Regularization

Seyoung Yun

- http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

- N. Srivstava et al, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting” http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

- Sergey Ioffe and Christian Szegedy“Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift” https://arxiv.org/abs/1502.03167

- C. Zhang et al “Understanding deep learning requires rethinking generalization” https://arxiv.org/abs/1611.03530

Regularization

training data

model

loss function

optimization

- **“A regularizer is anything that hurts the training process”**
  C. Zhung at ICLR2017 (https://www.youtube.com/watch?v=kCj51pTQPKI)

    - data augmentation

    - **weight decay - with an additional cost**

    - **dropout - by adding random noise**

bias & variance

Linear Regression

- RSS: cost of linear regression

$$\mathcal{L}(w,b) = \sum_{i=1}^{m} (y^{(i)} - \underbrace{w^\top x^{(i)} - b}_{\text{linear function}})^2$$

- regularizer

$$\boxed{\mathcal{L}(w,b) = \frac{1}{m}\sum_{i=1}^{m} (y^{(i)} - w^\top x^{(i)} - b)^2 + \frac{\lambda}{2m}\|w\|_2^2}$$

or

$$\mathcal{L}(w,b) = \frac{1}{m}\sum_{i=1}^{m} (y^{(i)} - w^\top x^{(i)} - b)^2 + \frac{\lambda}{2m}\|w\|_1$$

Weight Decay

$$J(W^{[1]}, b^{[1]}, \ldots, W^{[L]}, b^{[L]})$$

$$= \frac{1}{m} \sum_{i=1}^{m} l(\hat{Y}^{(i)}, Y^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^{L} \|W^{[l]}\|_F^2$$

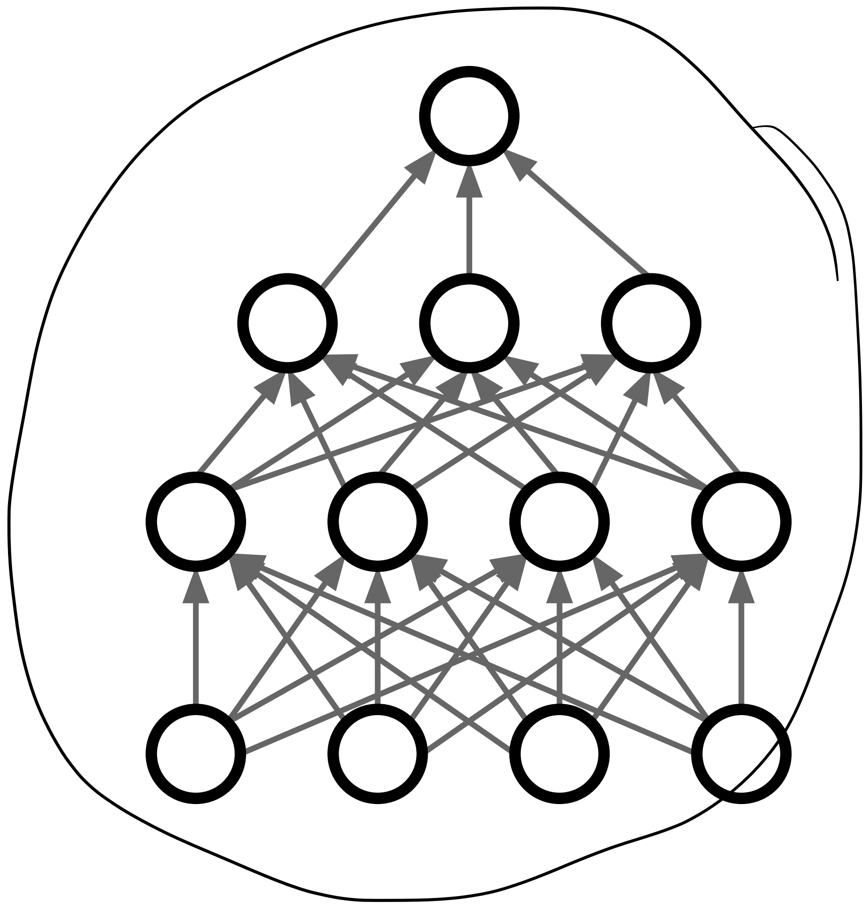
$$\|W^{[l]}\|_F^2 = \sum \sum (W_{jk}^{[l]})^2$$

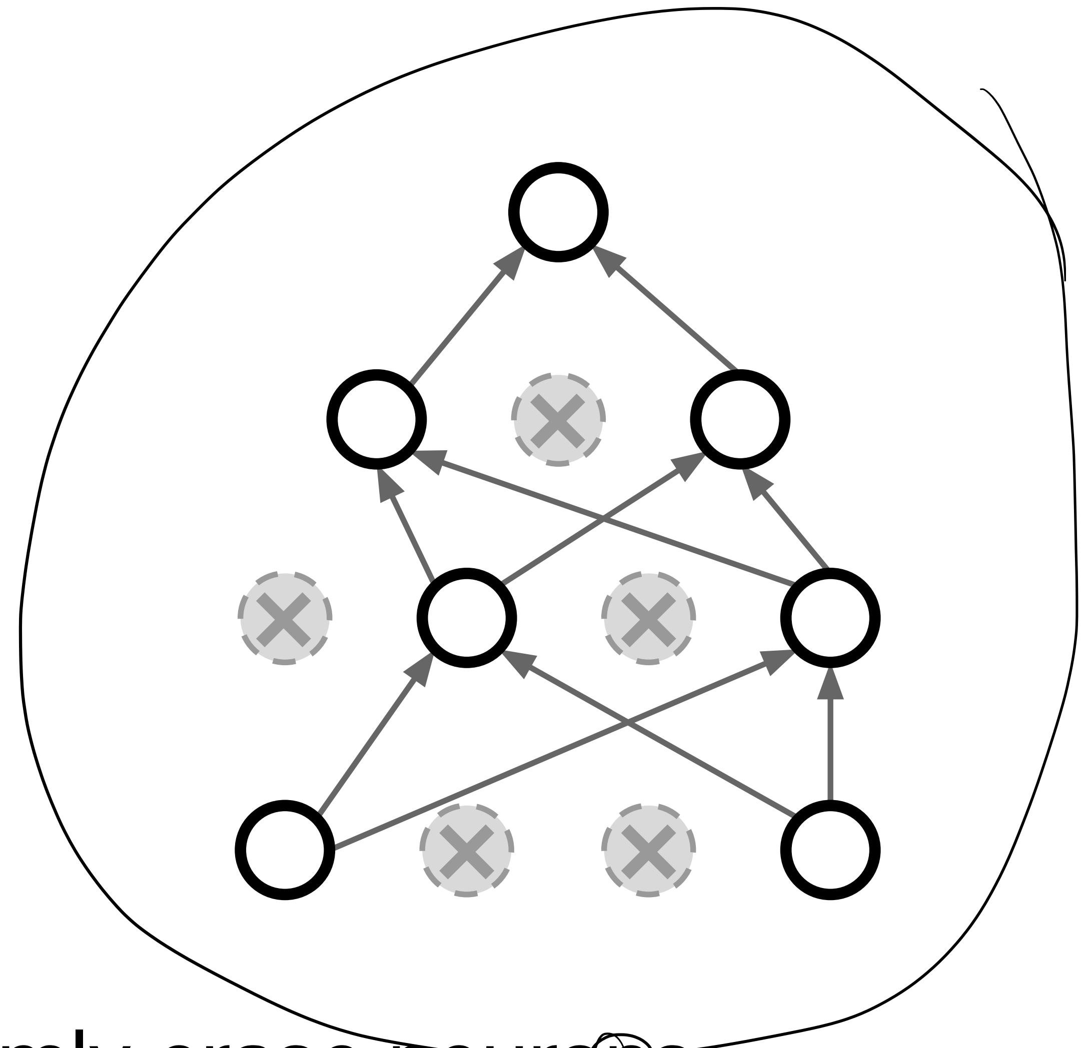
$$\Rightarrow \nabla_{W^{[l]}} J = \underbrace{\frac{1}{m} \sum_{i=1}^{m} \nabla_{W^{[l]}} l(\hat{Y}^{(i)}, Y^{(i)})}_{\text{Back prop.}} + \frac{\lambda}{m} \cdot W^{[l]}$$

$$W^{[l]}(t+1) = W^{[l]}(t) - \gamma \cdot \nabla_{W^{[l]}} J(W^{[l]}(t))$$

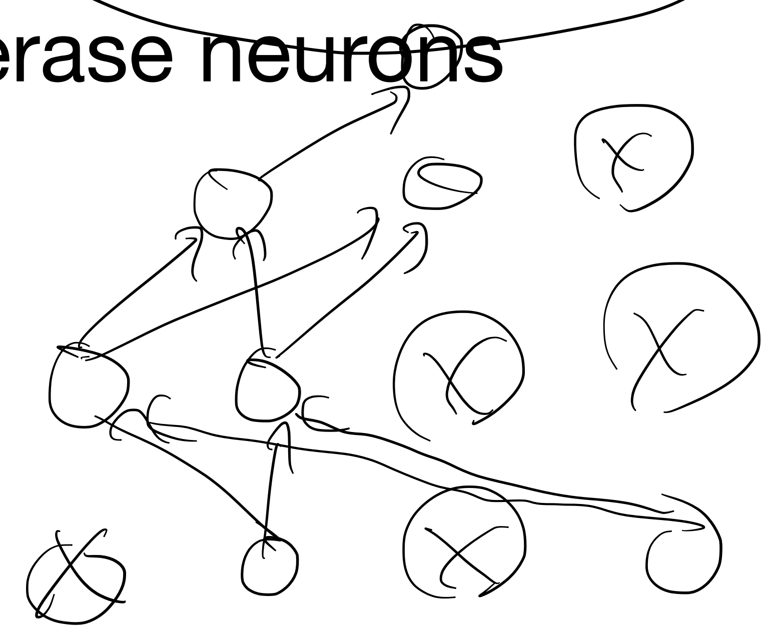
$$= \left(1 - \frac{\gamma\lambda}{m}\right) W^{[l]}(t) - \underbrace{\frac{\gamma}{m} \sum_{i=1}^{m} \nabla_{W^{[l]}} l(\hat{Y}^{(i)}, Y^{(i)})}_{\text{Back prop.}}$$

Dropout



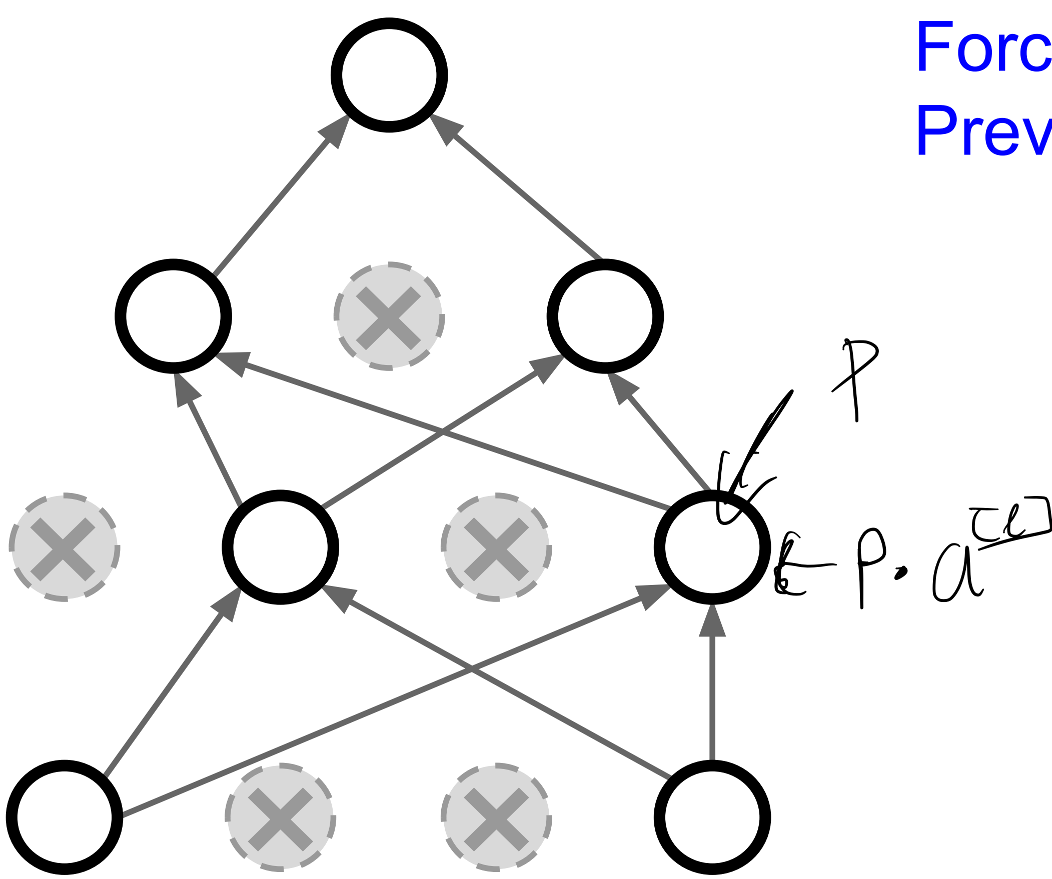


- In each forward pass, randomly erase neurons



Dropout

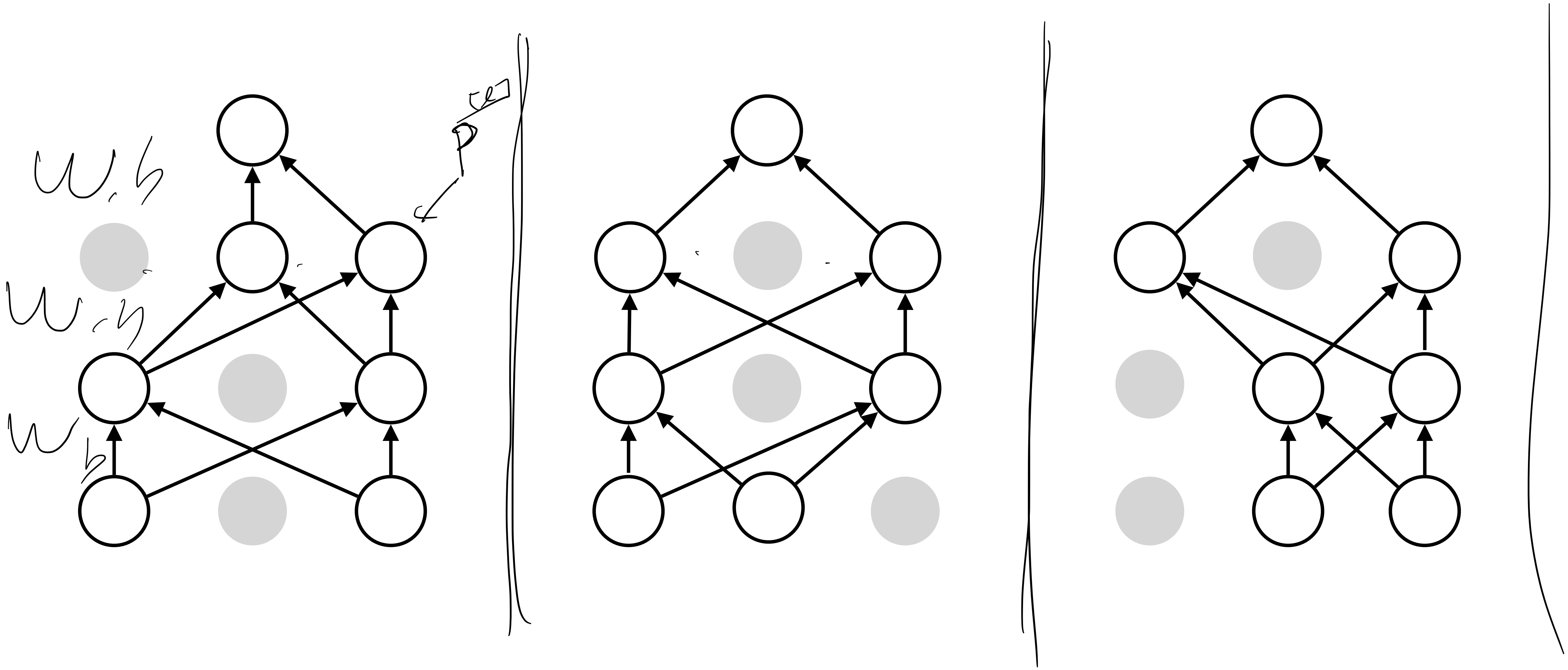
- Why can this be good?



Forces the network to have a redundant representation;
Prevents co-adaptation of features

Training

Ensemble of models



- Dropout is training a large ensemble of models (that share parameters).

- Each binary mask is one model

Dropout: Test time

no dropout.

$a^{[0]} = X$

$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

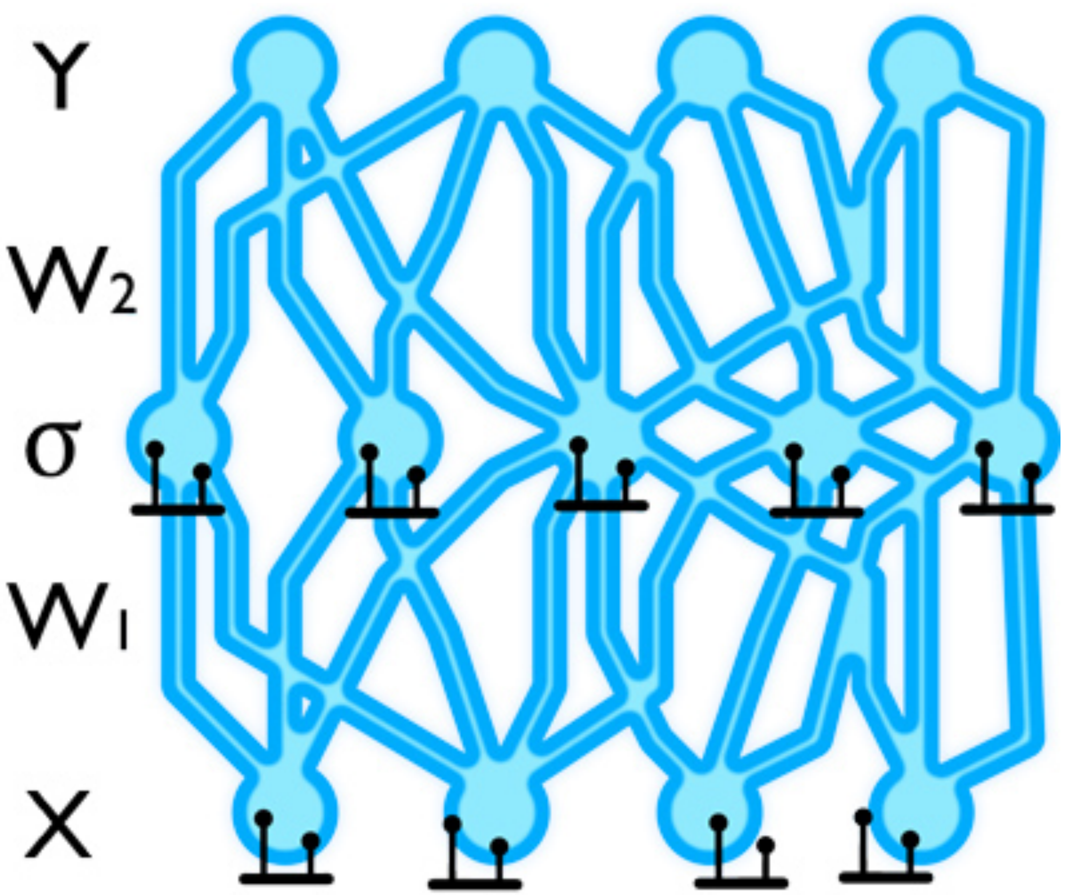
$a^{[1]} = \sigma^{[1]}(z^{[1]})$ $\Rightarrow$ $a^{[l]} = p^{[l]} \cdot \sigma^{[l]}(z^{[l]})$

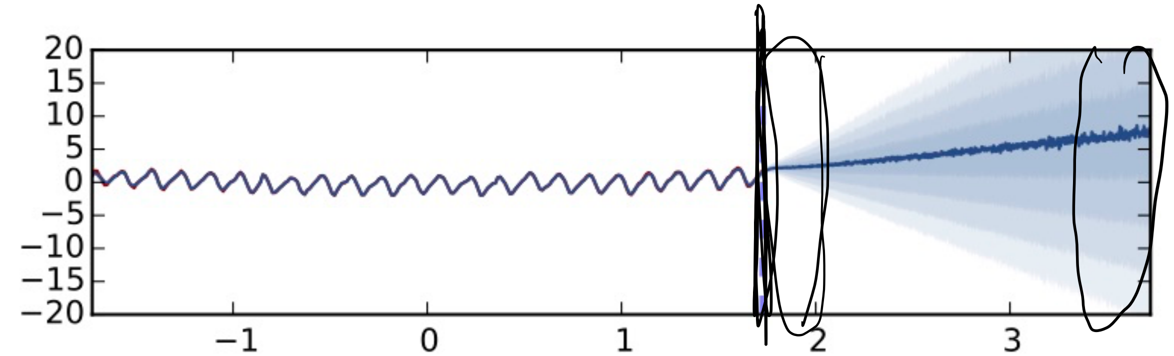
$\vdots$

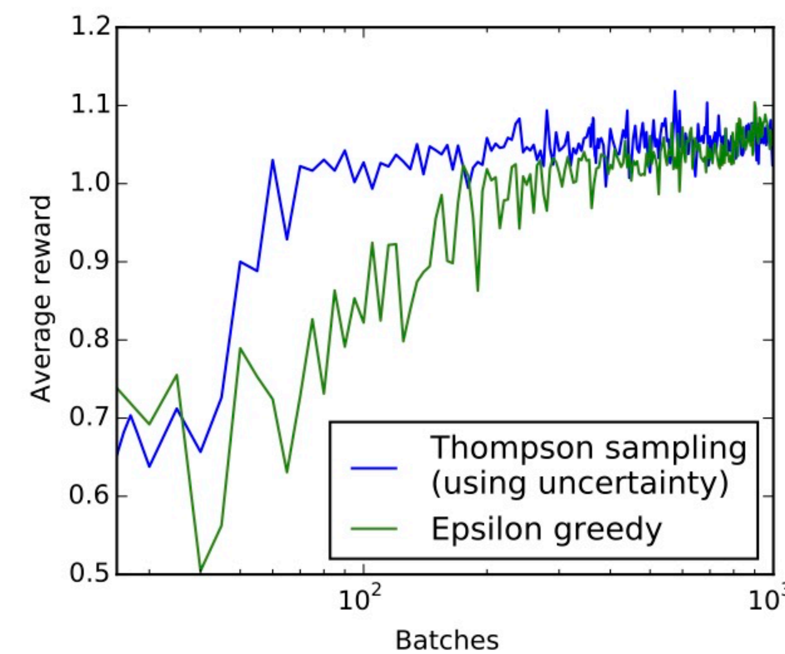
$\hat{Y} = a^{[L]}$

- No dropout at test time
- scaling by dropout probability

Dropout: uncertainty







- Dropout generates ensemble of models

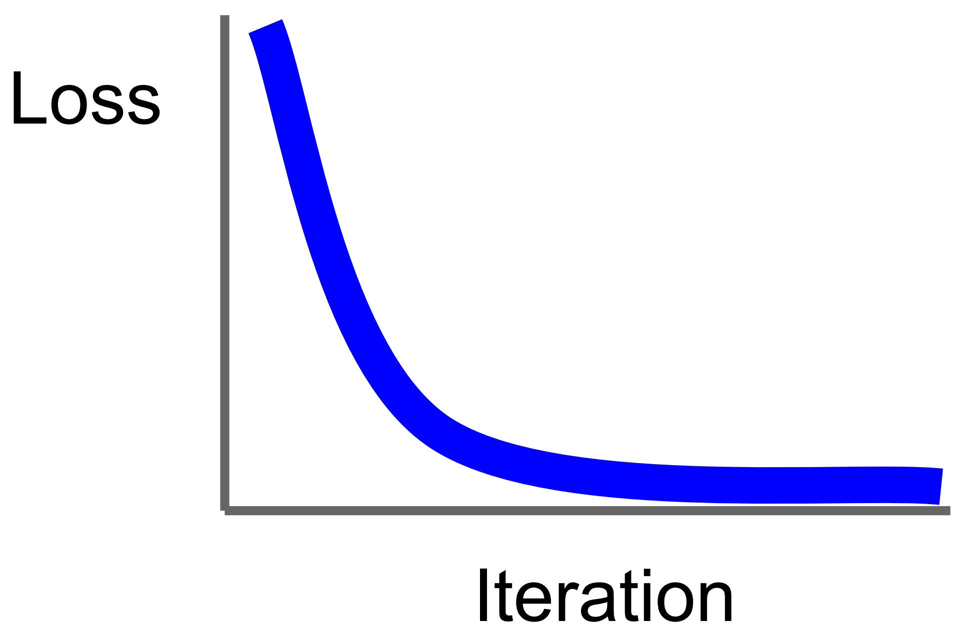
- From the ensemble, estimate mean and variance of the output

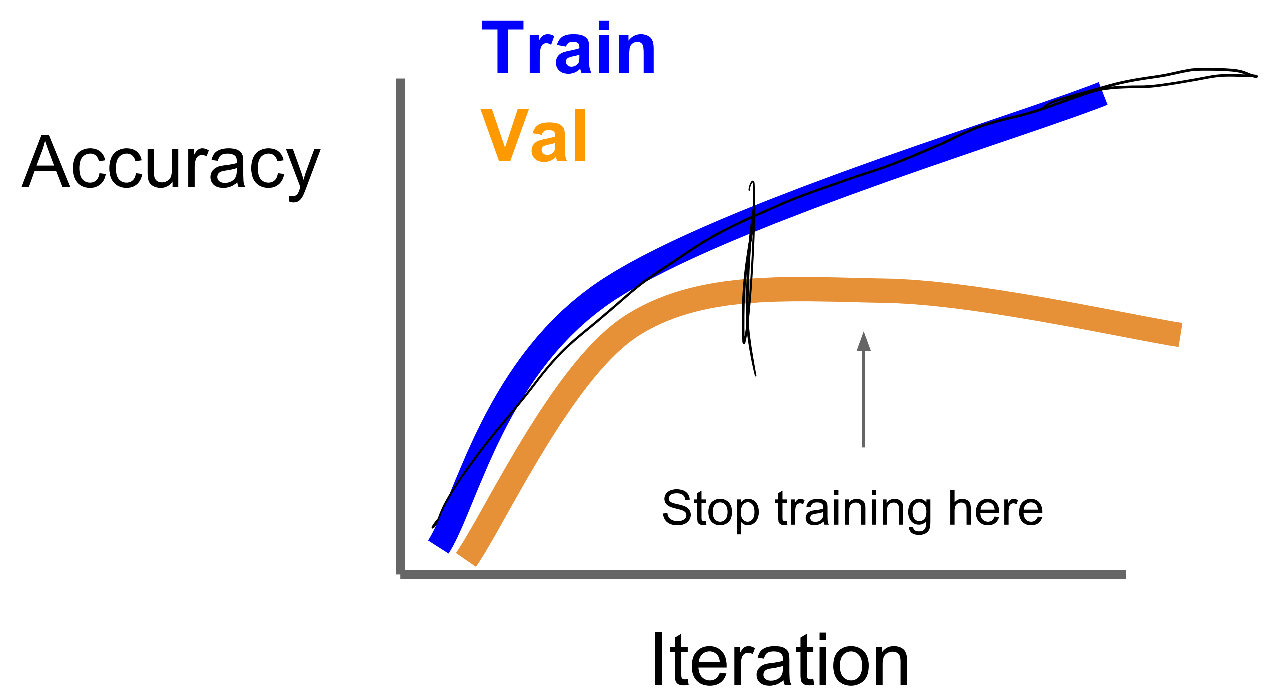
Y. Gal and Z. Ghahrami, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” https://arxiv.org/abs/1506.02142

SGD and Early stopping

- SGD adds noises to the network -> a regularizer
- Early stopping

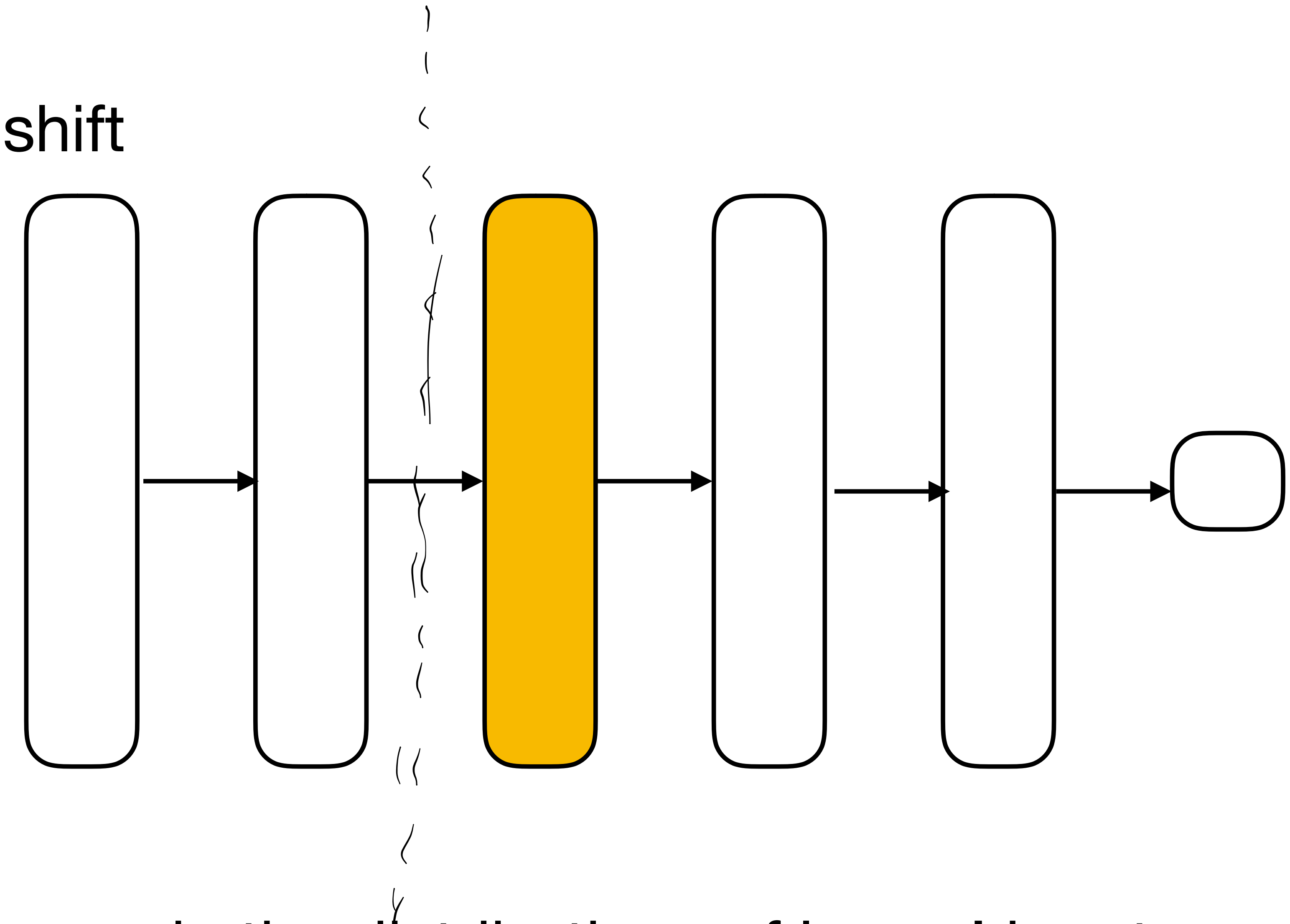
$$\mathcal{L}(w) = \sum_{i=1}^{n} \ell(y^{(i)}, \hat{y}^{(i)})$$





Batch Normalization

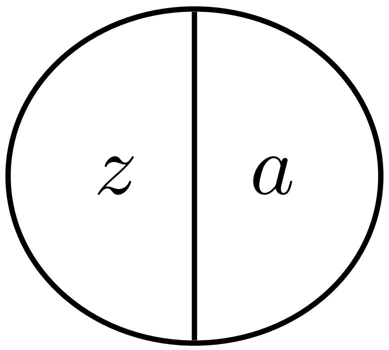
- Covariate shift

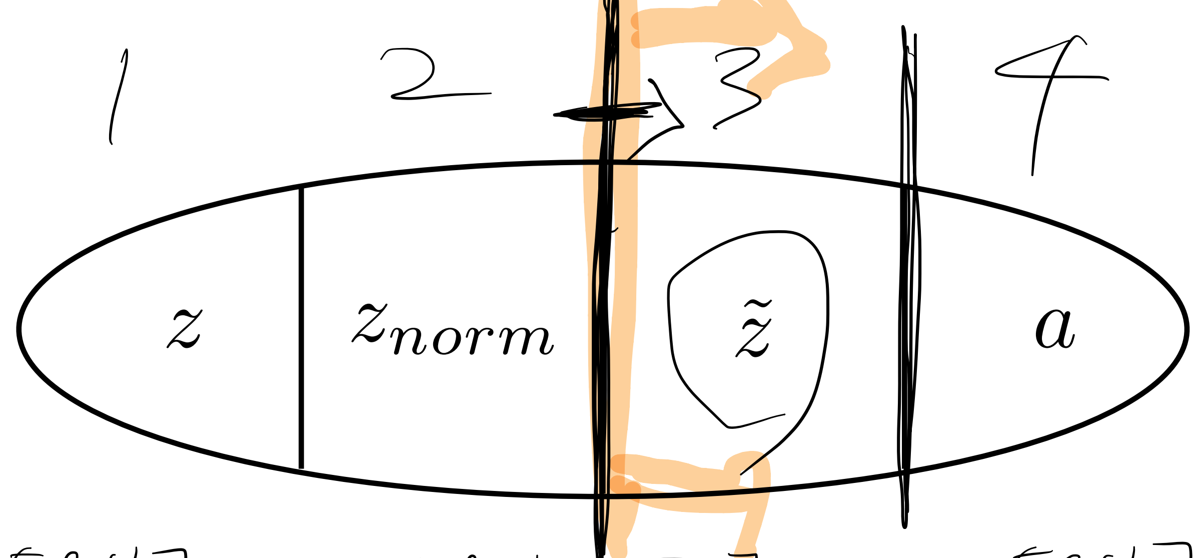


- “The change in the distributions of layers’ inputs presents a problem because the layers need to continuously adapt to the new distribution. When the input distribution to a learning system changes, it is said to experience covariate shift”

Batch Normalization

$\mathcal{N}(0,1)$





$$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$$

$$a^{[l+1]} = \sigma(z^{[l+1]})$$

$$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$$

$$z_{norm}^{[l+1]} = \frac{z^{[l+1]} - \mu_z^{[l+1]}}{\sigma_z^{[l+1]}}$$

$$\tilde{z}^{[l+1]} = \gamma^{[l+1]} \odot z_{norm}^{[l+1]} + \beta^{[l+1]}$$

Vector.

element-wise.

$$a^{[l+1]} = \sigma(\tilde{z}^{[l+1]})$$

Training with Batch Norm.

- Original: $\{w^{[1]}, b^{[1]}, \ldots, w^{[L]}, b^{[L]}\}$
- With Batch Norm.: $\{w^{[1]}, b^{[1]}, \beta^{[1]}, \gamma^{[1]}, \ldots, w^{[L]}, b^{[L]}, \beta^{[L]}, \gamma^{[L]}\}$

$w, b, \beta, \gamma \leftarrow w, b, \beta, \gamma - \eta \nabla \mathcal{L}(w, b, \beta, \gamma)$

learning

$\hat{\mu}^{[l]}, \hat{\sigma}^{[l]}$

with mini-Batch, $B = 8, 16, 32$.

$X^{\{1\}}, X^{\{2\}}, X^{\{3\}}, \ldots$

$X^{\{m\}} = \{X^{\{m\}(1)}, \ldots, X^{\{m\}(B)}\}$

$\hat{\mu}^{[l]} = \frac{1}{B} \sum_{b=1}^{B} Z^{[l]\{m\}(b)}$

$\hat{\sigma}^{2[l]} = \frac{1}{B-1} \sum_{b=1}^{B} (Z^{[l]\{m\}(b)} - \hat{\mu}^{[l]})^2$

Batch Norm at test time

$\underline{\mu^{[l]}, \sigma^{[l]}}$

$\mu^{[l]} \approx \hat{\mu}^{[l]}$ ← last mini-batch

$\approx \underline{\hat{\mu}^{[l]}}$ ← using all training data

$$\hat{\mu}^{[l]} = \frac{\eta^{M} \hat{\mu}^{[l]\{1\}} + \eta^{M-1} \hat{\mu}^{[l]\{2\}} + \cdots + \eta\, \mu^{[l]\{M\}}}{\eta^{M} + \cdots + \eta}$$

$\eta < 1$

$$\Rightarrow \boxed{\hat{\mu}^{[l]}} \Leftarrow (1-\eta)\, \boxed{\hat{\mu}^{[l]\{M\}}} + \eta\, \boxed{\hat{\mu}^{[l]}}$$

Batch Norm as regularization

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch.

- This adds some noise to the values $z^{[l]}$ within that minibatch. So similar to dropout, it adds some noise to each hidden layer's activations.

- This has a slight regularization effect.