Preventing Overfitting Problems

Thang Vu

Methods

- Weight Decay
- Early Stopping
- Dropout
- Batch Normalization

Weight Decay

- It is also well known as ‚Regularization' (L2)
- New cost function to be minimized

$$C'(\theta) = C(\theta) + \lambda \frac{1}{2} \|\theta\|^2$$ → Regularization term

- Original cost to minimize
  (e.g. cross entropy)

$$\theta = W^1, W^2, \ldots$$

Weight Decay

- It is also well known as ‚Regularization' (L2)
- New cost function to be minimized

$$C'(\theta) = C(\theta) + \lambda \frac{1}{2}\|\theta\|^2 \quad \text{Gradient:} \quad \frac{\partial C'}{\partial w} = \frac{\partial C}{\partial w} + \lambda w$$

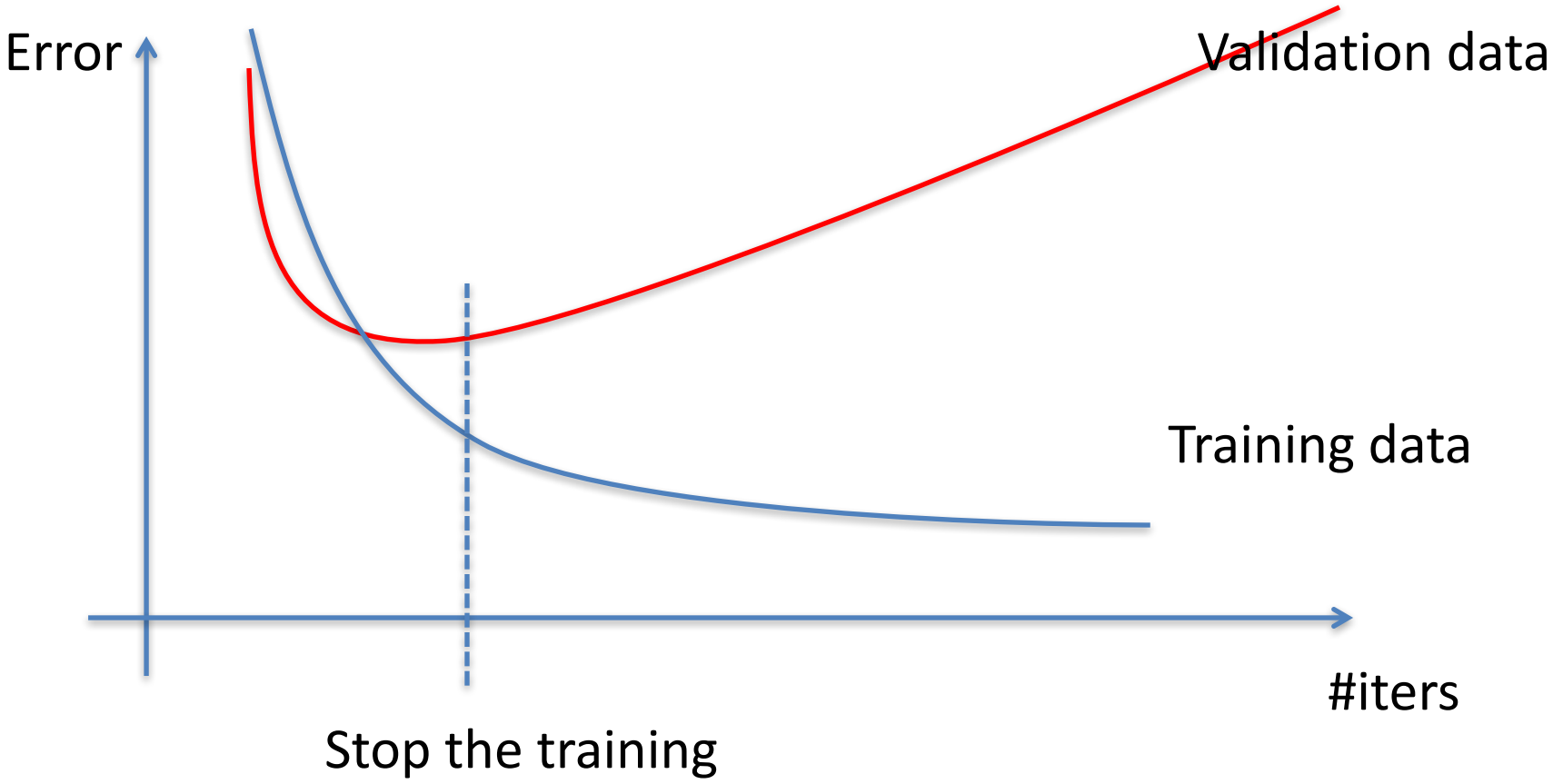
- Update:

$$w^{t+1} \leftarrow w^t - \eta \frac{\partial C'}{\partial w} = w^t - \eta\left(\frac{\partial C}{\partial w} + \lambda w^t\right)$$

$$= (1 - \eta\lambda) w^t - \eta \frac{\partial C}{\partial w}$$

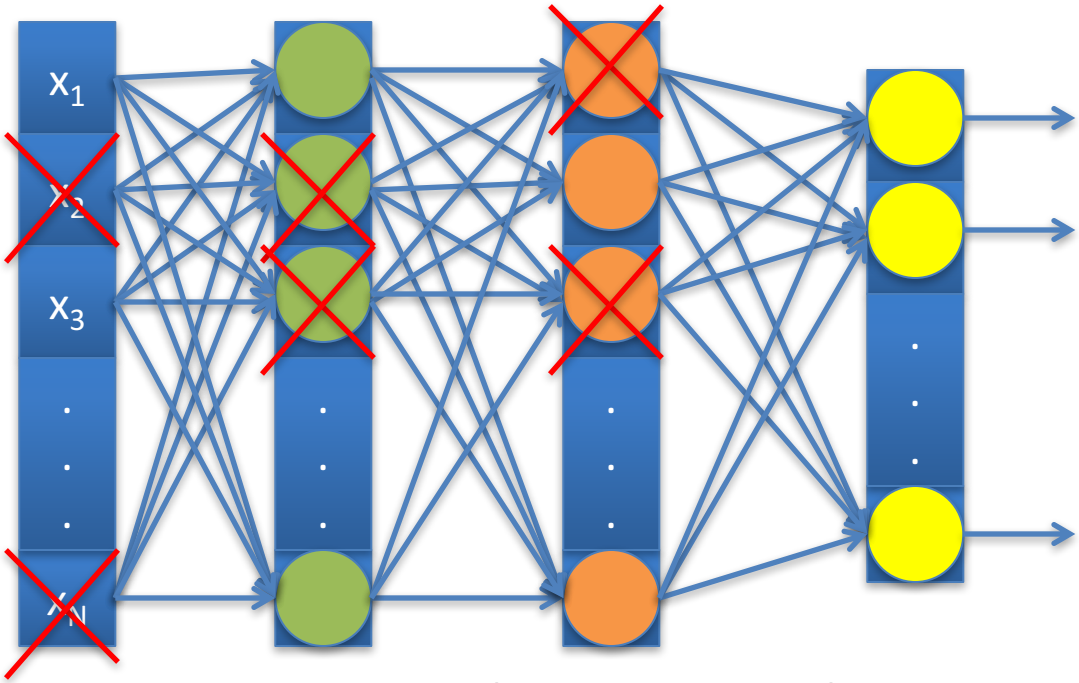
smaller and smaller

Early Stopping



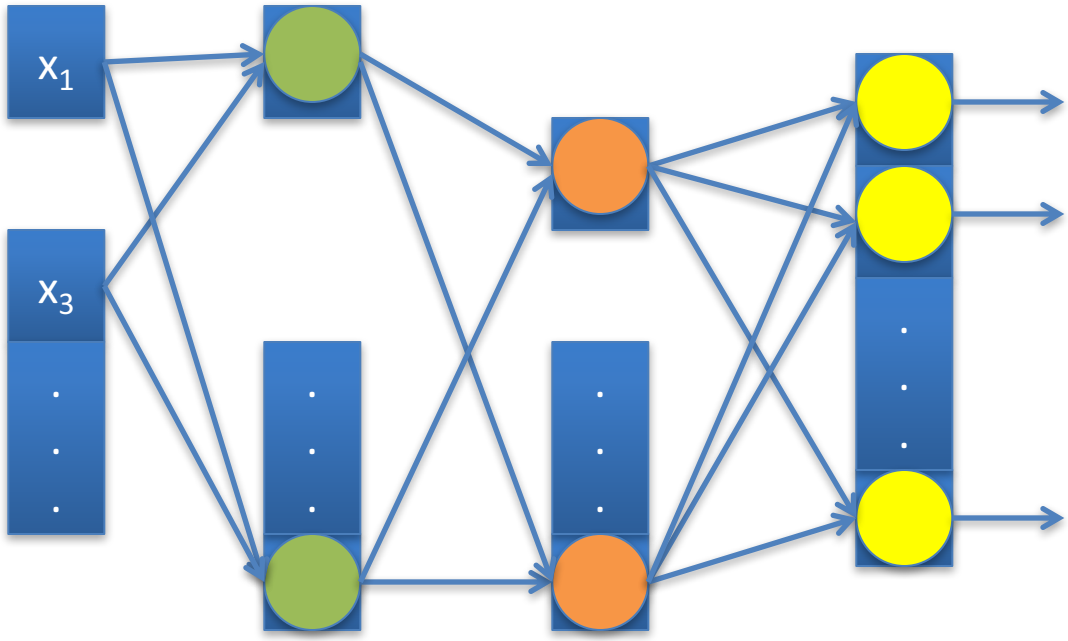
Dropout

Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014



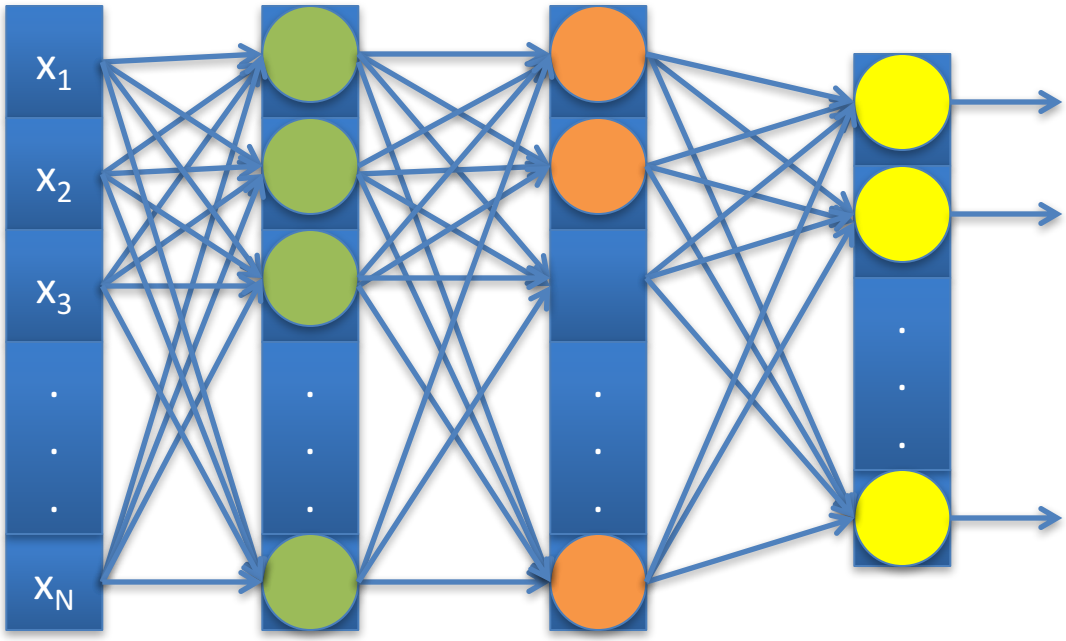
- In each iteration: Each neuron has p% to dropout during training

Dropout



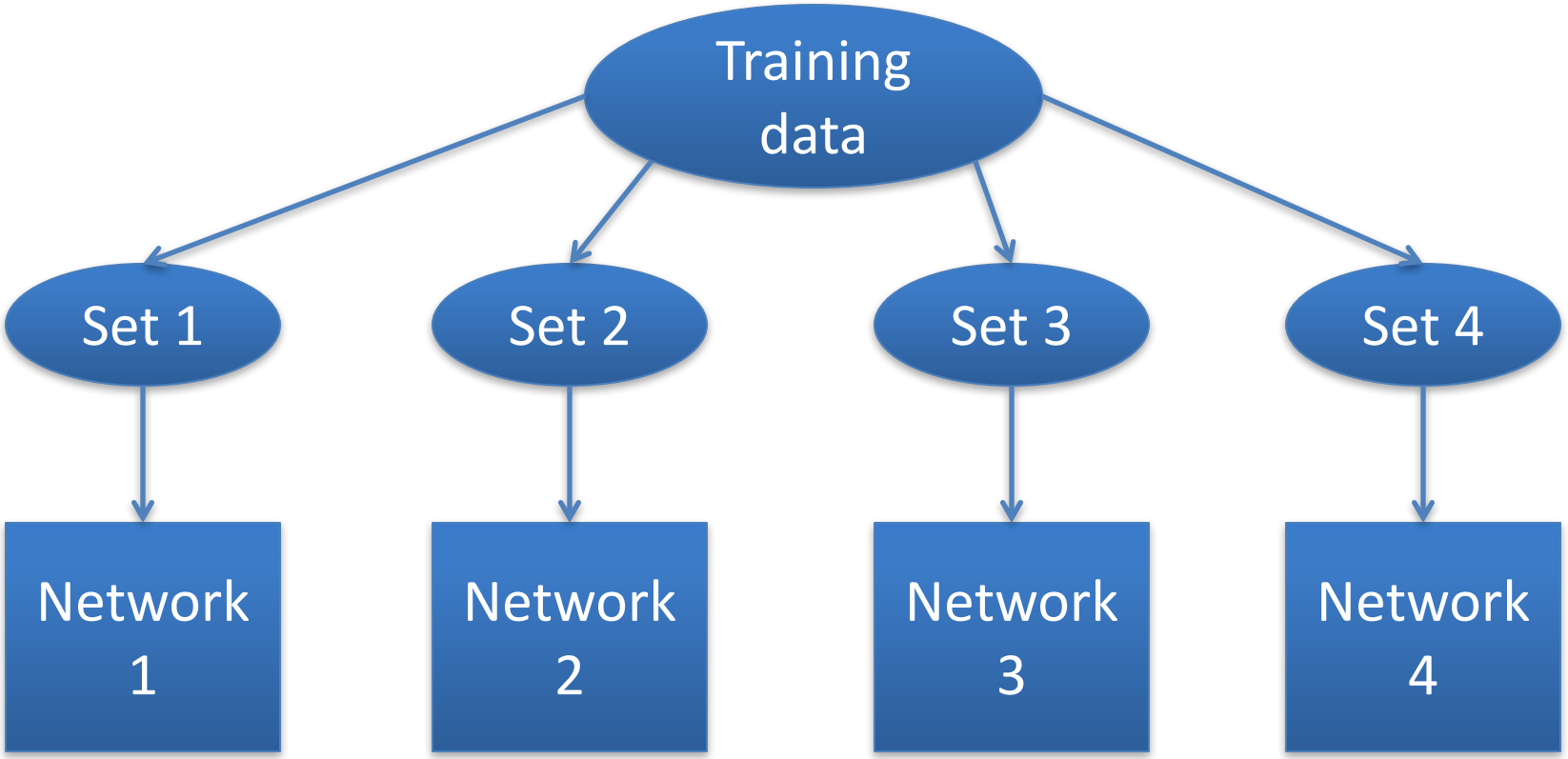
- In each iteration: Each neuron has p% to dropout during training, i.e. network structure is changed
- Only update the existing networks

Dropout



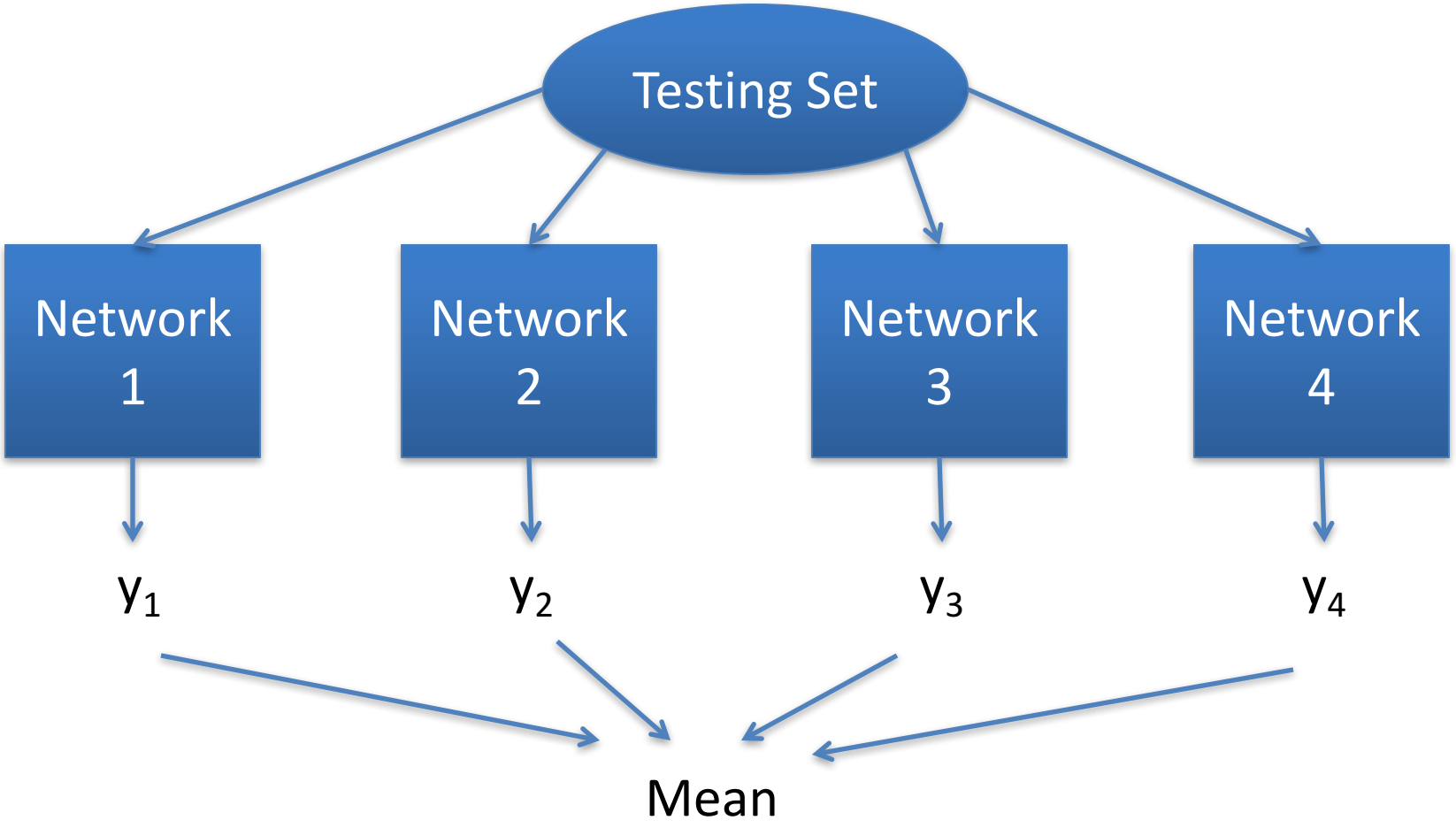
- No dropout during testing: Scale the weight with 1-p% if the dropout weight is p% during training

Dropout - Ensemble



- Train with a bunch of networks with different structures

Dropout - Ensemble

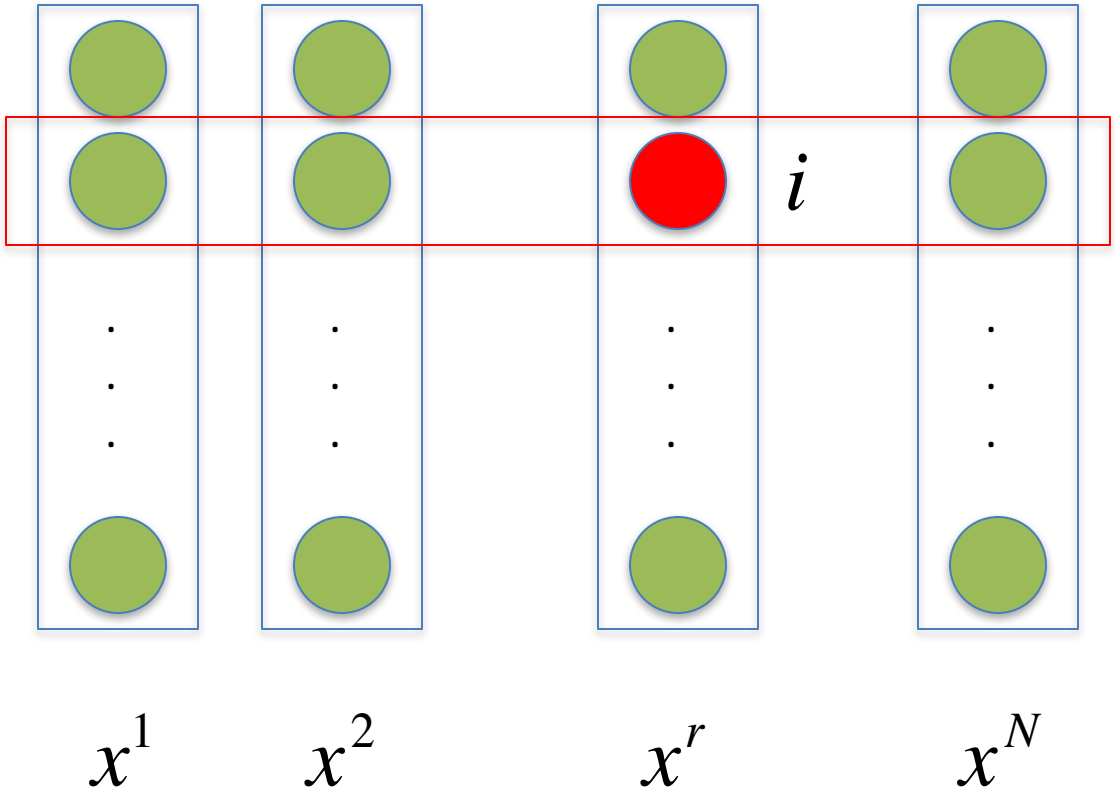


Batch Normalization

- Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015

- Normalize the input for each of the sub networks
  - Mean = 0
  - Variance = 1

Data Processing

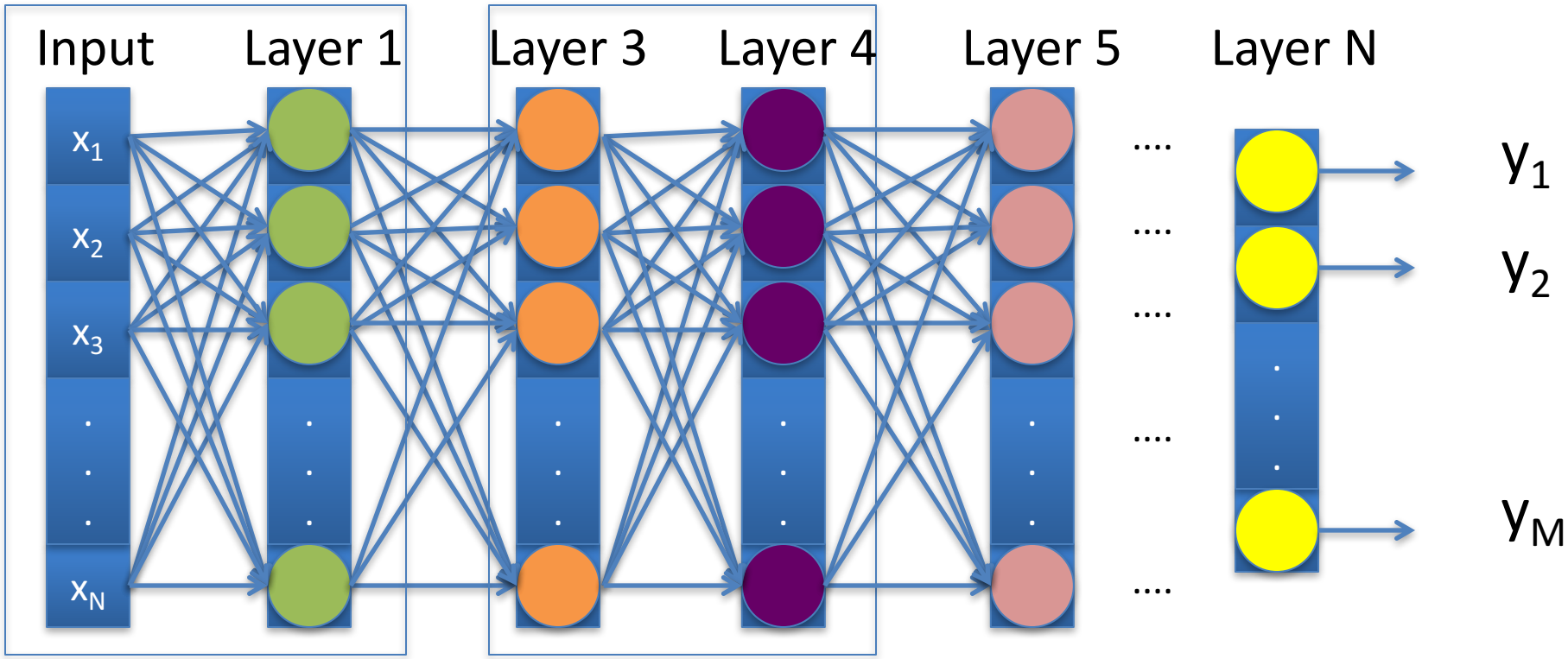


- For each of the dimension compute the mean m_i σ_i

$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

- Mean = 0
- Variance = 1

Multilayer Perceptron



Batch Normalization

- Normalization of the input
- BN normalizes each layer, for each mini-batch

$$x \quad \Rightarrow \quad \hat{x} \leftarrow \frac{x-\mu}{\sigma} \quad \Rightarrow \quad y = \gamma\hat{x} + \beta$$

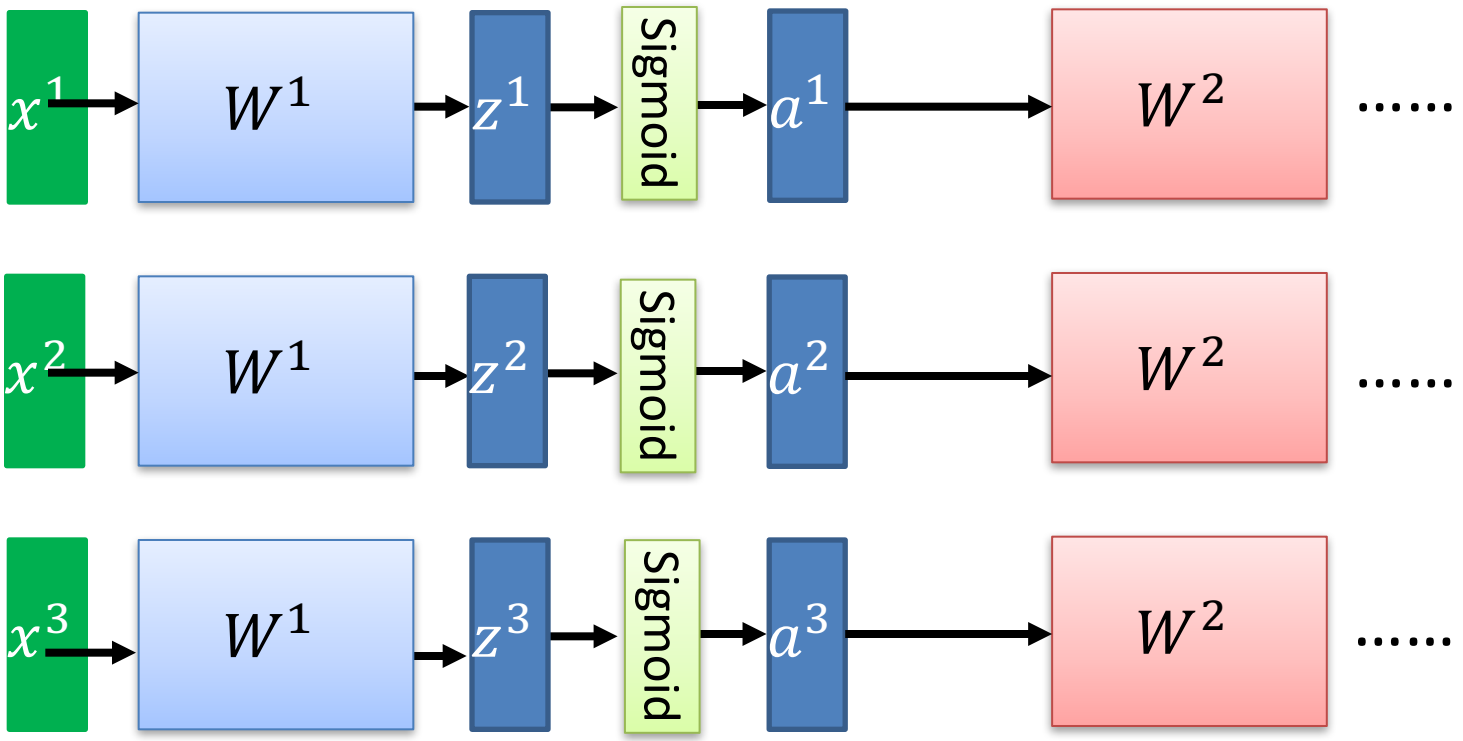
μ Mean of x in a mini-batch

σ Std. of x in a mini-batch

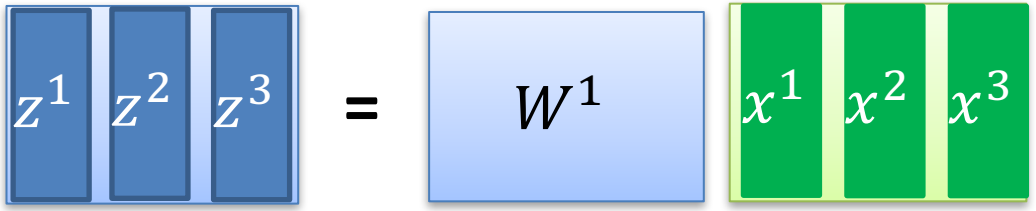
γ Scale, parameter to learn

β Shift, parameter to learn

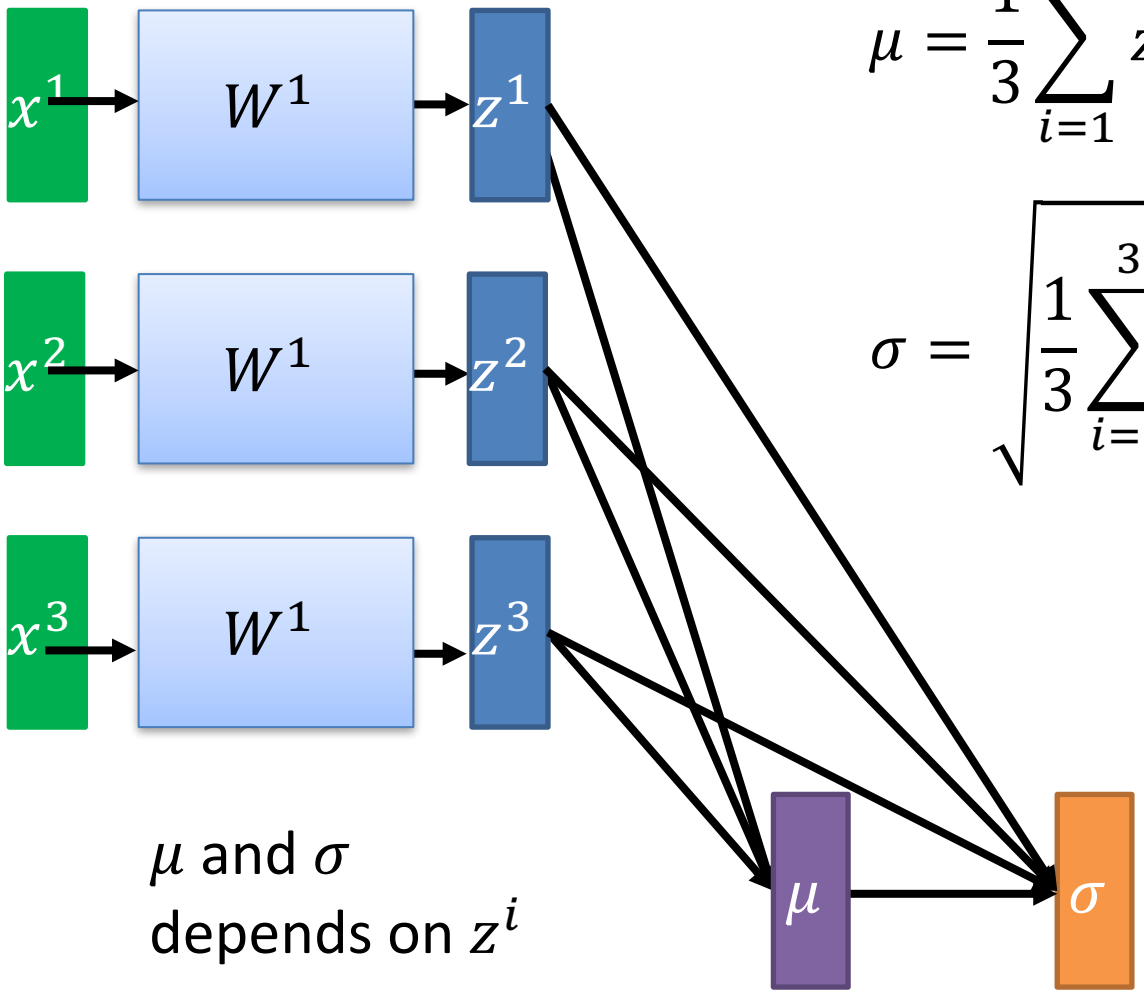
Batch Normalization



Batch



Batch Normalization

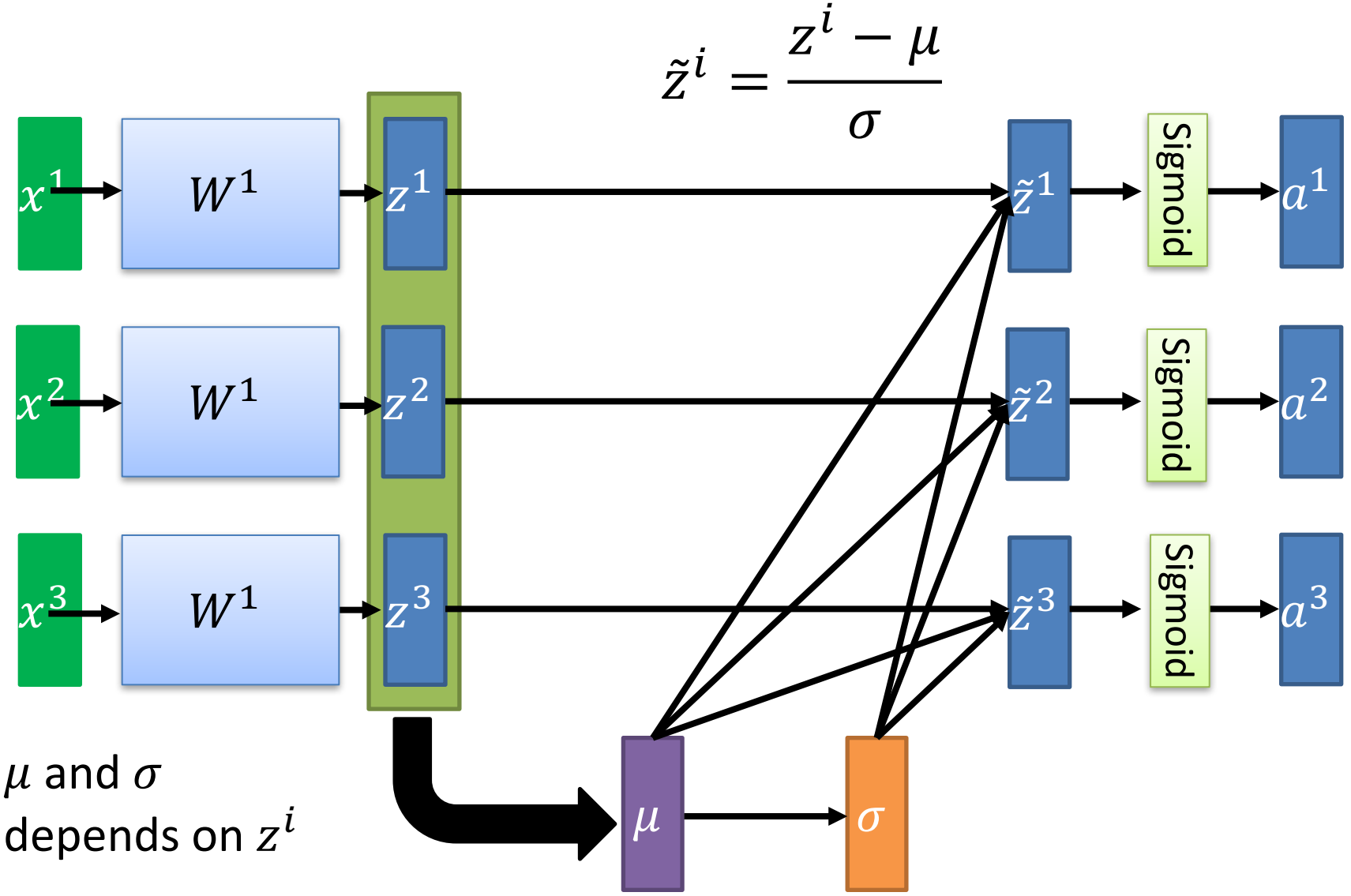


$$\mu = \frac{1}{3}\sum_{i=1}^{3} z^i$$

$$\sigma = \sqrt{\frac{1}{3}\sum_{i=1}^{3}(z^i - \mu)^2}$$

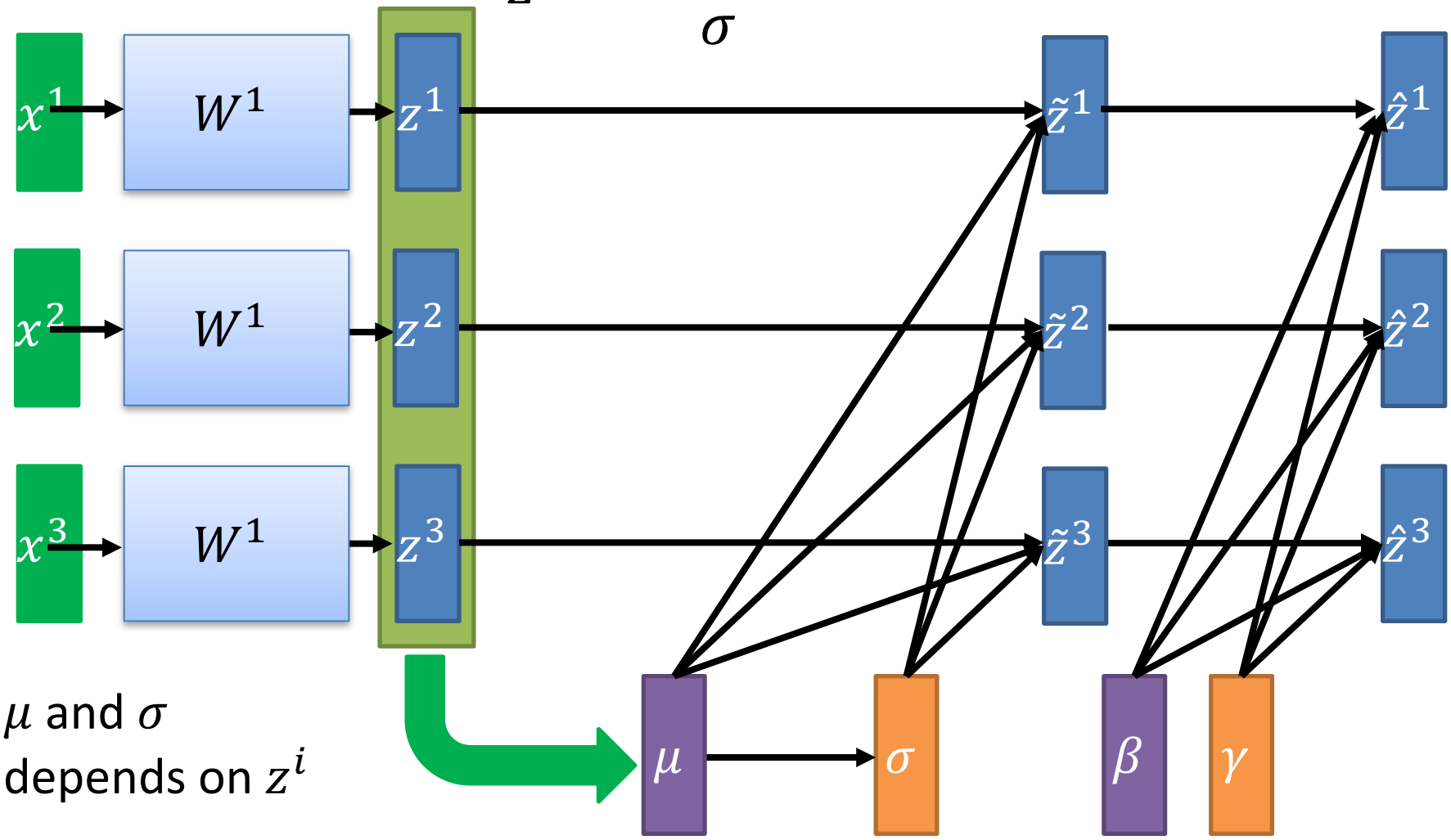
μ and σ depends on z^i

Batch Normalization



Batch Normalization

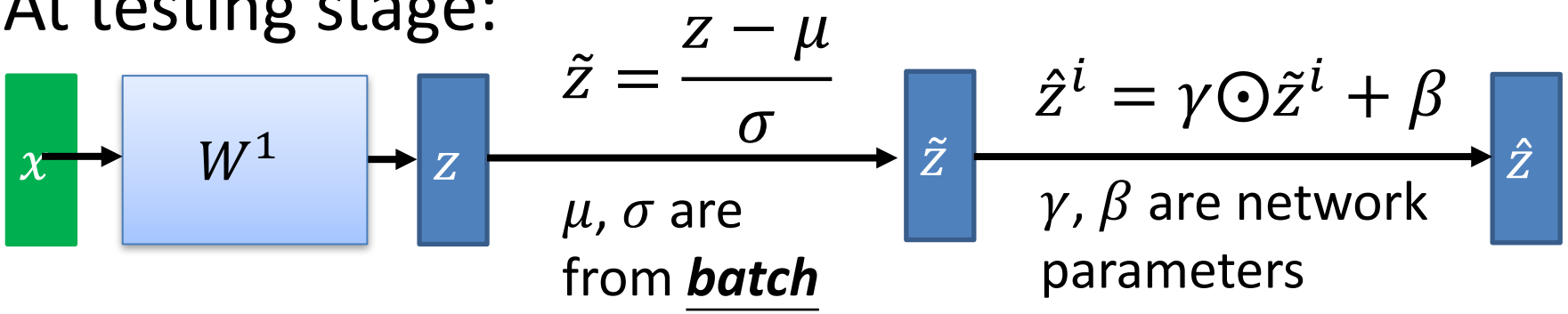
$$\tilde{z}^i = \frac{z^i - \mu}{\sigma} \qquad \hat{z}^i = \gamma \odot \tilde{z}^i + \beta$$



μ and σ depends on z^i

Batch normalization

- At testing stage:

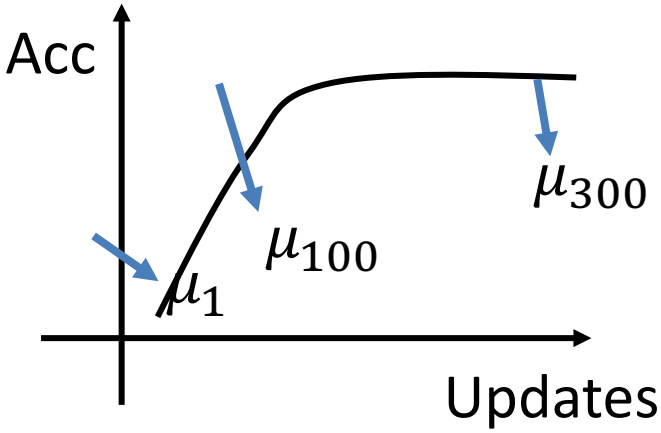


We do not have ***batch*** at testing stage.

Ideal solution:

Computing μ and σ using the whole training dataset.

Practical solution:



Computing the moving average of μ and σ of the batches during training.

Re-tune Your Models

Simply using BN will not be effective

Re-tune your model:

- Increase the learning rate
- Remove dropout
- Reduce the weight decay regularization term
- Accelerate the learning rate decay