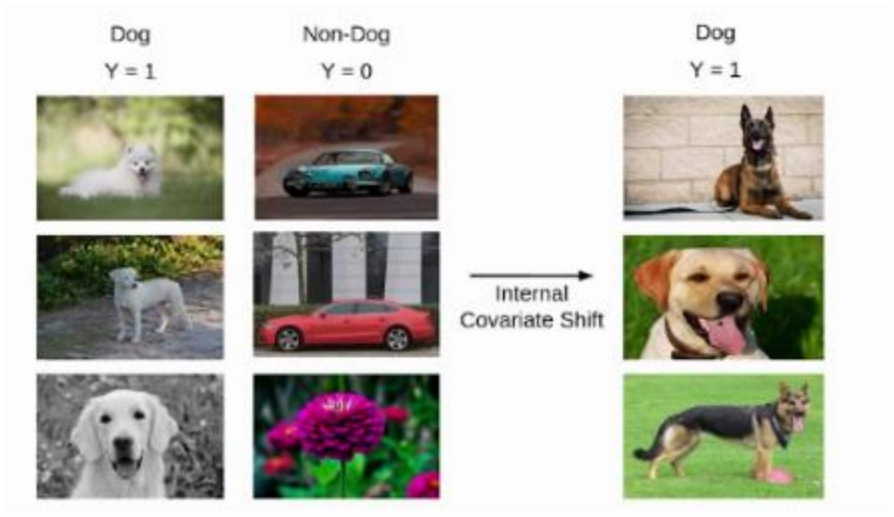
BY: Nidhi S. Periwal,
Teaching Assistant,
DoCSE, SVNIT, Surat

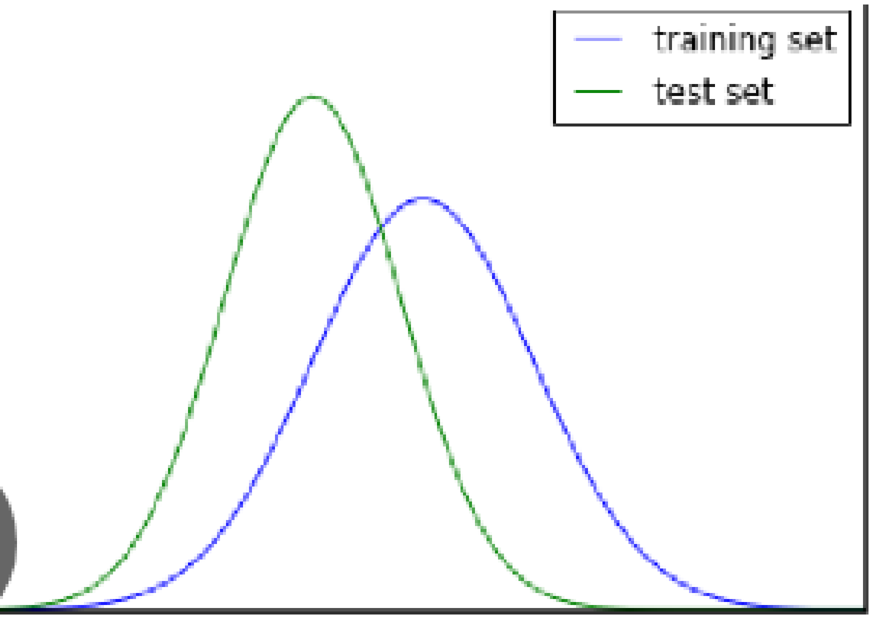
Training Challenge

- Training Deep Neural Networks is complicated by the fact that the **distribution of each layer's inputs changes during training**, as the parameters of the previous layers change**. This slows down the training by requiring lower learning rates and careful parameter initialization**, and makes it notoriously hard to train models with saturating nonlinearities.
- An **internal covariate shift occurs when there is a change in the input distribution to our network**. When the input distribution changes, hidden layers try to learn to adapt to the new distribution. **This slows down the training process. If a process slows down, it takes a long time to converge to a global minimum**

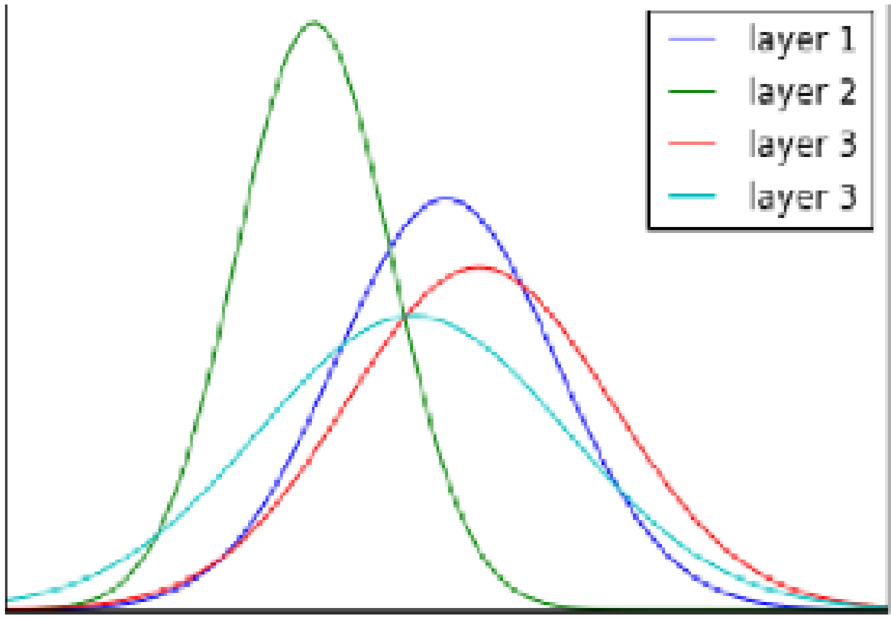
- Internal covariate shift
  - As the inputs flow through the network, their distributions deviate from unit Gaussian. In fact **the input distribution at a particular layer depends on the parameters of all the preceding layers**. The extent of deviation increases as the network becomes deeper.
  - Solution : Only normalizing the inputs to the network does not solve the problem. **We need a mechanism which normalizes the inputs of every single layer.**

- Images will have certain distribution as well. Using these images model will update its parameters.
- **New images will have a slightly different distribution from the previous images.** Now the model will change its parameters according to these new images. Hence the distribution of the hidden activation will also change. This change in hidden activation is known as an internal covariate shift.





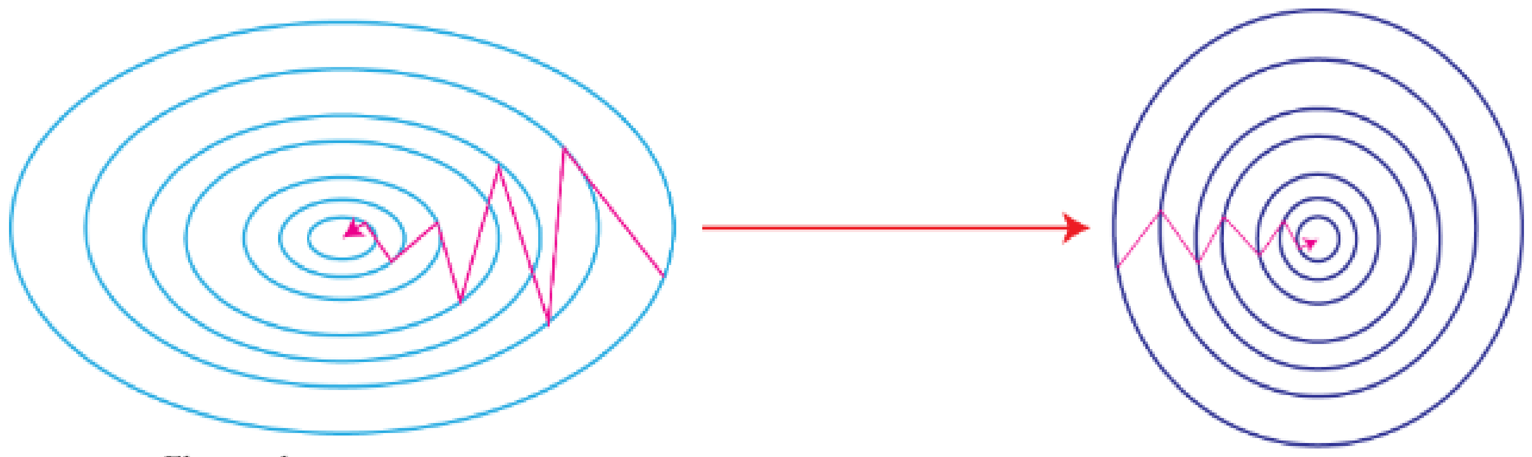
(a) Covariate shift



(b) Internal covariate shift

Normalization

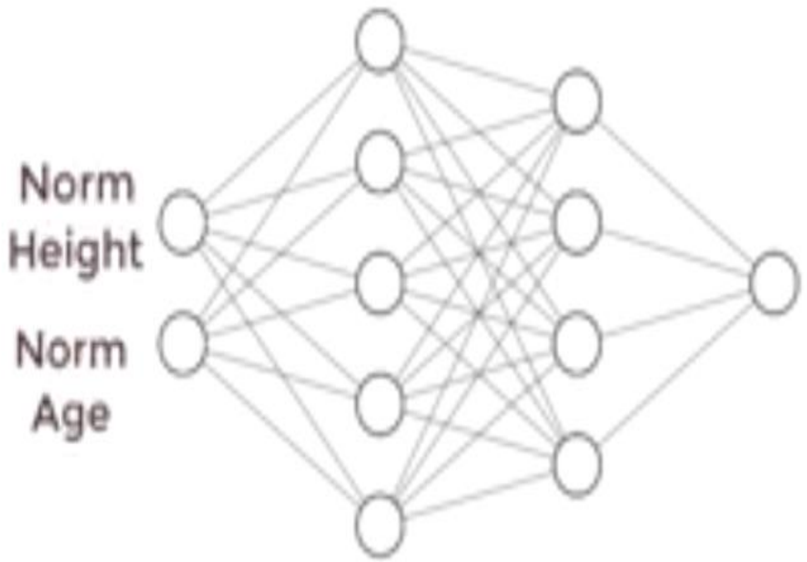
- **Normalization is a data pre-processing tool used to bring the numerical data to a common scale without distorting its shape.**
- If the inputs of a neural network are normalized before training, the process of training gets faster.
- **This is because with normal data, the contour of cost function tends to be more elongated and elliptical. The learning rate needs to be smaller and no. of steps for gradient descent is higher.** However, because of normalization, the contour gets more symmetrical and circular as shown in Fig. Therefore, a larger learning rate can be adopted and gradient will be faster



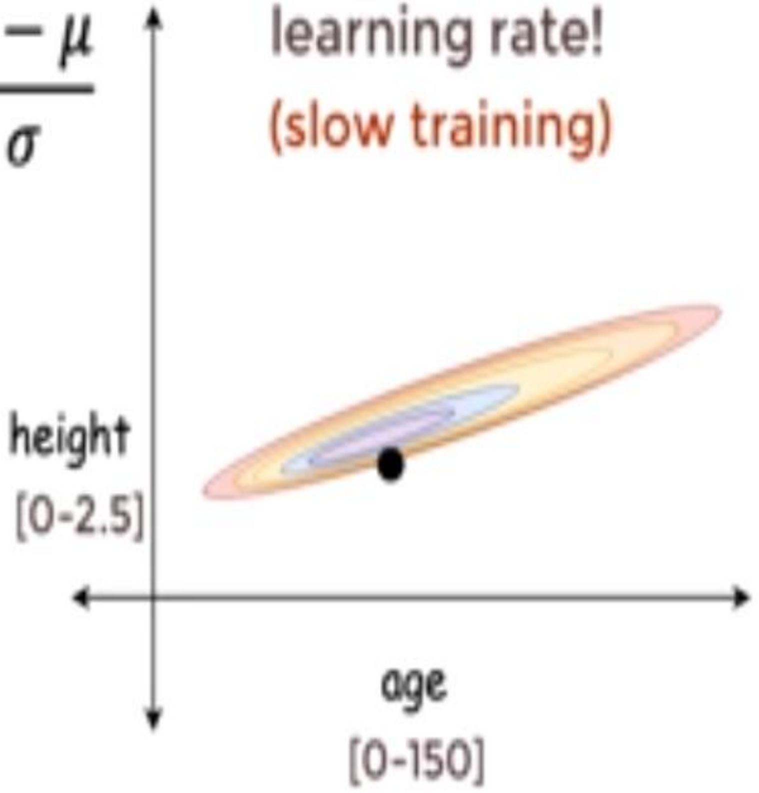
Why Batch Normalization?

1. Speeds up training

$$\hat{x} = \frac{x - \mu}{\sigma}$$

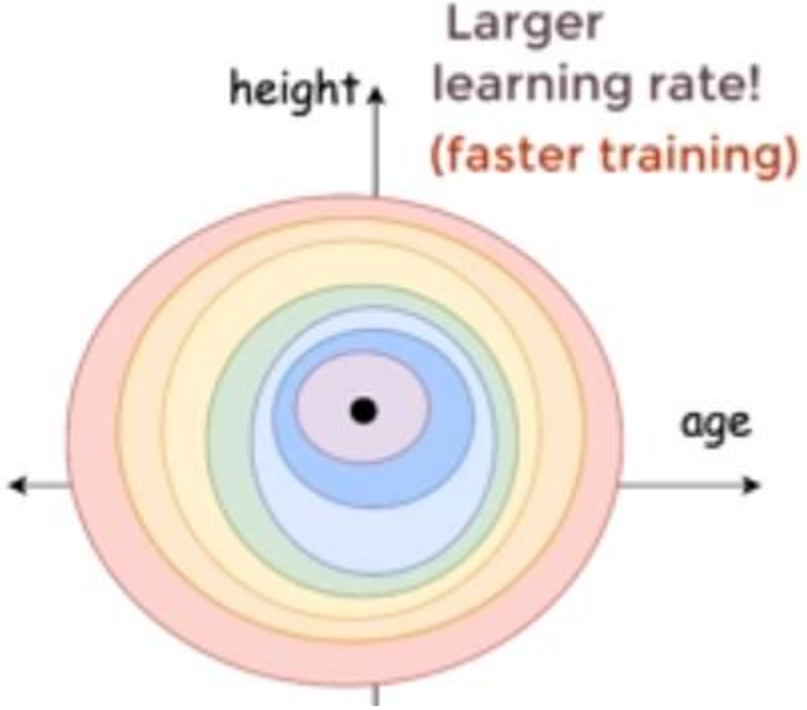


Need Small
learning rate!
(slow training)



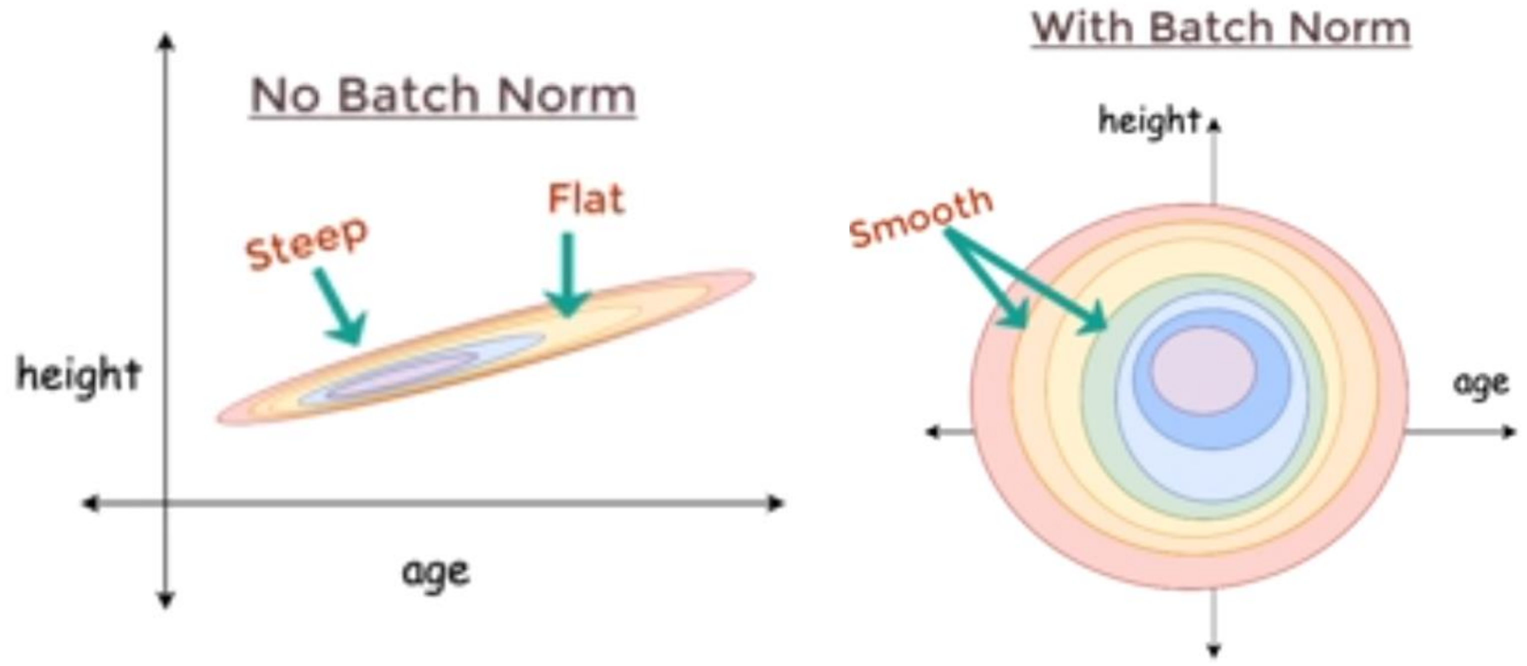
Why Batch Normalization?

1. Speeds up training



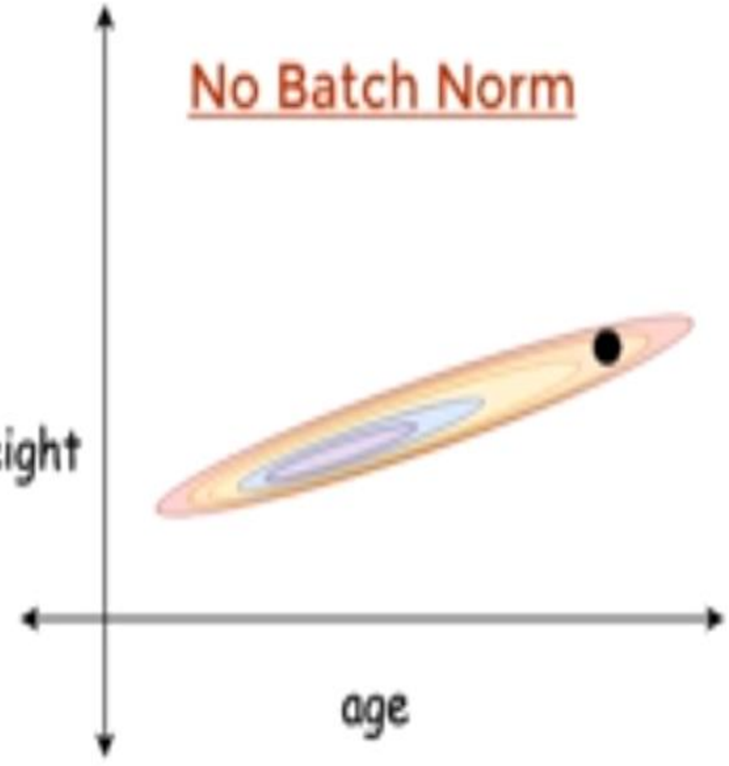
Why Batch Normalization?

1. Speeds up training

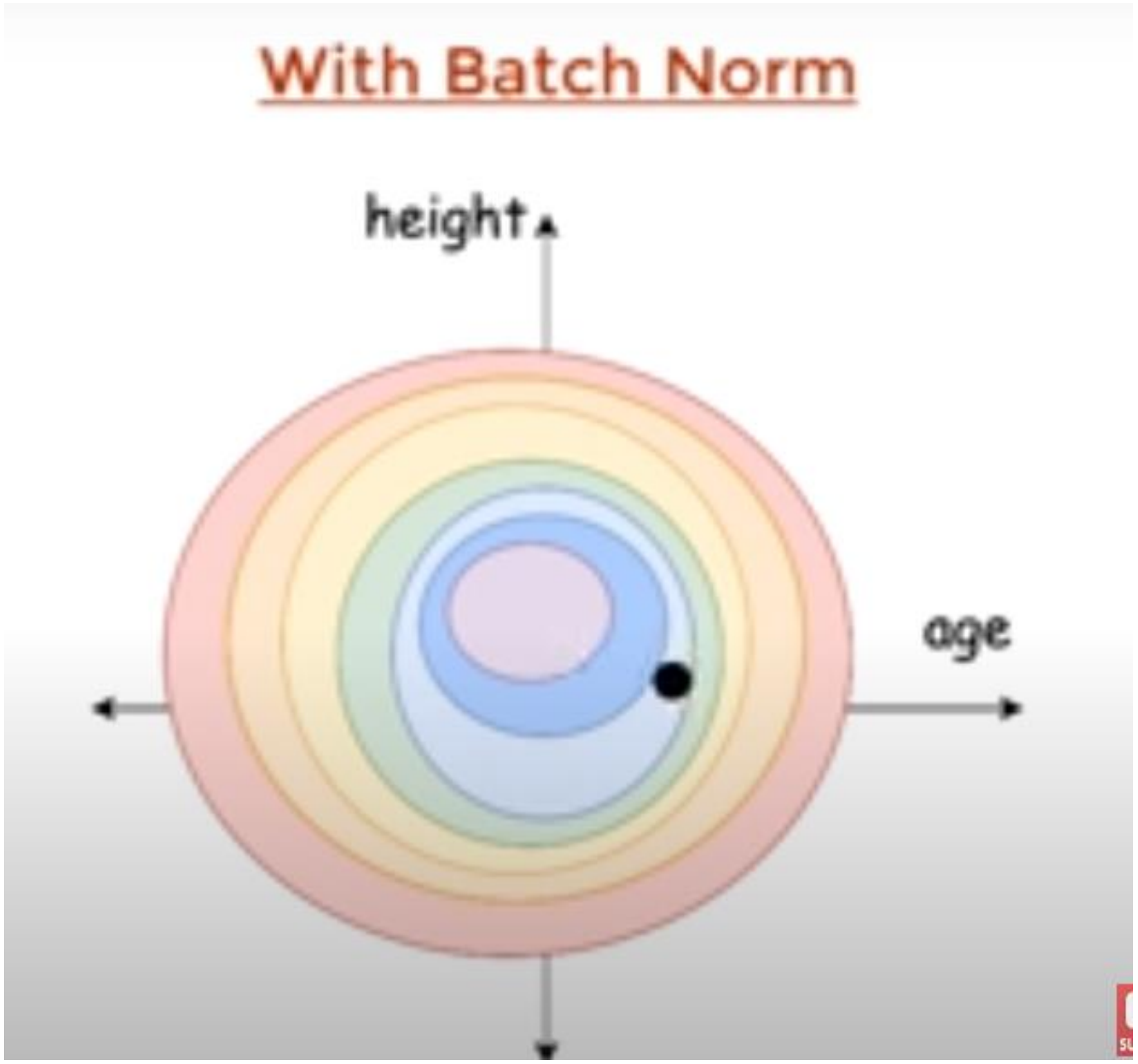


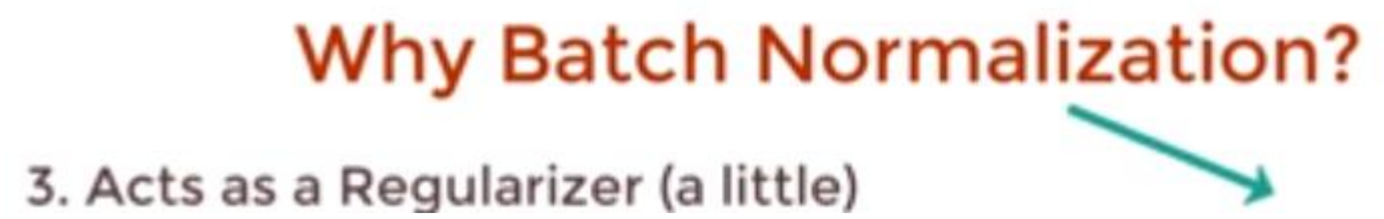
Why Batch Normalization?

2. Allows sub-optimal starts



1000 iterations till convergence


With Batch Norm
height
age



Why Batch Normalization?

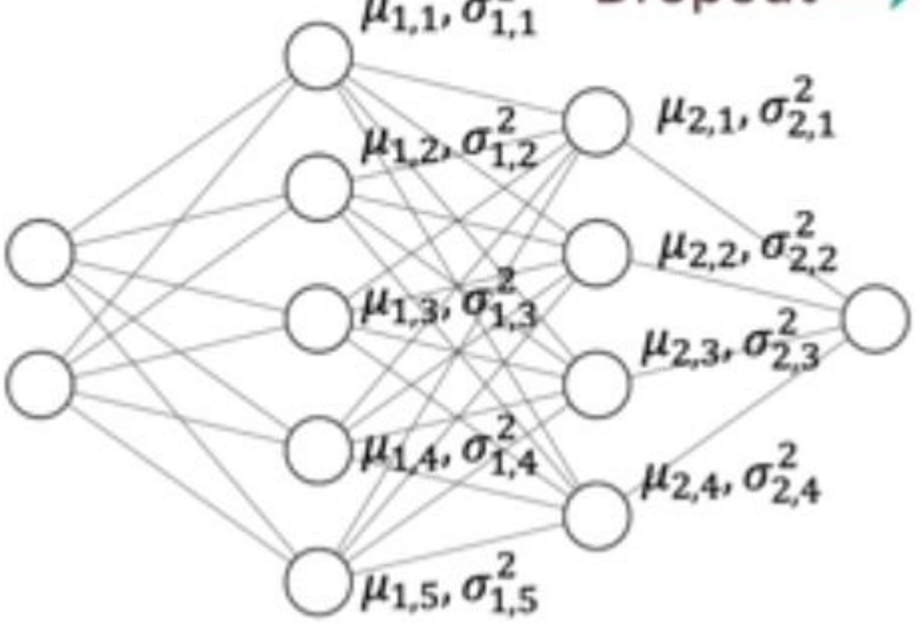
3. Acts as a Regularizer (a little)

- Batch normalization regularizes gradient from distraction to outliers and flows towards the common goal (by normalizing them) within a range of the mini-batch.
- Resulting in the acceleration of the learning process.

Why Batch Normalization?

3. Acts as a Regularizer (a little)

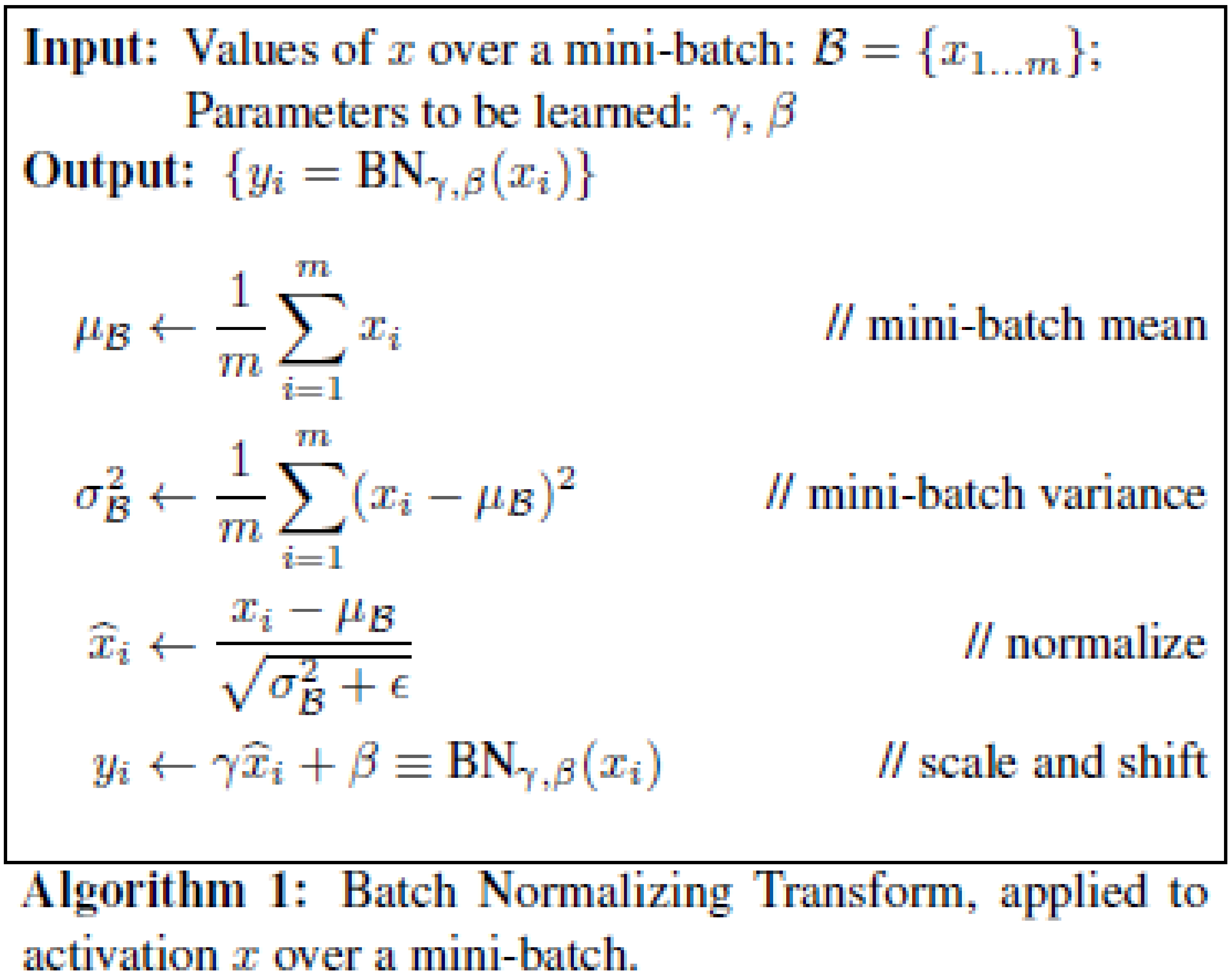
“Dropout” $\rightarrow$ “randomness”



$\mu_{1,1}, \sigma^2_{1,1}$
$\mu_{1,2}, \sigma^2_{1,2}$
$\mu_{1,3}, \sigma^2_{1,3}$
$\mu_{1,4}, \sigma^2_{1,4}$
$\mu_{1,5}, \sigma^2_{1,5}$
$\mu_{2,1}, \sigma^2_{2,1}$
$\mu_{2,2}, \sigma^2_{2,2}$
$\mu_{2,3}, \sigma^2_{2,3}$
$\mu_{2,4}, \sigma^2_{2,4}$

Batch Normalization

- **Batch Normalization attempts to normalize a batch of inputs before they are fed to a non-linear activation unit** (like ReLU, sigmoid, etc).
- The idea is to feed a **normalized input to an activation function so as to prevent it from entering into the saturated regime**.
- Batch normalization provides an elegant way of **reparametrizing almost any deep network. The reparametrization significantly reduces the problem of coordinating updates across many layers**.
- It does this scaling the output of the layer, specifically by standardizing the activations of each input variable per mini-batch, such as the activations of a node from the previous layer.
- This process is also called **“whitening” when applied to images in computer vision.**
- By **whitening** the inputs to each layer, we would take a step towards achieving the **fixed distributions of inputs that would remove the ill effects of the internal covariate shift**.



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

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

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

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

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

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

Batch Norm Details

$$\mu_B = \frac{1}{3}(4 + 7 + 5) = 5.33$$

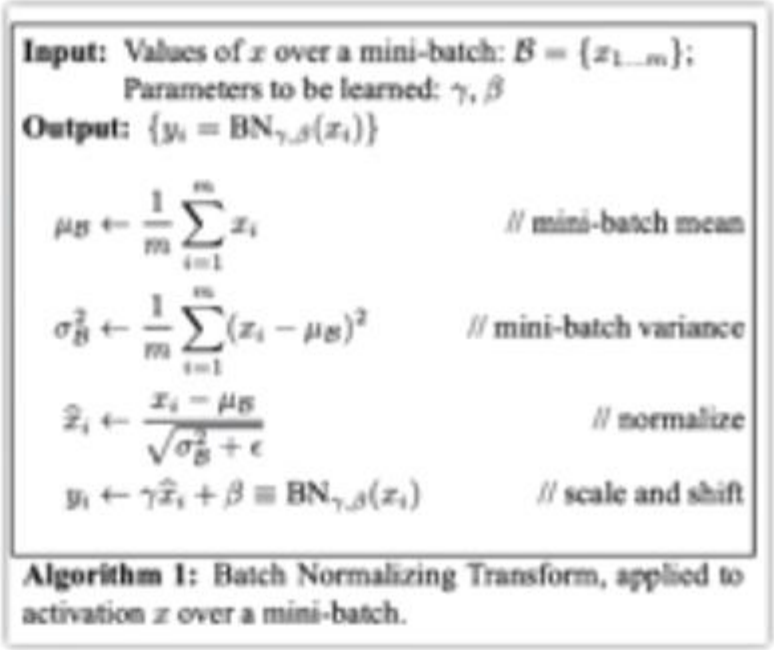
$$\sigma_B^2 = \frac{1}{3}(4 - 5.33)^2 + (5 - 5.33)^2 + (7 - 5.33)^2$$

$$= 1.5555$$

$$\widehat{x_1} = \frac{4 - 5.33}{\sqrt{1.55 + \epsilon}} \approx -1.068 \searrow$$

$$\widehat{x_2} = \frac{7 - 5.33}{\sqrt{1.55 + \epsilon}} \approx 1.341 \rightarrow \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

$$\widehat{x_3} = \frac{5 - 5.33}{\sqrt{1.55 + \epsilon}} \approx -0.265 \nearrow$$



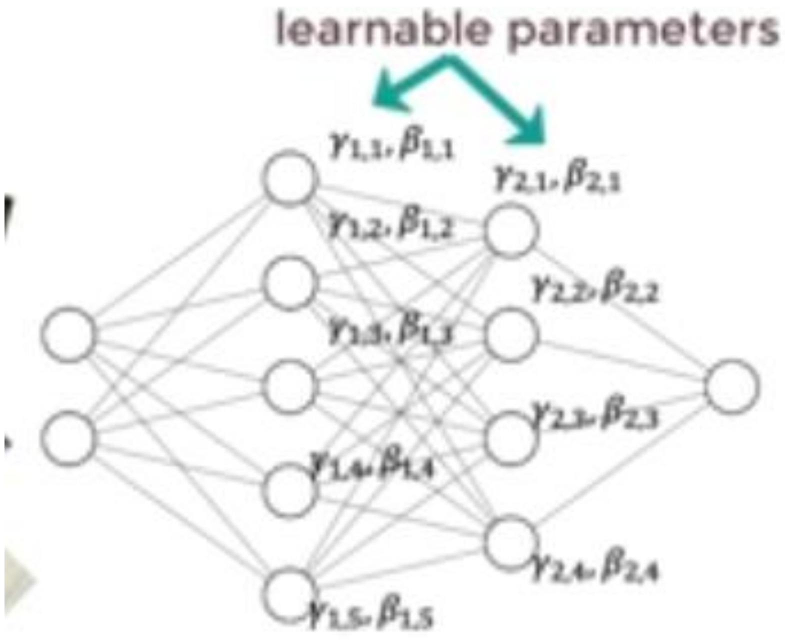
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

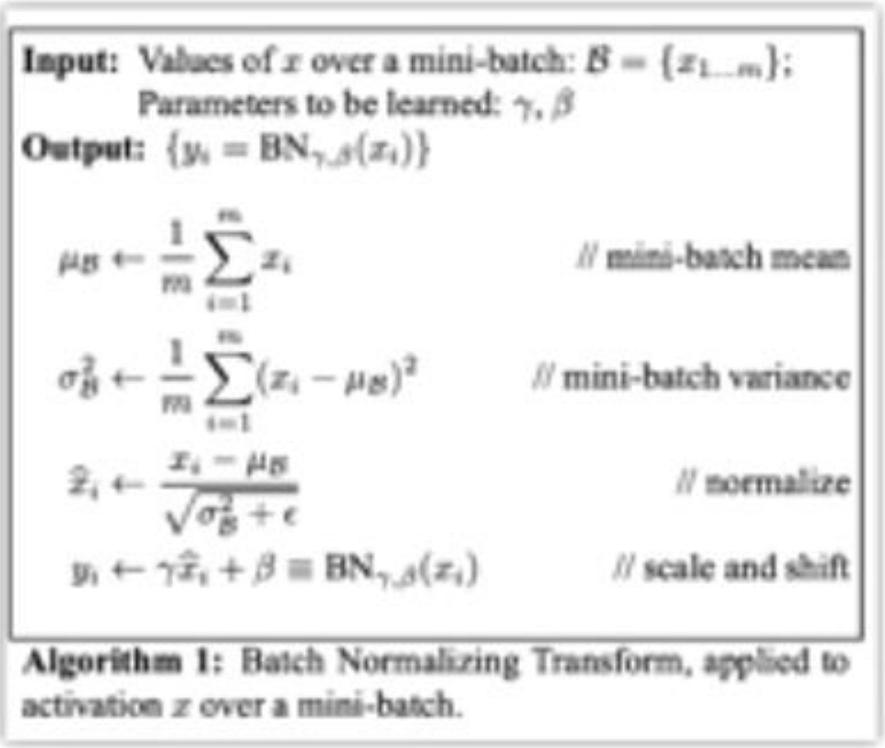
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

```
$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$    // mini-batch mean
$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2$    // mini-batch variance
$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}$    // normalize
$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i)$    // scale and shift
```

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

Batch Norm Details


learnable parameters
$\gamma_{1,1}, \beta_{1,1}$
$\gamma_{2,1}, \beta_{2,1}$
$\gamma_{1,2}, \beta_{1,2}$
$\gamma_{2,2}, \beta_{2,2}$
$\gamma_{1,3}, \beta_{1,3}$
$\gamma_{2,3}, \beta_{2,3}$
$\gamma_{1,4}, \beta_{1,4}$
$\gamma_{2,4}, \beta_{2,4}$
$\gamma_{1,5}, \beta_{1,5}$



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

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

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

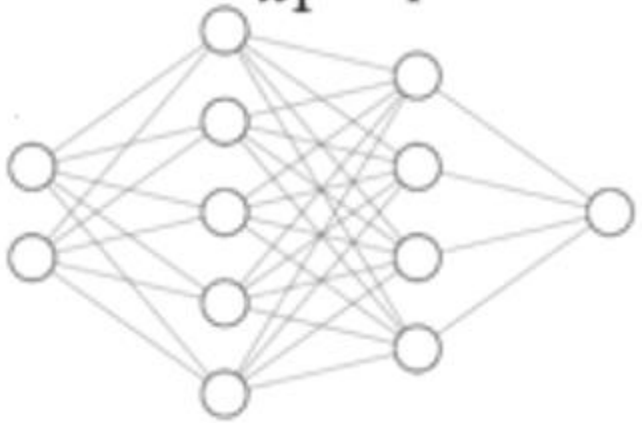
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

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

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

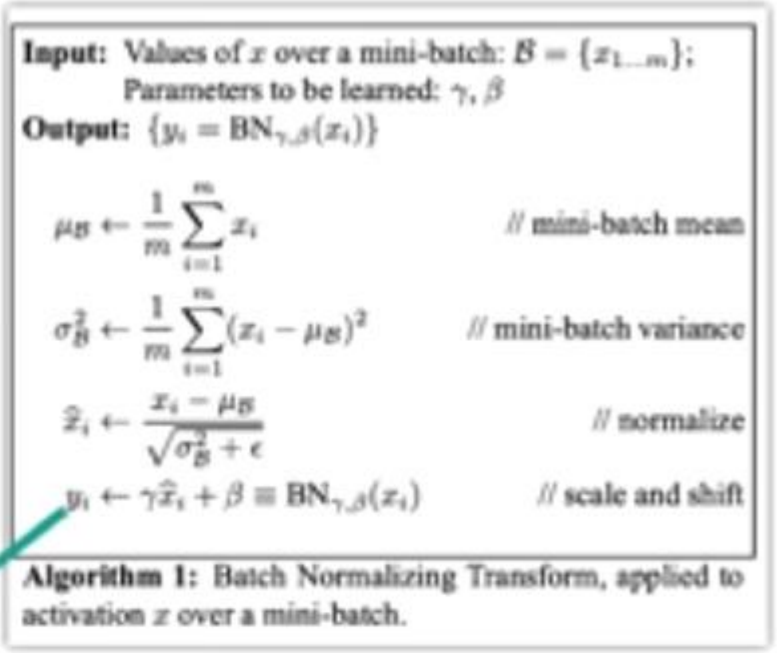
Batch Norm Details

$x_1 = 4$



$$\widehat{x_1} = \frac{4 - 5.33}{\sqrt{1.55 + \epsilon}} \approx -1.068$$

$$y_1 = \gamma_{1,1}(-1.068) + \beta_{1,1}$$



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

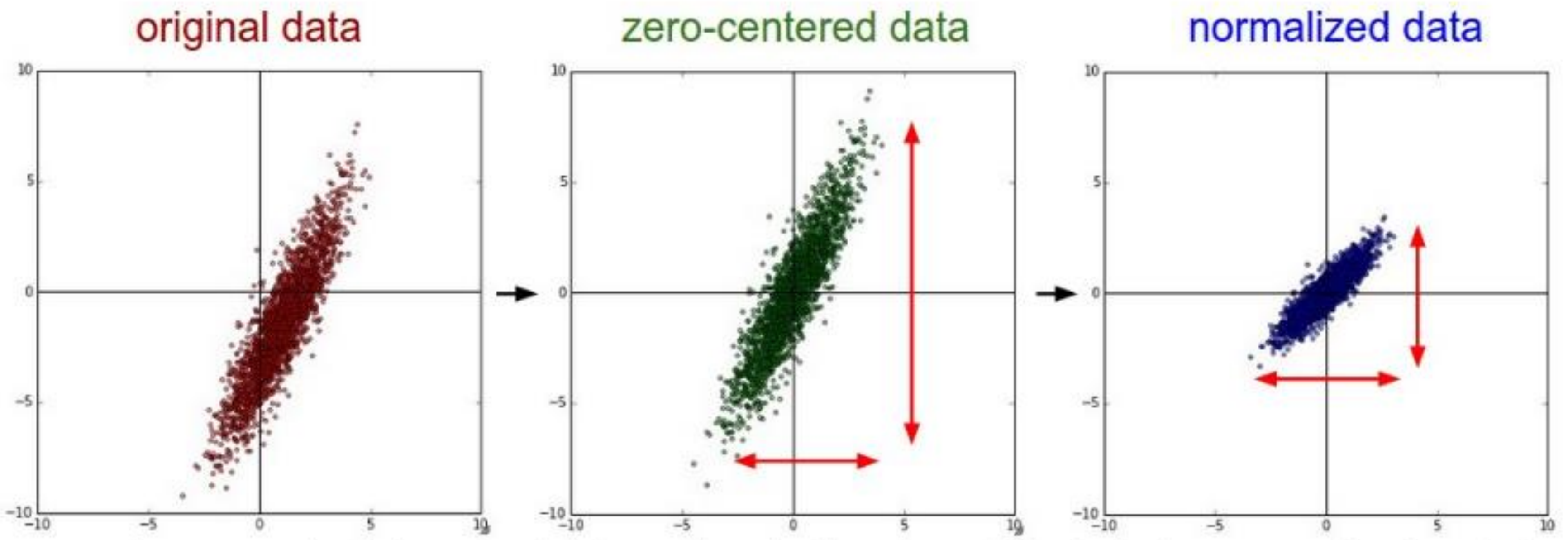
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$ // mini-batch mean

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

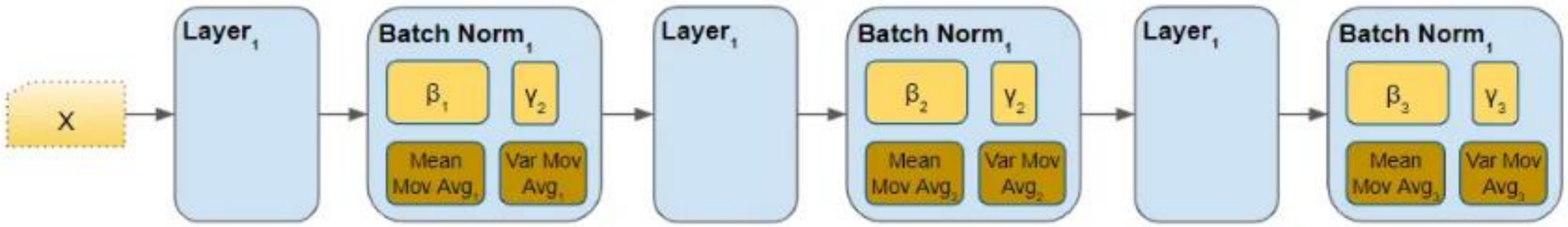
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$ // normalize

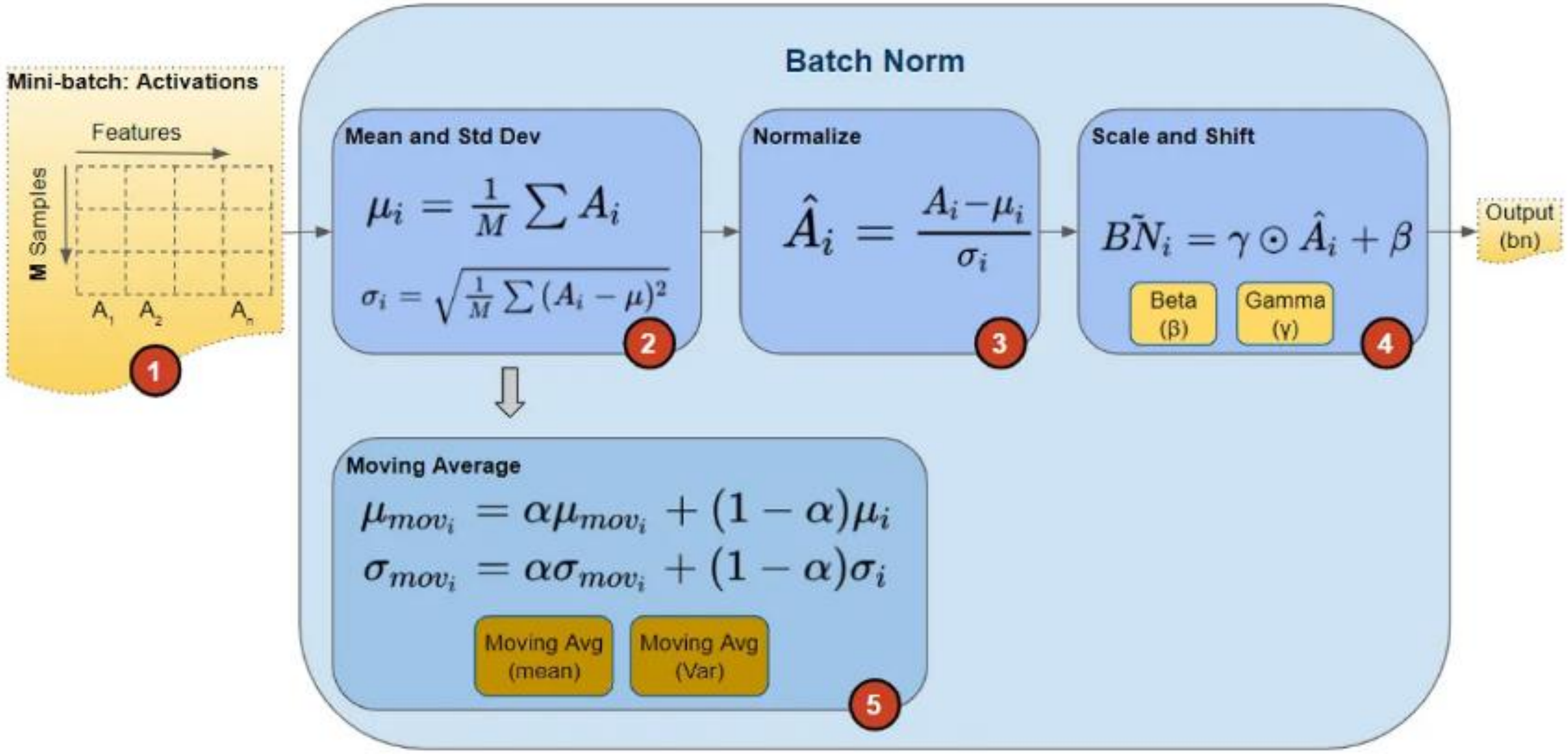
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$ // scale and shift

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

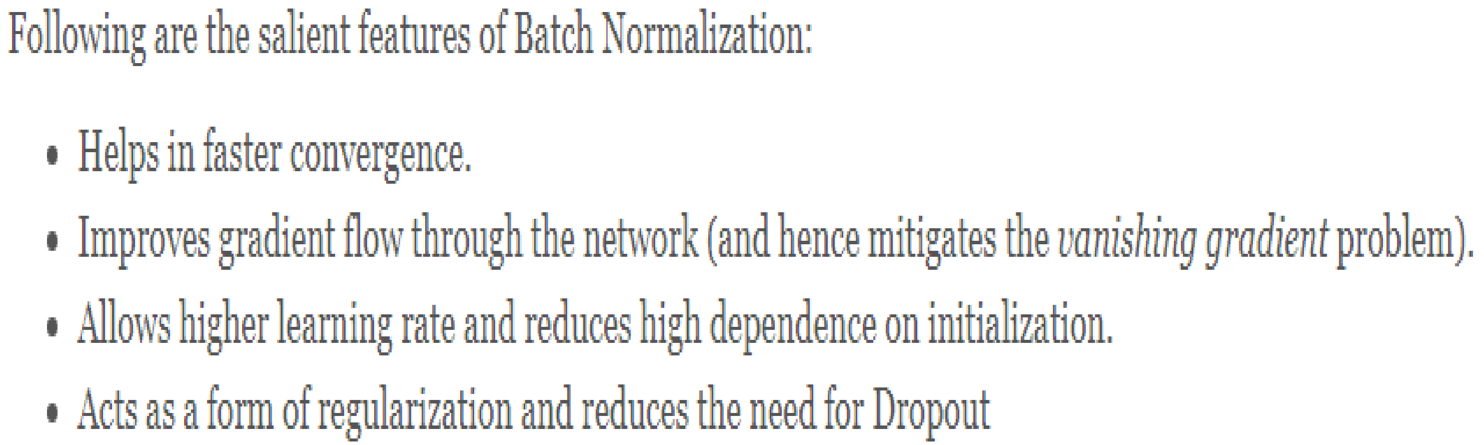


Common data preprocessing pipeline. **Left**: Original toy, 2-dimensional input data. **Middle**: The data is zero-centered by subtracting the mean in each dimension. The data cloud is now centered around the origin. **Right**: Each dimension is additionally scaled by its standard deviation. The red lines indicate the extent of the data - they are of unequal length in the middle, but of equal length on the right.


X
Layer$_1$
Batch Norm$_1$
β_1
γ_2
Mean Mov Avg$_1$
Var Mov Avg$_1$
Layer$_1$
Batch Norm$_1$
β_2
γ_2
Mean Mov Avg$_2$
Var Mov Avg$_2$
Layer$_1$
Batch Norm$_1$
β_3
γ_3
Mean Mov Avg$_3$
Var Mov Avg$_3$


Mini-batch: Activations
Features
M Samples
A_1 A_2 A_n
1
Batch Norm
Mean and Std Dev
$\mu_i = \frac{1}{M}\sum A_i$
$\sigma_i = \sqrt{\frac{1}{M}\sum (A_i - \mu)^2}$
2
Normalize
$\hat{A}_i = \frac{A_i - \mu_i}{\sigma_i}$
3
Scale and Shift
$B\tilde{N}_i = \gamma \odot \hat{A}_i + \beta$
Beta (β)
Gamma (γ)
4
Output (bn)
Moving Average
$\mu_{mov_i} = \alpha\mu_{mov_i} + (1-\alpha)\mu_i$
$\sigma_{mov_i} = \alpha\sigma_{mov_i} + (1-\alpha)\sigma_i$
Moving Avg (mean)
Moving Avg (Var)
5

- Batch normalization **acts to standardize only the mean and variance of each unit in order to stabilize learning**
- Batch normalization can have a dramatic effect on **optimization performance, especially for convolutional networks and networks with sigmoidal nonlinearities.**
- BatchNorm impacts network training in a fundamental way: **it makes the landscape of the corresponding optimization problem be significantly more smooth. This ensures, in particular, that the gradients are more predictive and thus allow for use of larger range of learning rates and faster network convergence.**


Following are the salient features of Batch Normalization:

- Helps in faster convergence.
- Improves gradient flow through the network (and hence mitigates the *vanishing gradient* problem).
- Allows higher learning rate and reduces high dependence on initialization.
- Acts as a form of regularization and reduces the need for Dropout

Thank you!