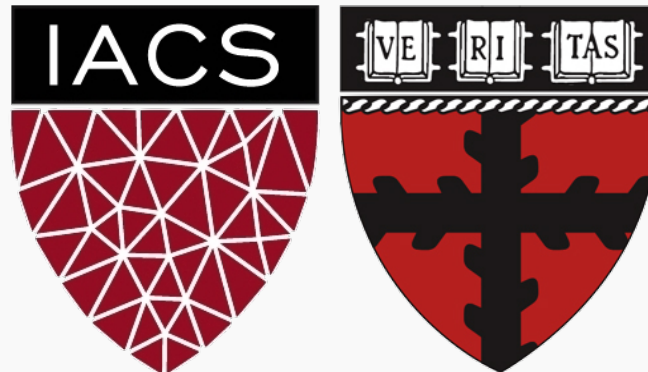


# Lecture 19: Regularization

CS109A Introduction to Data Science

Pavlos Protopapas and Kevin Rader



# Outline

---

## Regularization of NN

- Norm Penalties
- Early Stopping
- Data Augmentation
- Sparse Representation
- Bagging
- Dropout

# Outline

---

## Regularization of NN

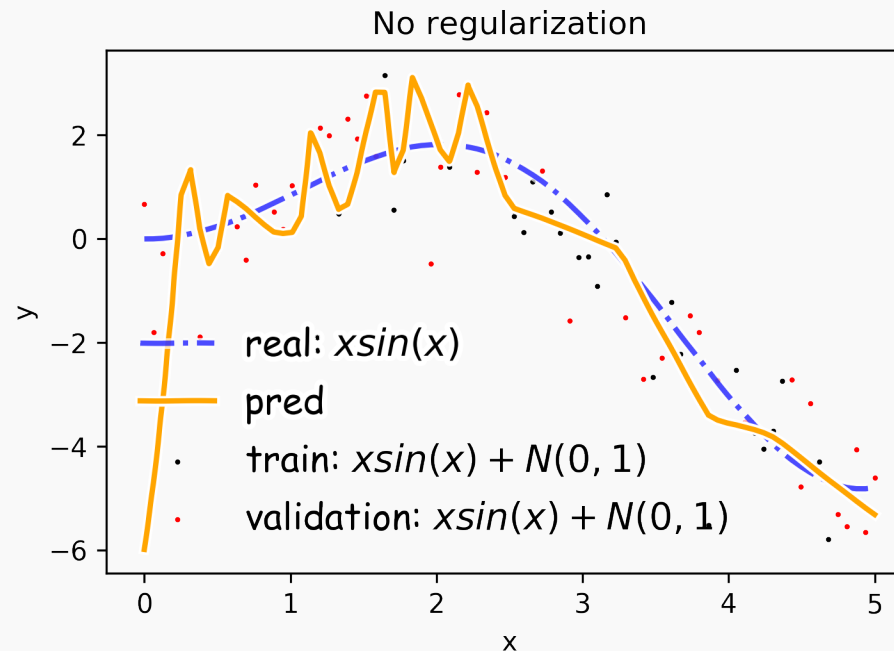
- **Norm Penalties**
- Early Stopping
- Data Augmentation
- Sparse Representation
- Bagging
- Dropout

# Regularization

Regularization is any modification we make to a learning algorithm that is intended to **reduce its generalization** error but not its training error.

# Overfitting

Fitting a deep neural network with 5 layers and 100 neurons per layer can lead to a very good prediction on the training set but poor prediction on validation set.



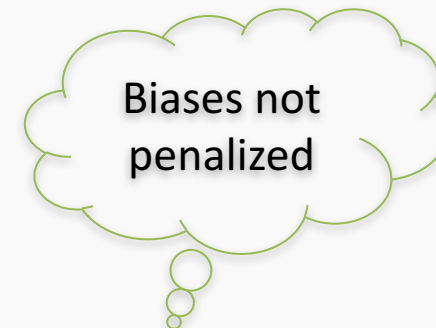
# Norm Penalties

We used to optimize:

$$J(W; X, y)$$

Change to ...

$$J_R(W; X, y) = J(W; X, y) + \alpha \Omega(W)$$



$L_2$  regularization:

- Weights decay
- MAP estimation with Gaussian prior

$$\Omega(W) = \frac{1}{2} \| W \|_2^2$$

$L_1$  regularization:

- encourages sparsity
- MAP estimation with Laplacian prior

$$\Omega(W) = \frac{1}{2} \| W \|_1$$

# Norm Penalties

We used to optimize:

Change to ...

$$\begin{aligned} W^{(i+1)} &= W^{(i)} - \lambda \frac{\partial J}{\partial W} \\ J_R(W; X, y) &= J(W; X, y) + \frac{1}{2} \alpha W^2 \\ W^{(i+1)} &= W^{(i)} - \lambda \frac{\partial J}{\partial W} - \lambda \alpha W \end{aligned}$$

weights  
decay in  
proportion  
to its size.

Biases not  
penalized

$L_2$  regularization:

- **Decay of weights**
- MAP estimation with Gaussian prior

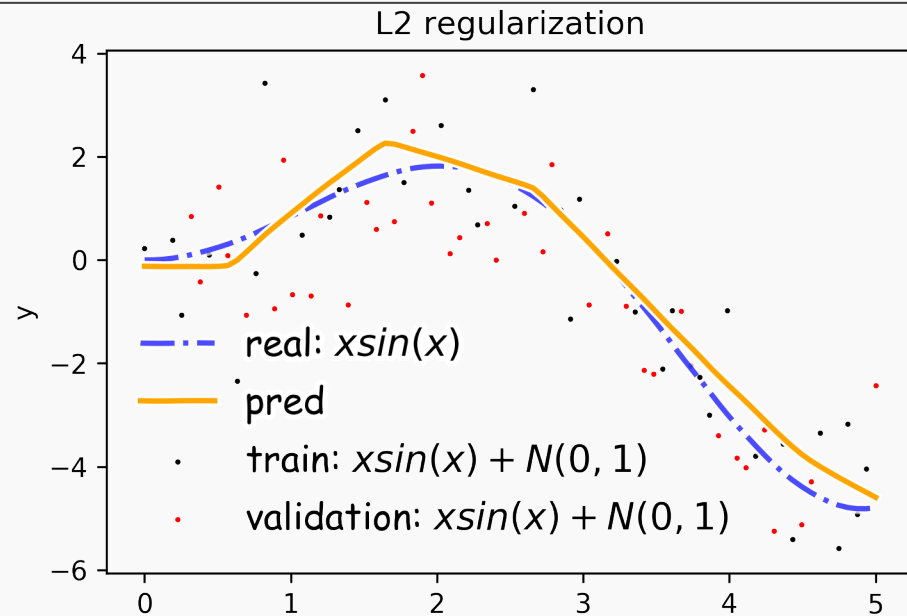
$L_1$  regularization:

- encourages sparsity
- MAP estimation with Laplacian prior

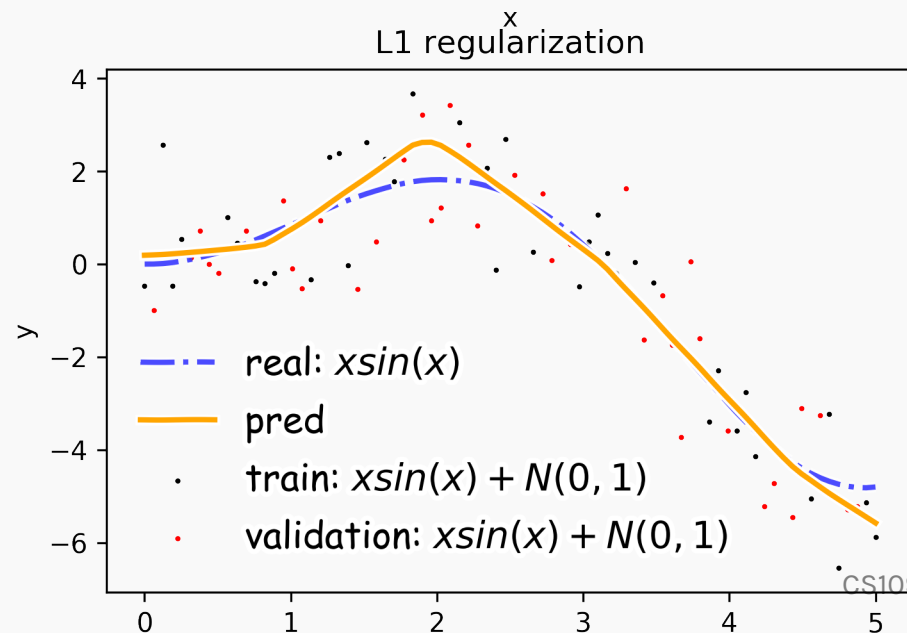
$$\Omega(W) = \frac{1}{2} \|W\|_2^2$$

$$\Omega(W) = \frac{1}{2} \|W\|_1$$

# Norm Penalties



$$\Omega(W) = \frac{1}{2} \|W\|_2^2$$



$$\Omega(W) = \frac{1}{2} \|W\|_1$$



# Norm Penalties as Constraints

---

$$\min_{\Omega(W) \leq K} J(W; X, y)$$

Useful if  $K$  is known in advance

Optimization:

- Construct Lagrangian and apply gradient descent
- Projected gradient descent

# Outline

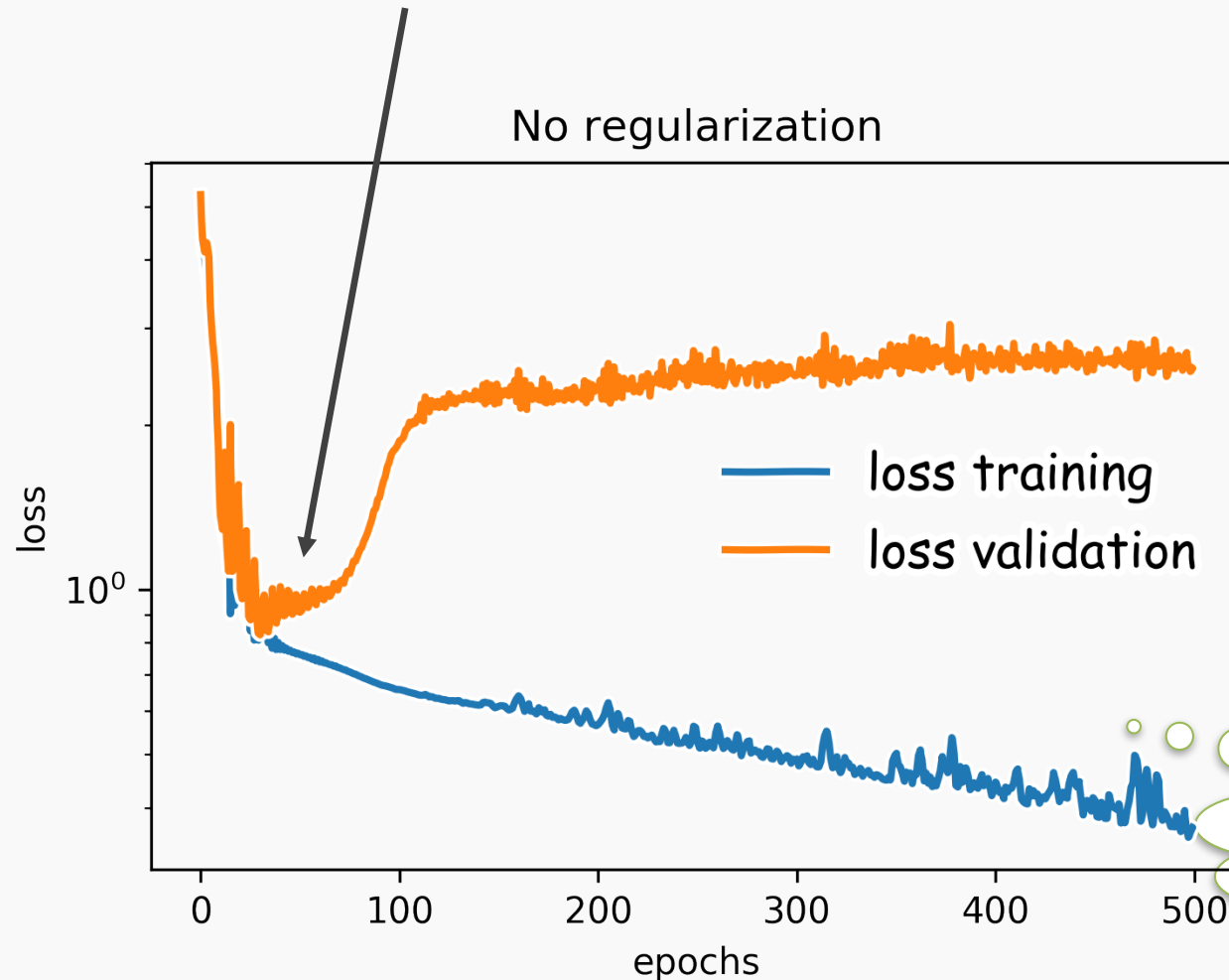
---

## Regularization of NN

- Norm Penalties
- **Early Stopping**
- Data Augmentation
- Sparse Representation
- Bagging
- Dropout

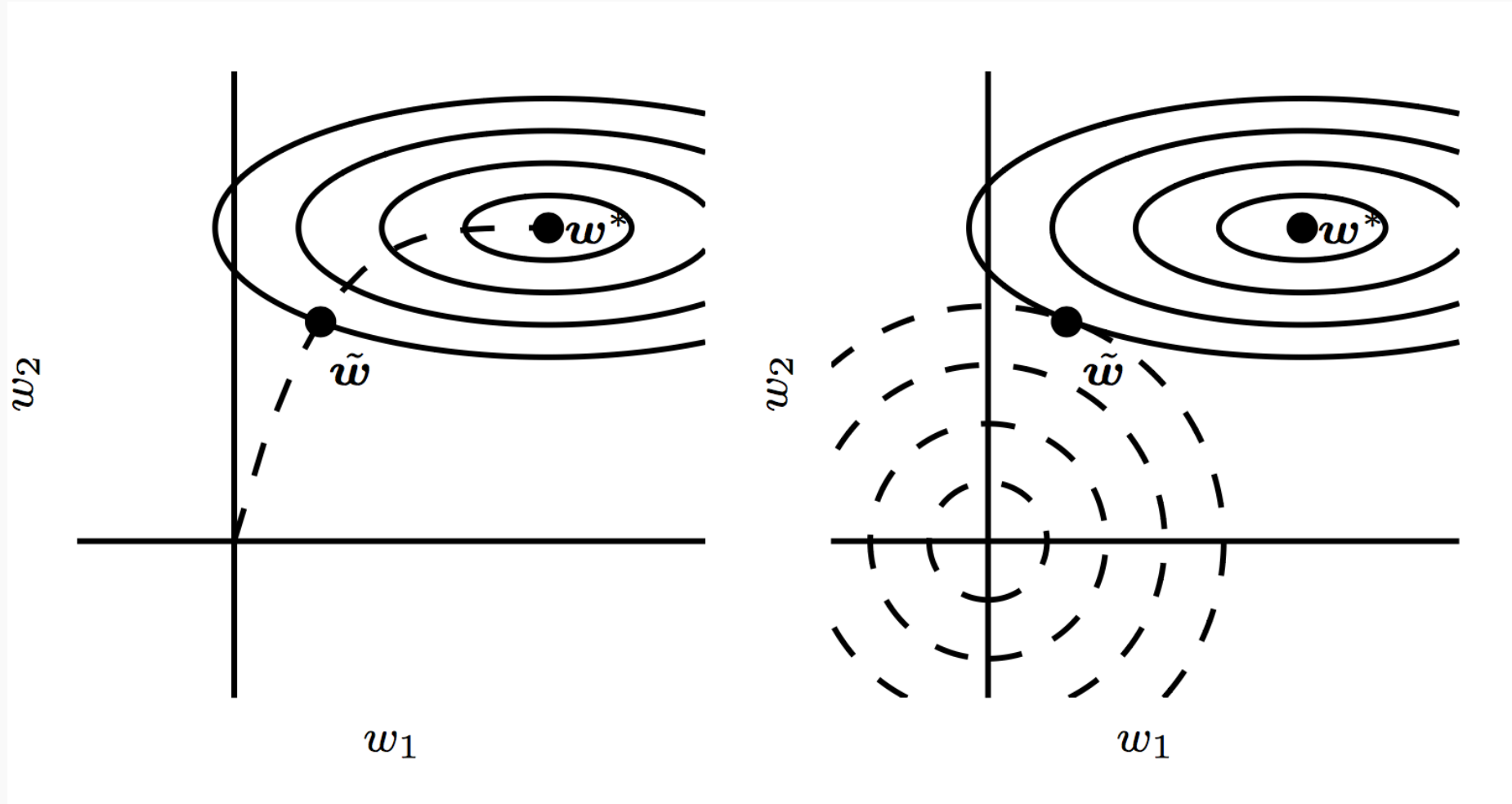
# Early Stopping

Early stopping: terminate while validation set performance is better



Training time can be treated as a hyperparameter

# Early Stopping



# Outline

---

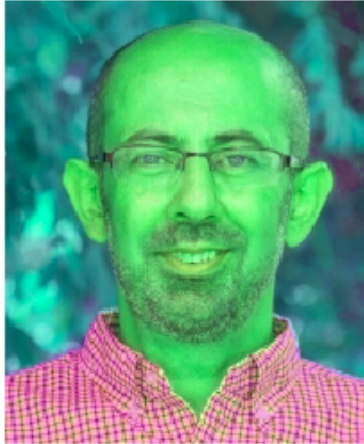
## Regularization of NN

- Norm Penalties
- Early Stopping
- **Data Augmentation**
- Sparse Representation
- Bagging
- Dropout

# Data Augmentation



hue



crop-and-pan



elastic



flip-lr



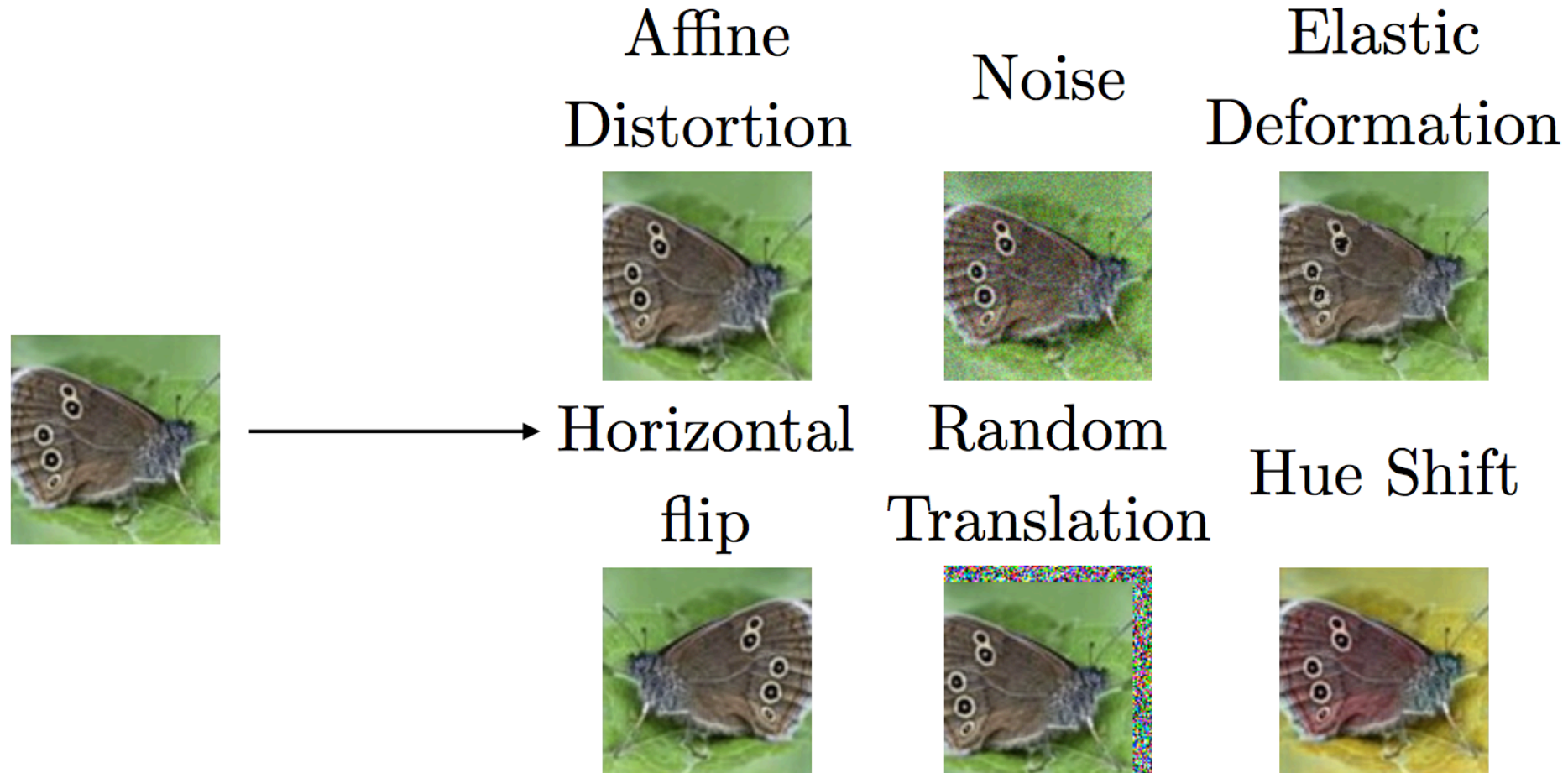
flip-ud



rotate



# Data Augmentation



# Outline

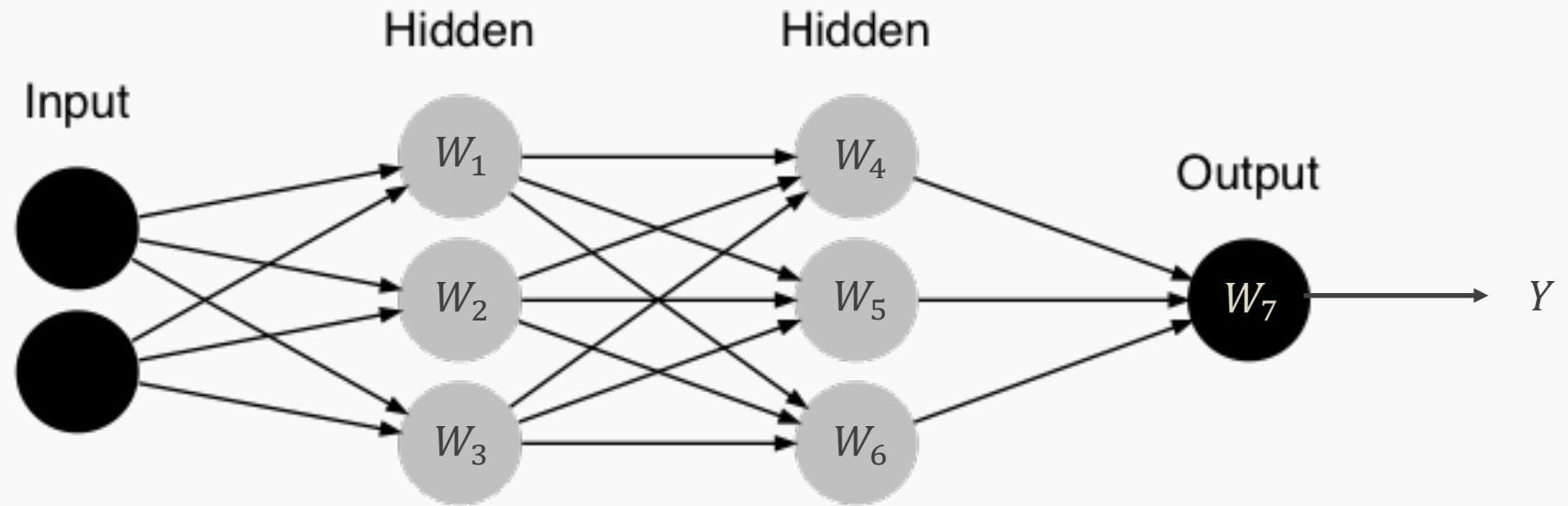
---

## Regularization of NN

- Norm Penalties
- Early Stopping
- Data Augmentation
- **Sparse Representation**
- Bagging
- Dropout



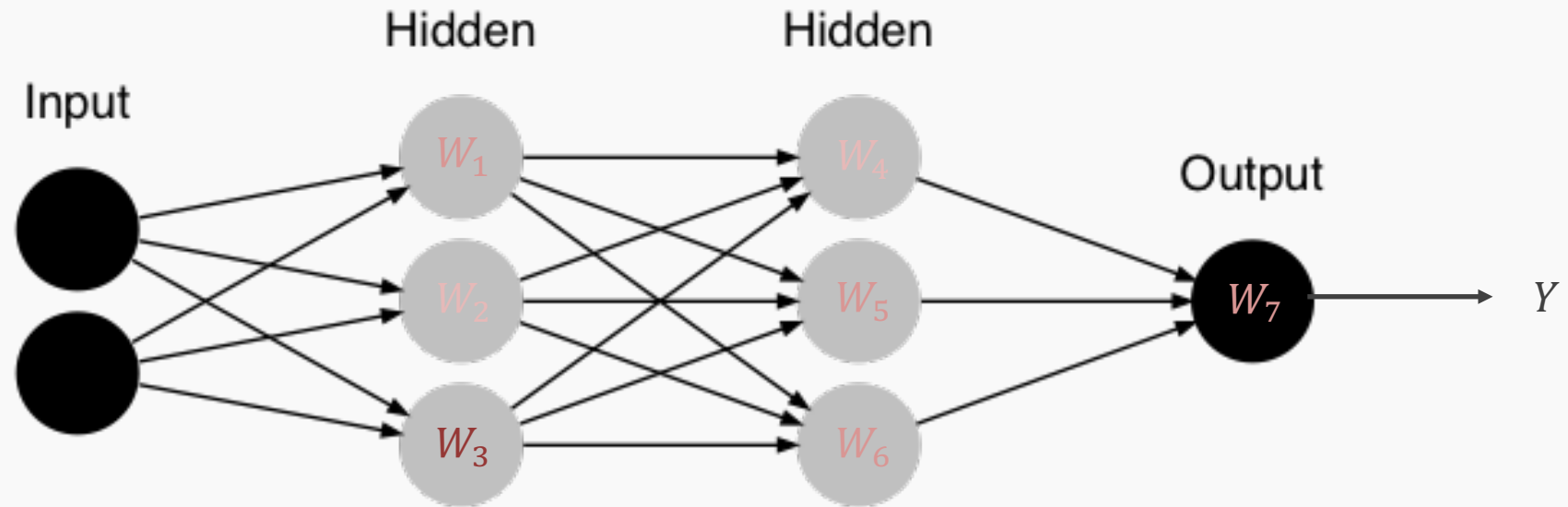
# Sparse Representation



$$J(\theta; X, y)$$

$$[4.34] = \underbrace{[3.2 \quad 2.0 \quad 1.8]}_{W_7} \begin{bmatrix} 2 \\ -2.2 \\ 1.3 \end{bmatrix}$$

# Sparse Representation



$$J_R(W; X, y) = J(\theta; X, y) + \alpha \Omega(W)$$

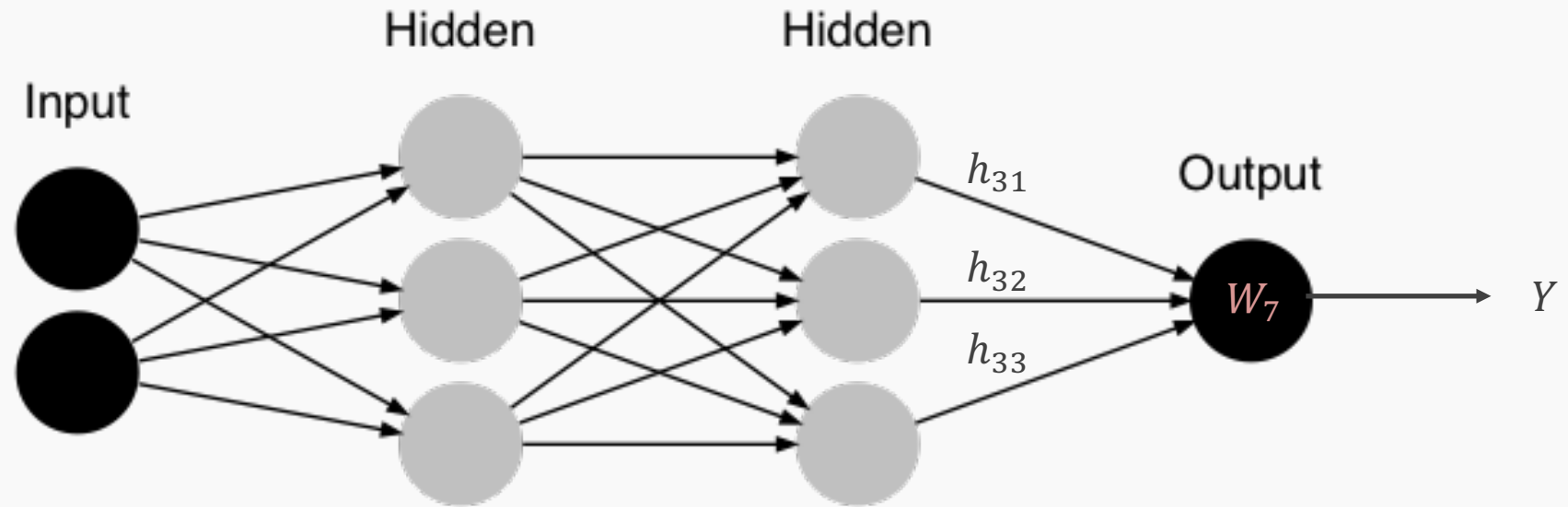
$$[0.69] = \underbrace{[0.5 \quad .2 \quad 0.1]}_{W_7} \begin{bmatrix} 2 \\ -2.2 \\ 1.3 \end{bmatrix}$$

$W_7$



**Weights in output layer**

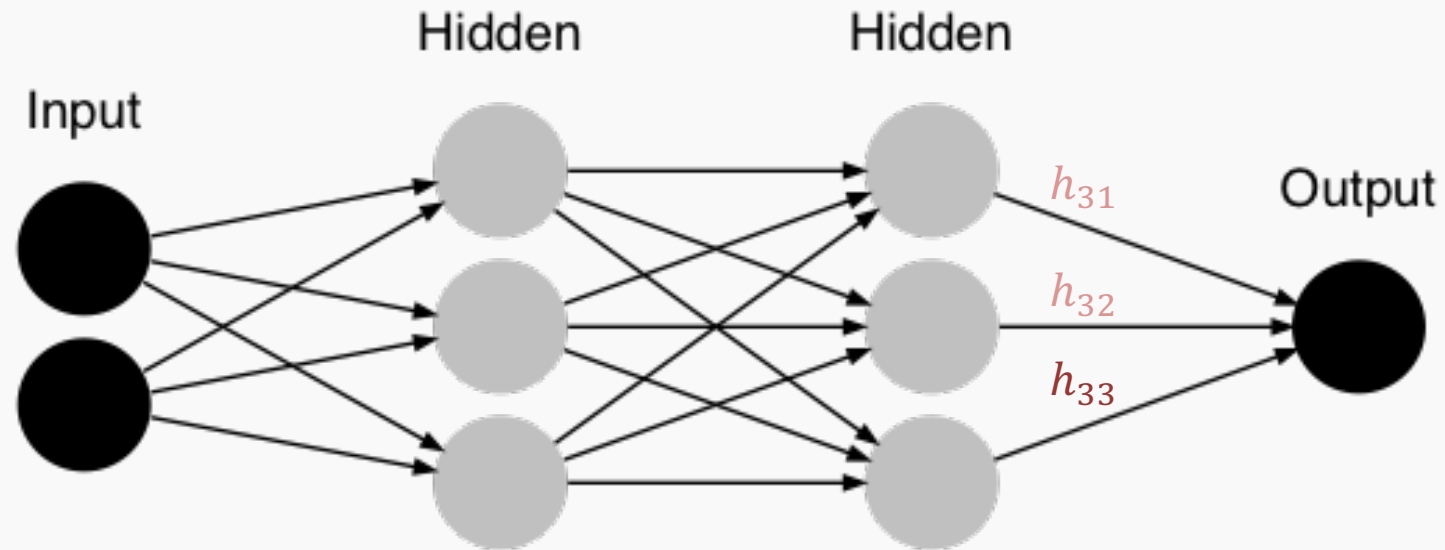
# Sparse Representation



$$J(\theta; X, y)$$

$$[4.34] = [3.2 \quad 2 \quad 1] \left[ \begin{array}{c} 2 \\ -2.2 \\ 1.3 \end{array} \right] \Bigg\} h_{31}, h_{32}, h_{33}$$

# Sparse Representation



$$J_R(W; X, y) = J(\theta; X, y) + \alpha \Omega(h)$$

$$[1.3] = [3.2 \quad 2 \quad 1] \begin{bmatrix} 0 \\ -0.2 \\ .9 \end{bmatrix} \quad \left. \vphantom{\begin{bmatrix} 0 \\ -0.2 \\ .9 \end{bmatrix}} \right\} h_{31}, h_{32}, h_{33}$$

Output of hidden layer

# Outline

---

## Regularization of NN

- Norm Penalties
- Early Stopping
- Data Augmentation
- Sparse Representation
- **Bagging**
- Dropout

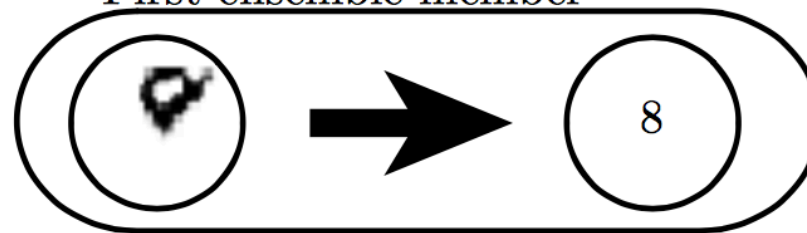
Original dataset



First resampled dataset



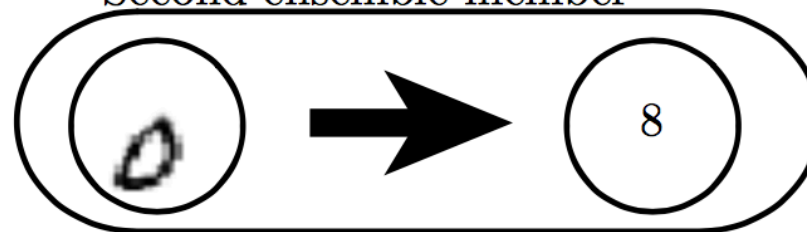
First ensemble member



Second resampled dataset



Second ensemble member



# Outline

---

## Regularization of NN

- Norm Penalties
- Early Stopping
- Data Augmentation
- Sparse Representation
- Bagging
- **Dropout**

# Noise Robustness

---

## Random **perturbation of network weights**

- Gaussian noise: Equivalent to minimizing loss with regularization term
- Encourages smooth function: small perturbation in weights leads to small changes in output

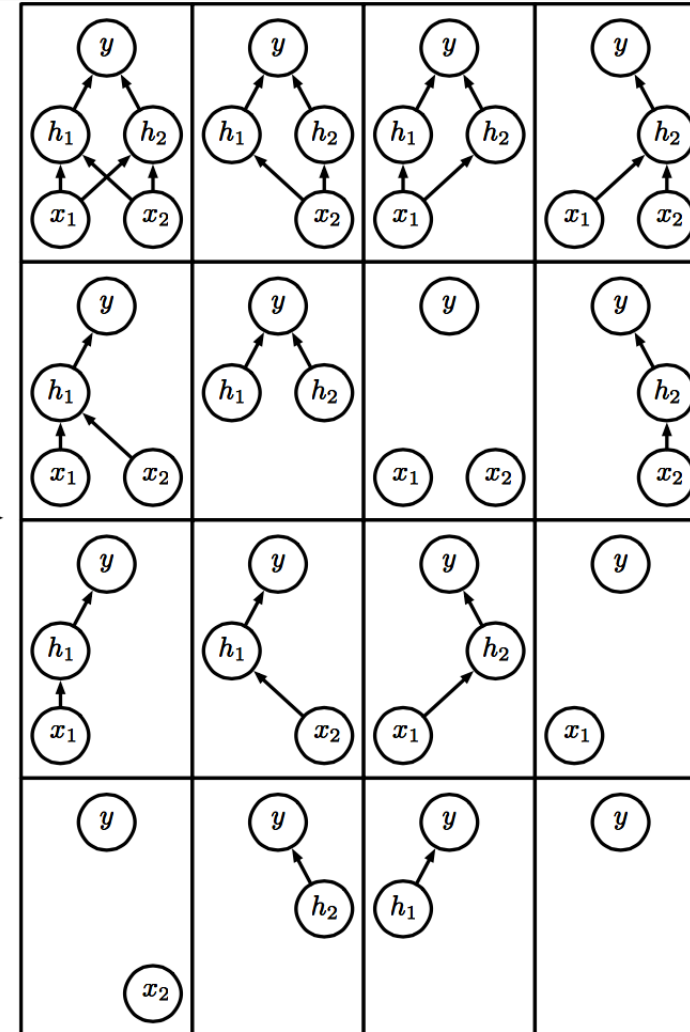
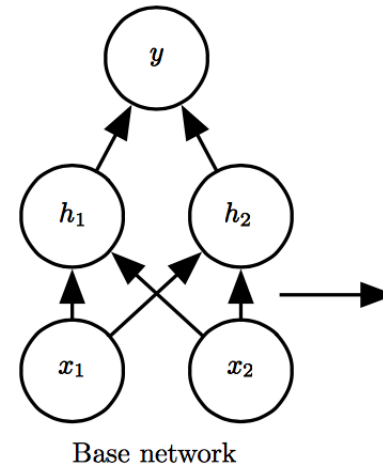
## Injecting **noise in output labels**

- Better convergence: prevents pursuit of hard probabilities



# Dropout

Train all sub-networks  
obtained by removing non-  
output units from base  
network



Ensemble of subnetworks

# Dropout: Stochastic GD

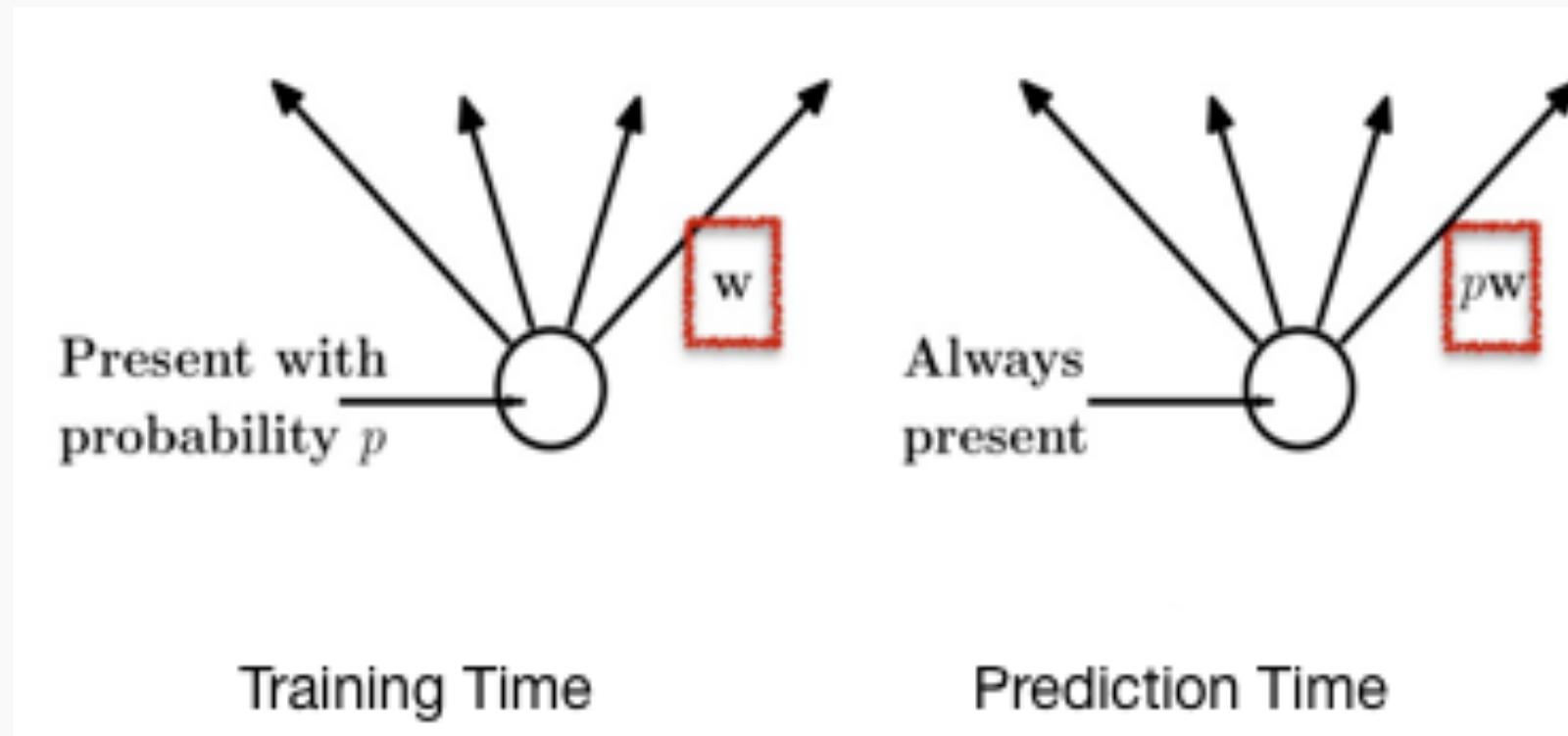
For each new example/mini-batch:

- Randomly **sample a binary mask  $\mu$**  independently, where  $\mu_i$  indicates if input/hidden node  $i$  is included
- **Multiply output of node  $i$  with  $\mu_i$** , and perform gradient update

Typically, an input node is **included** with **prob=0.8**, hidden node with **prob=0.5**.

# Dropout: Weight Scaling

During prediction time use all units, but scale weights with probability of inclusion



# Adversarial Examples

# Adversarial Examples



+



=



Panda 57% confidence

noise

Gibbon 99.3% confidence

Training on adversarial examples is mostly intended to improve security, but can sometimes provide generic regularization.

# Recap

---

## Regularization of NN

- Norm Penalties
- Early Stopping
- Data Augmentation
- Sparse Representation
- Bagging
- Dropout

Advanced Topics will focus on Image Analysis