

# Deep Learning:

# Regularization & Generalization

**Robert Legenstein**

Institute of Machine Learning and Neural Computation

[robert.legenstein@tugraz.at](mailto:robert.legenstein@tugraz.at)

Deep Learning VO - WS 25/26

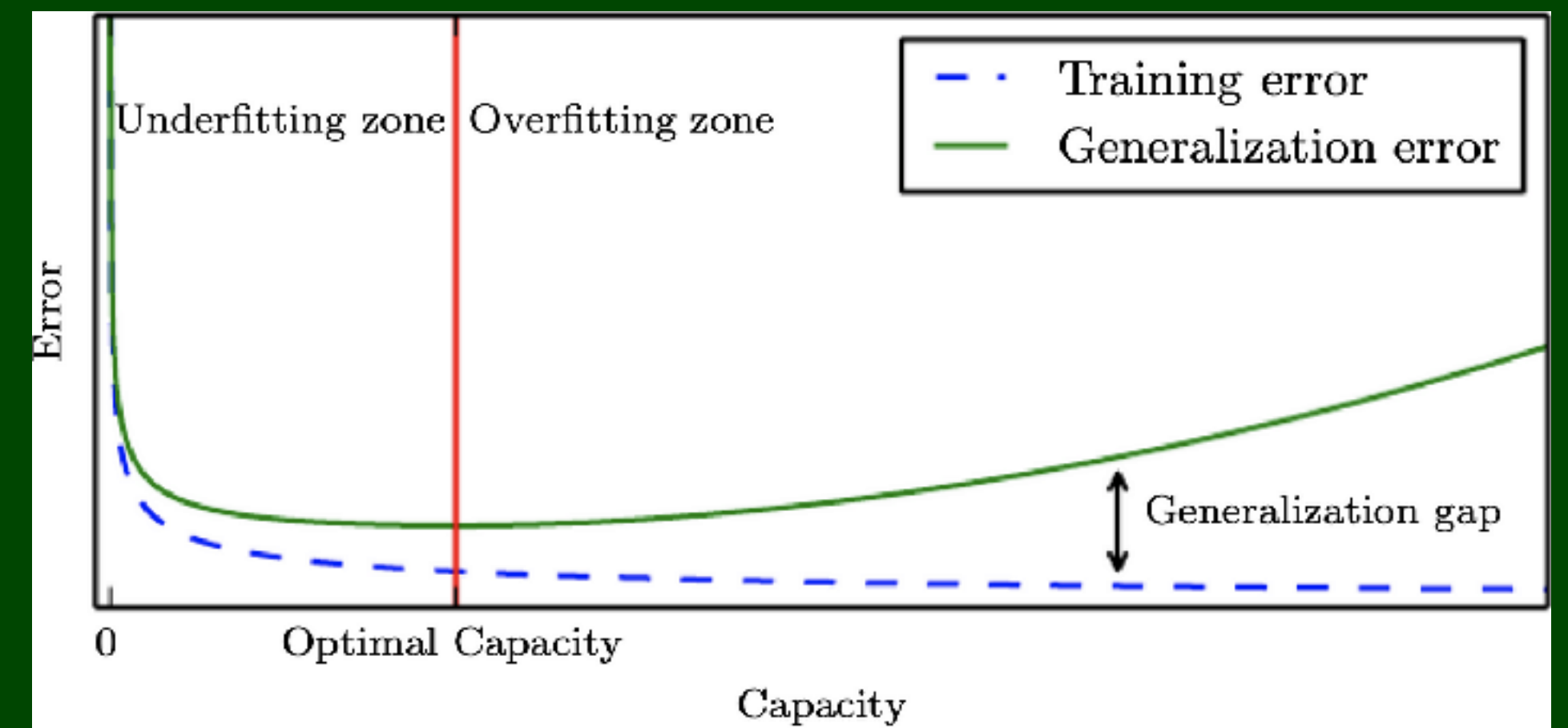
Lecture 6

# Today

- ❑ Regularization
  - ❑ Parameter Norm Penalties
  - ❑ Early Stopping
  - ❑ Dropout
  - ❑ Dataset Augmentation
  - ❑ Further Techniques

## Recap

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



# (1) Parameter Norm Penalties

---

Adding a parameter norm penalty  $\Omega(\mathbf{w})$  to the error  $E$ .

$$\tilde{E}(\mathbf{w}; \mathcal{D}) = E(\mathbf{w}; \mathcal{D}) + \lambda \Omega(\mathbf{w})$$

- Typically only regularize weights, leave biases unregularized.
- Sometimes it is desirable to use different  $\lambda$  for different layers.

$\mathbf{w}$  : network parameters

$\mathcal{D}$  : data

$\lambda \in [0, \infty)$  : relative contribution of  
the regularizer

# $L_2$ regularization (weight decay)

Adding a parameter norm penalty  $\Omega(\mathbf{w})$  to the error  $E$ .

$$\tilde{E}(\mathbf{w}; \mathcal{D}) = E(\mathbf{w}; \mathcal{D}) + \lambda \Omega(\mathbf{w})$$

**$L_2$  regularization:**

$$\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_2^2 = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

$$\nabla_{\mathbf{w}} \tilde{E}(\mathbf{w}; \mathcal{D}) = \nabla_{\mathbf{w}} E(\mathbf{w}; \mathcal{D}) + \lambda \nabla_{\mathbf{w}} \Omega(\mathbf{w})$$

$$= \nabla_{\mathbf{w}} E(\mathbf{w}; \mathcal{D}) + \lambda \mathbf{w}$$

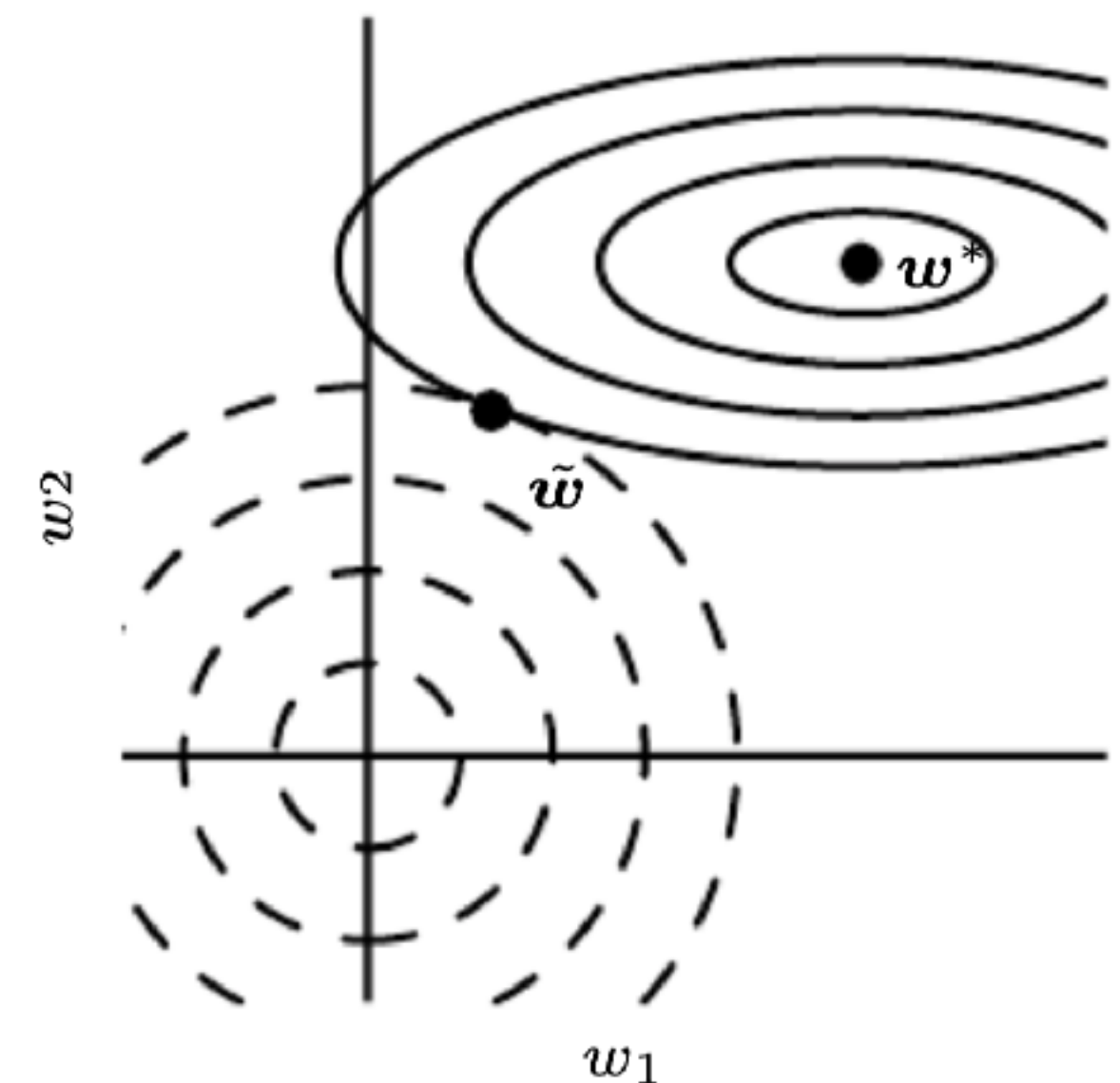
$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon (\lambda \mathbf{w} + \nabla_{\mathbf{w}} E(\mathbf{w}; \mathcal{D})) \quad \longrightarrow \text{leads to **weight decay**}$$

- Can be interpreted as MAP inference: it would correspond to a zero-mean Gaussian prior over weights.

$\mathbf{w}$  : network parameters

$\mathcal{D}$  : data

$\lambda \in [0, \infty)$  : relative contribution of the regularizer



# $L_1$ regularization

Adding a parameter norm penalty  $\Omega(\mathbf{w})$  to the error  $E$ .

$$\tilde{E}(\mathbf{w}; \mathcal{D}) = E(\mathbf{w}; \mathcal{D}) + \lambda \Omega(\mathbf{w})$$

$\mathbf{w}$  : network parameters

$\mathcal{D}$  : data

$\lambda \in [0, \infty)$  : relative contribution of the regularizer

**$L_1$  regularization:**

$$\Omega(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_i |w_i|$$

Weight updates using the sub-gradient:  $\nabla_{\mathbf{w}} \|\mathbf{w}\|_1 = \text{sign}(\mathbf{w})$

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon (\lambda \text{sign}(\mathbf{w}) + \nabla_{\mathbf{w}} E(\mathbf{w}; \mathcal{D})) \longrightarrow \text{leads to **sparse** parameter vectors (many entries are 0).}$$

- As MAP inference: corresponds to a Laplacian prior over weights.

$$p(w_i) = \frac{1}{2\sigma} e^{-\frac{|w_i|}{\sigma}}$$

- A linear model with least squares error and  $L_1$  norm regularization is known as LASSO (least absolute shrinkage and selection operator).

# L<sub>1</sub> regularization

Adding a parameter norm penalty  $\Omega(\mathbf{w})$  to the error  $E$ .

$$\tilde{E}(\mathbf{w}; \mathcal{D}) = E(\mathbf{w}; \mathcal{D}) + \lambda \Omega(\mathbf{w})$$

**L<sub>1</sub> regularization:**

$$\Omega(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_i |w_i|$$

Weight updates using the sub-gradient:  $\nabla_{\mathbf{w}} \|\mathbf{w}\|_1 = \text{sign}(\mathbf{w})$

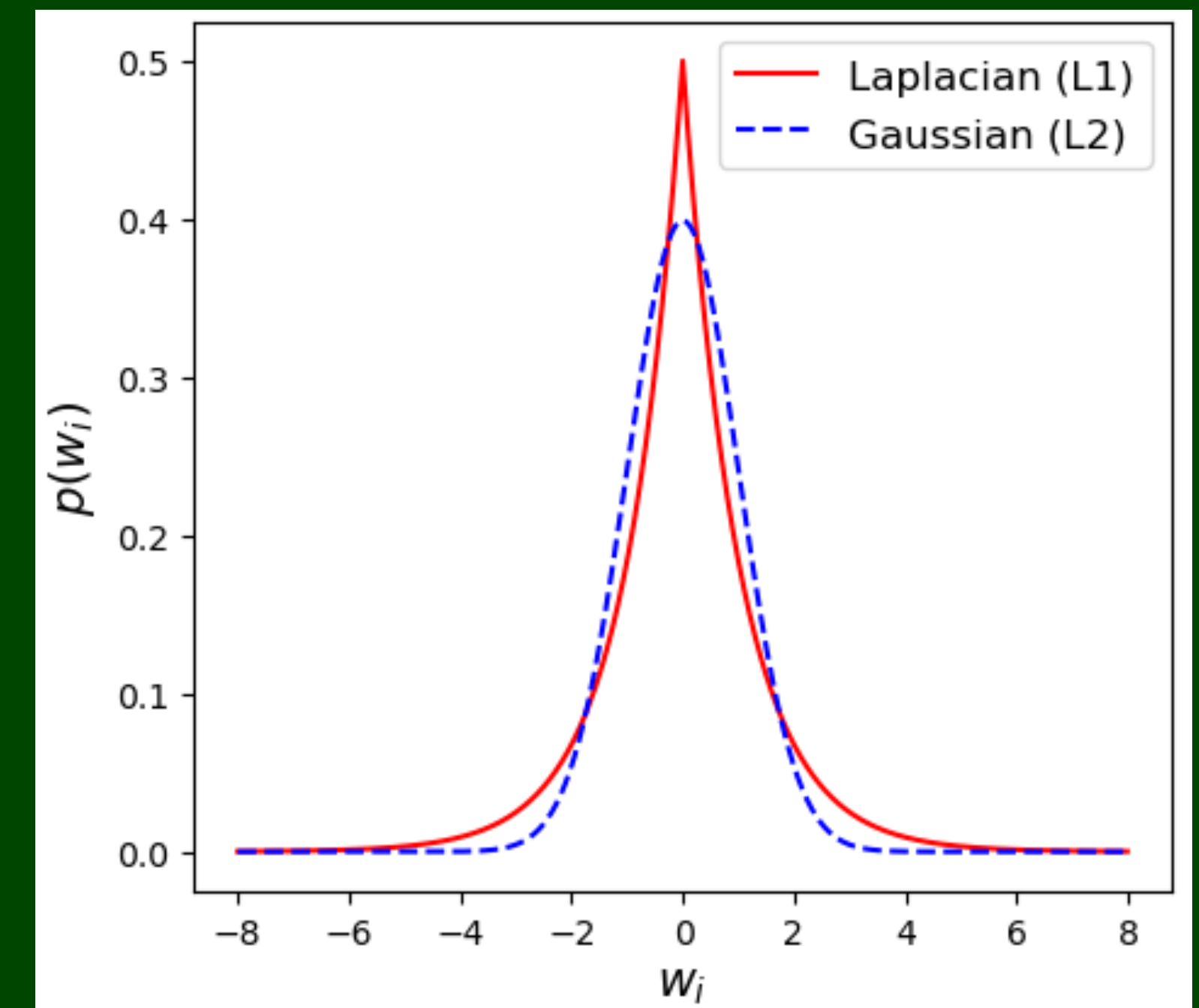
$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon (\lambda \text{sign}(\mathbf{w}) + \nabla_{\mathbf{w}} E(\mathbf{w}; \mathcal{D})) \rightarrow \text{leads to sparse vectors}$$

- As MAP inference: corresponds to a Laplacian prior over weights

$$p(w_i) = \frac{1}{2\sigma} e^{-\frac{|w_i|}{\sigma}}$$

- A linear model with least squares error and  $L_1$  norm regularization is called LASSO (least absolute shrinkage and selection operator).

Impact of L<sub>1</sub> vs L<sub>2</sub> regularization on weights:



With L<sub>1</sub> regularization:

- ▶ zero weights are more probable (sparse)
- ▶ remaining weights get higher values (w.r.t. L<sub>2</sub>)  
> since Laplacian dist. is heavier tailed

## (2) Early Stopping

- Training error decreases over training. However, test error first decreases, then increases.

### Early stopping:

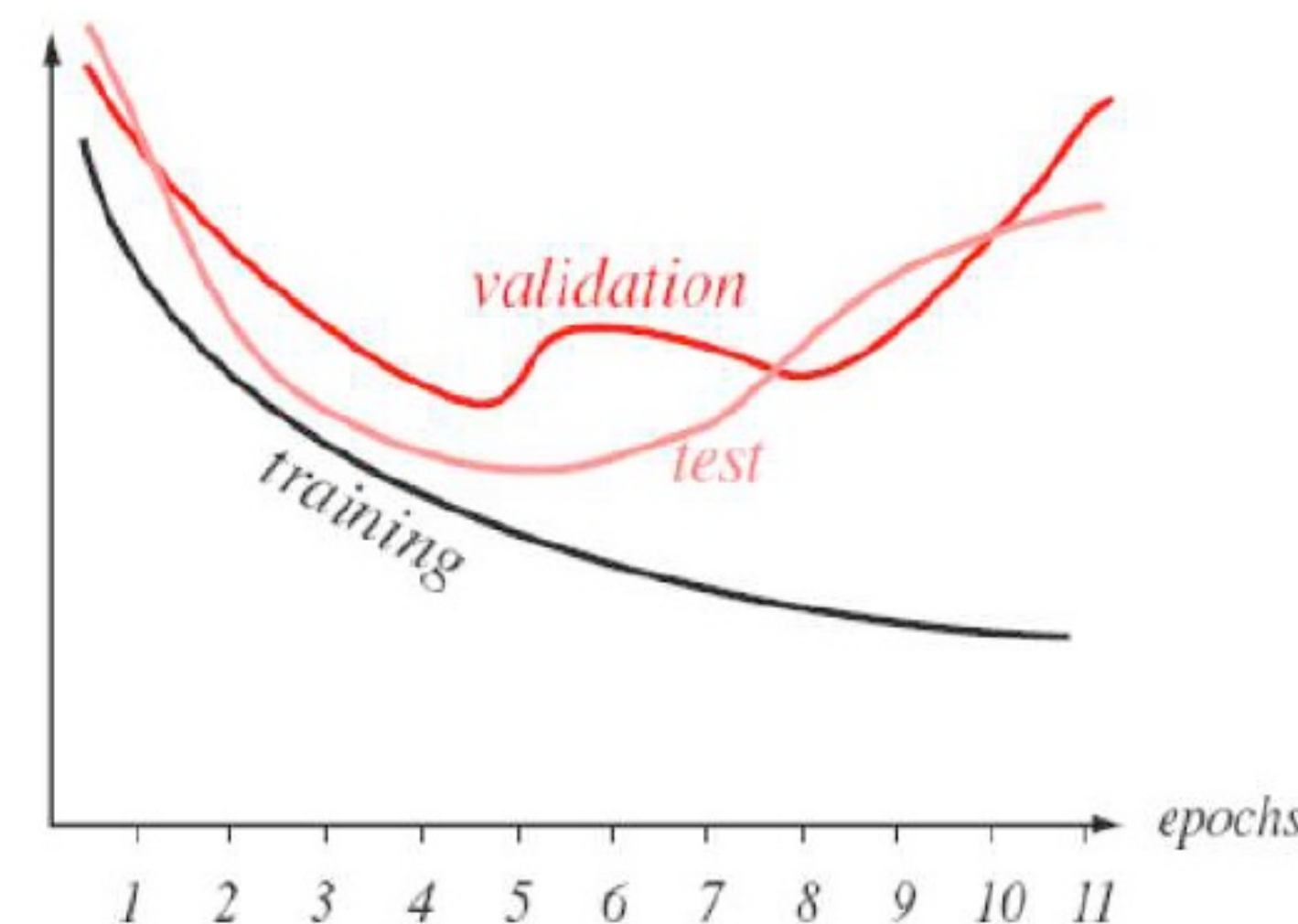
- Monitor error on a validation set.
- Store parameters whenever validation error decreases.
- Use parameters of best validation error as final setting.

### Alternative (to make better use of data):

- Use early stopping to determine number of epochs.
- Then retrain with validation set included with the determined number of epochs.

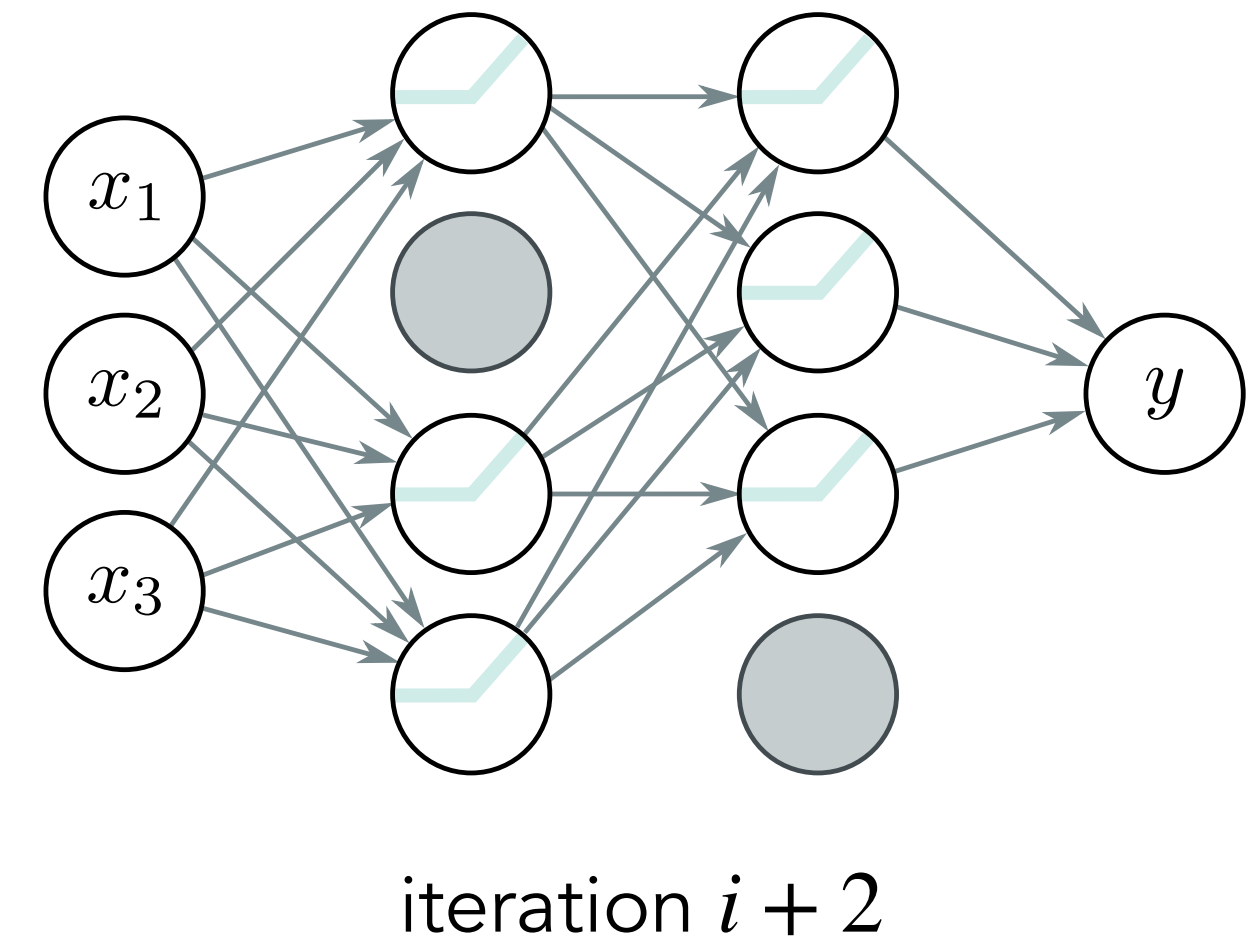
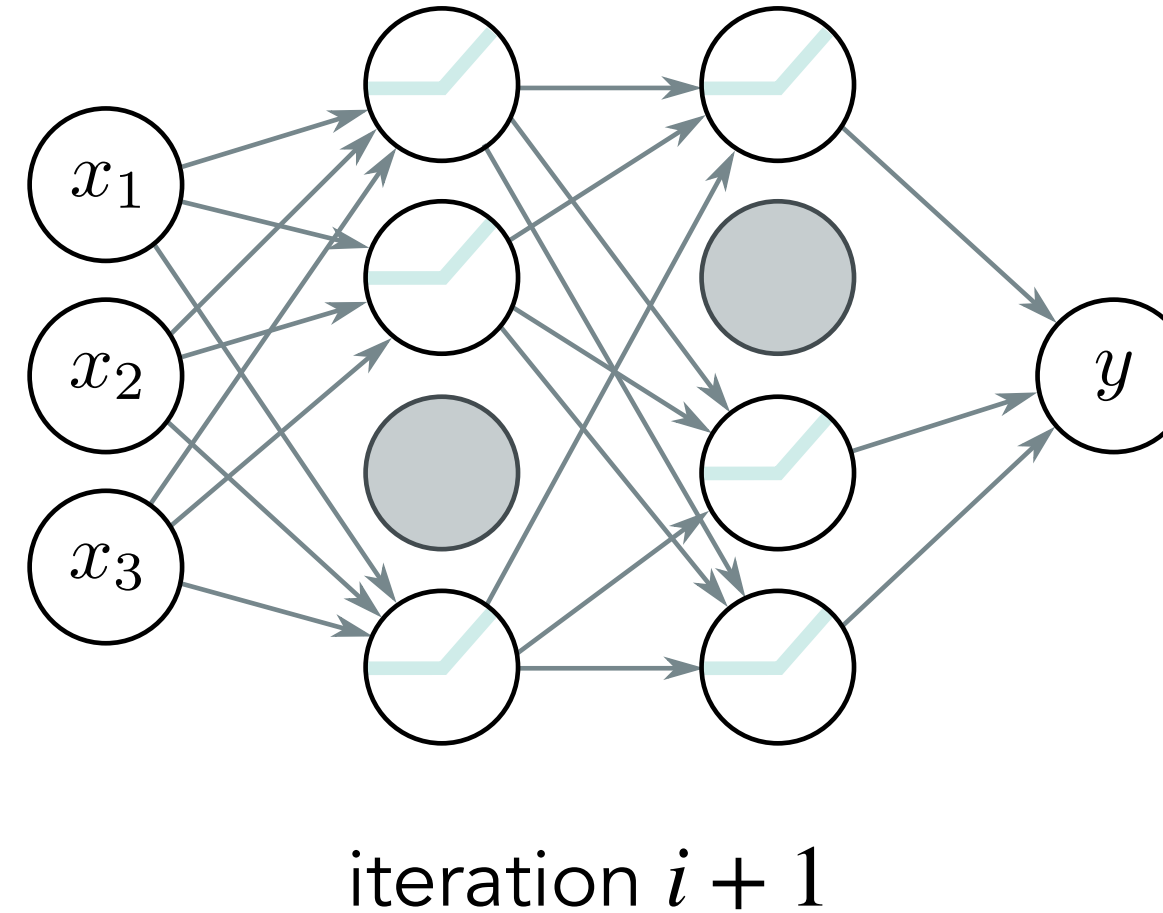
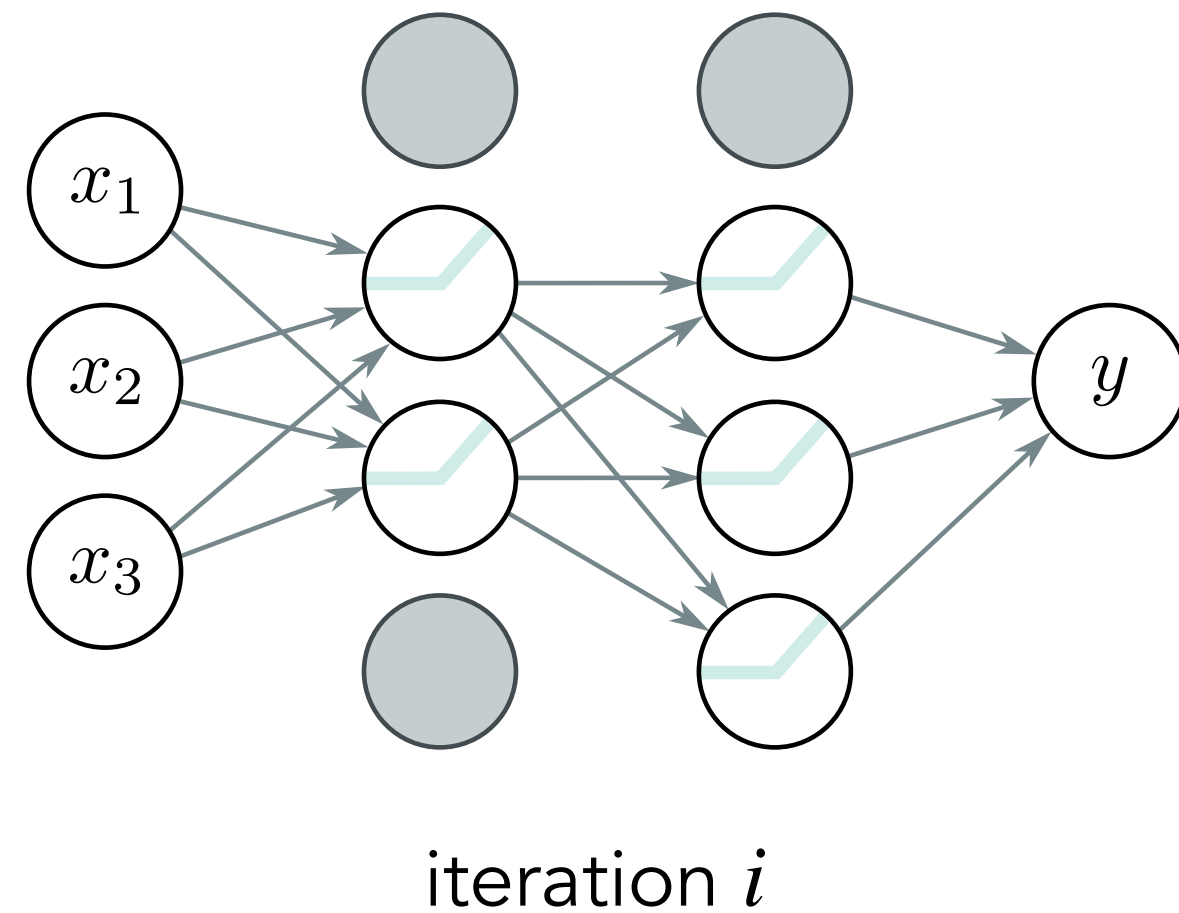
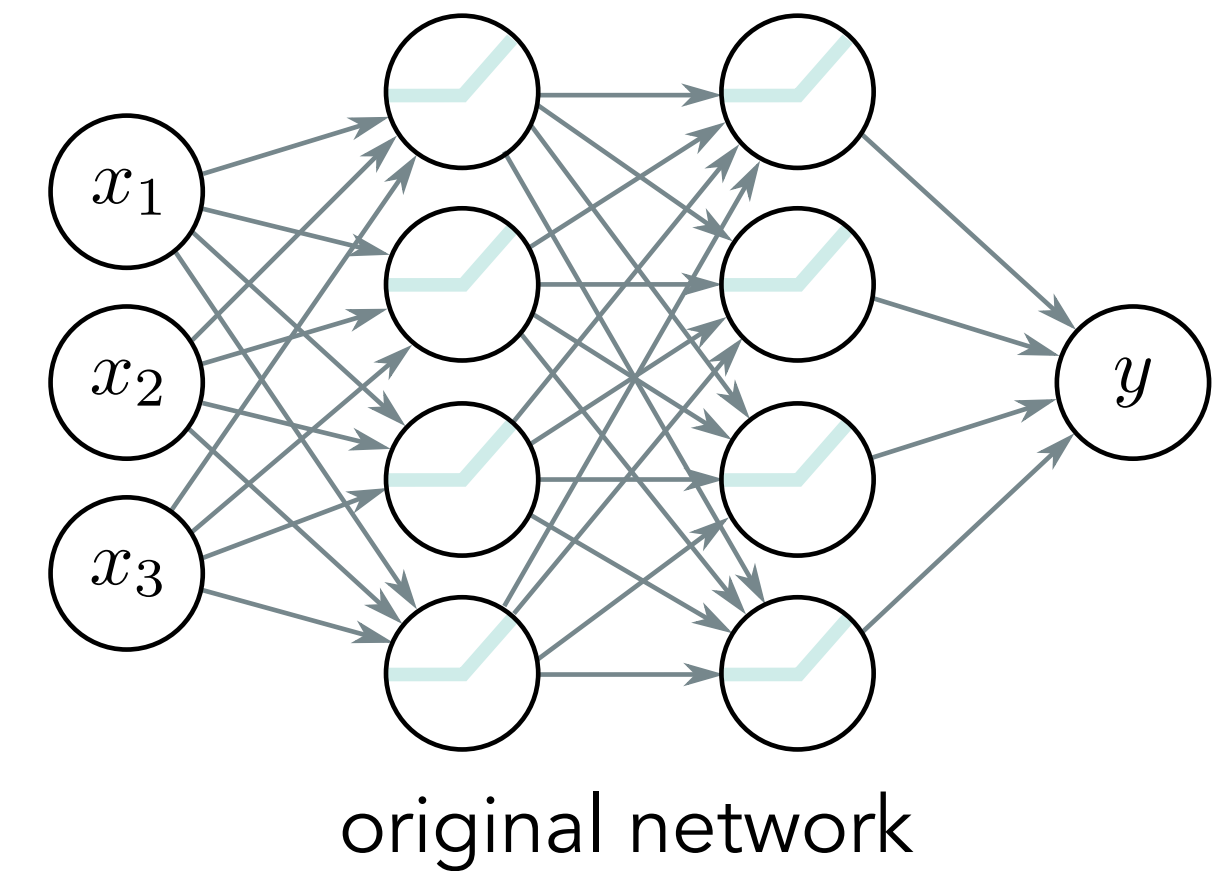
### How early stopping acts as a regularizer:

- We start training with small parameters. With training, parameters and model complexity grows.
- The network learns more and more details of the data, becoming more nonlinear (complex).
- Since we monitor validation error, we can stop at a particularly good point of model complexity.



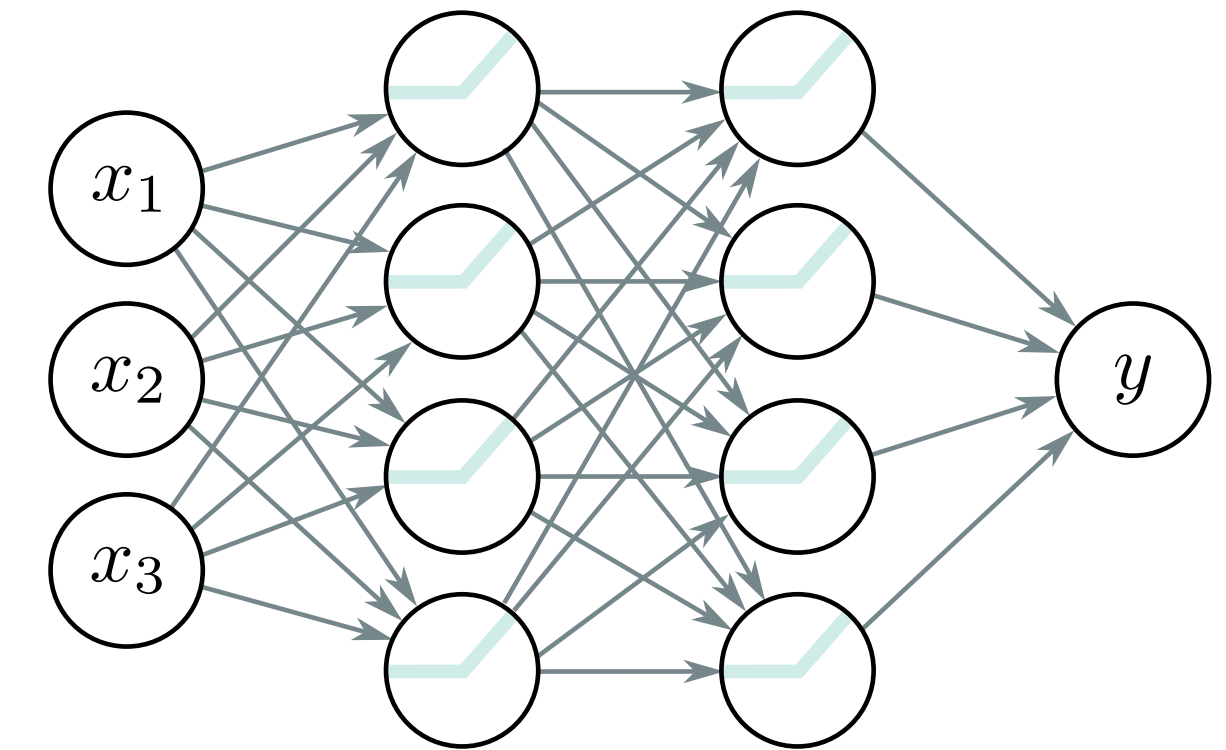
# (3) Dropout

- During training with minibatches, in each minibatch, drop each neuron with probability  $1 - p$  (e.g.,  $p = 0.5$ ).
  - "dropping" means: In both the forward and backward-pass, the neuron is ignored and its output is set to 0.



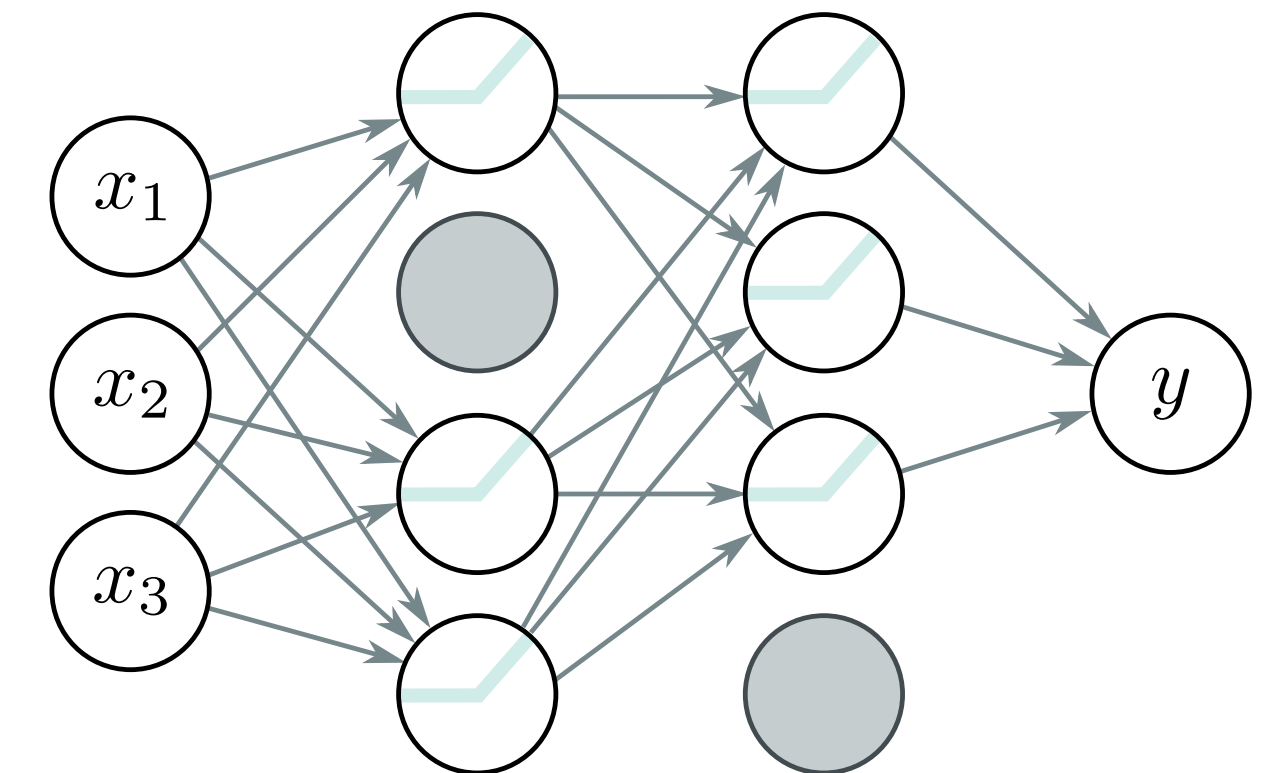
# (3) Dropout

- During training with minibatches, in each minibatch, drop each neuron with probability  $1 - p$  (e.g.,  $p = 0.5$ ).
  - "dropping" means: In both the forward and backward-pass, the neuron is ignored and its output is set to 0.



## Idea:

- Neurons cannot fully rely on the output of other neurons.
- Co-specialization is not possible.
- Sometimes, also inputs are dropped out during training.



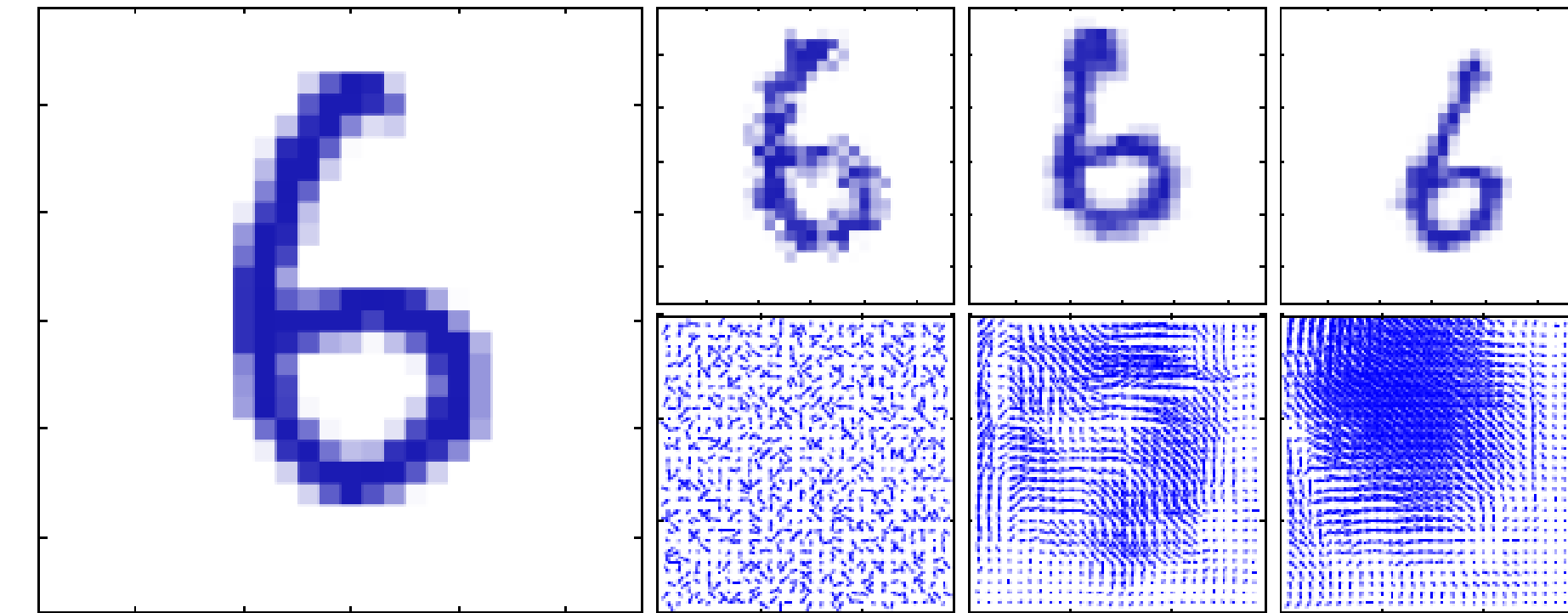
## Final network (application after training):

- Use all weights (and neurons), but rescale:  $w_{ij}^{final} = p w_{ij}$

# (4) Dataset Augmentation

**Augment training data with transformed instances of the original data.**

Boost the size of training set by deformation of input samples.



# (4) Dataset Augmentation

Augment training data with transformed instances of the original data.

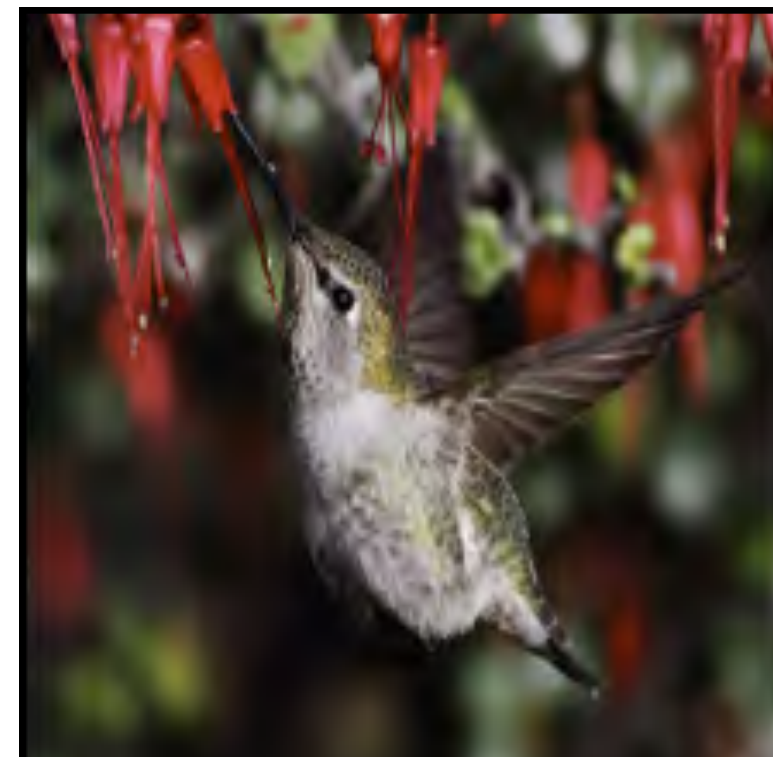
e.g., image classification: boost the size of training set with common image transformations.



original input



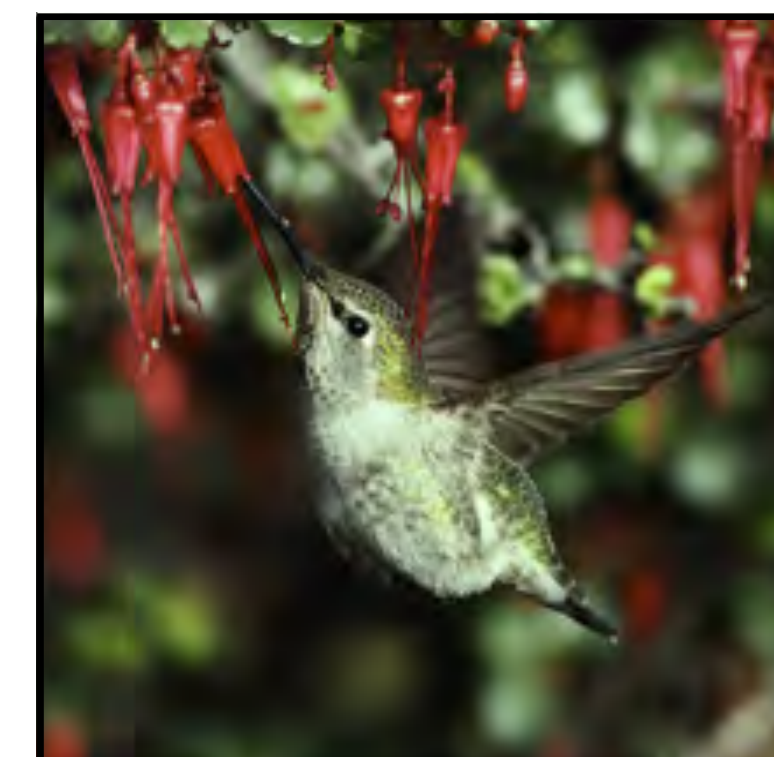
horizontal flip



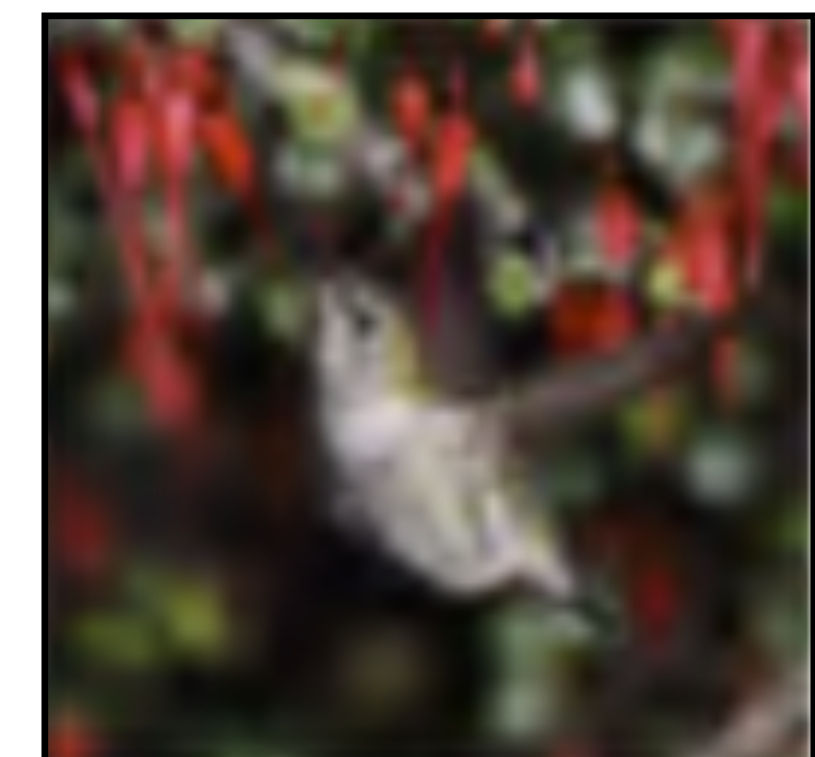
vertical stretch



rotate and crop



color balance

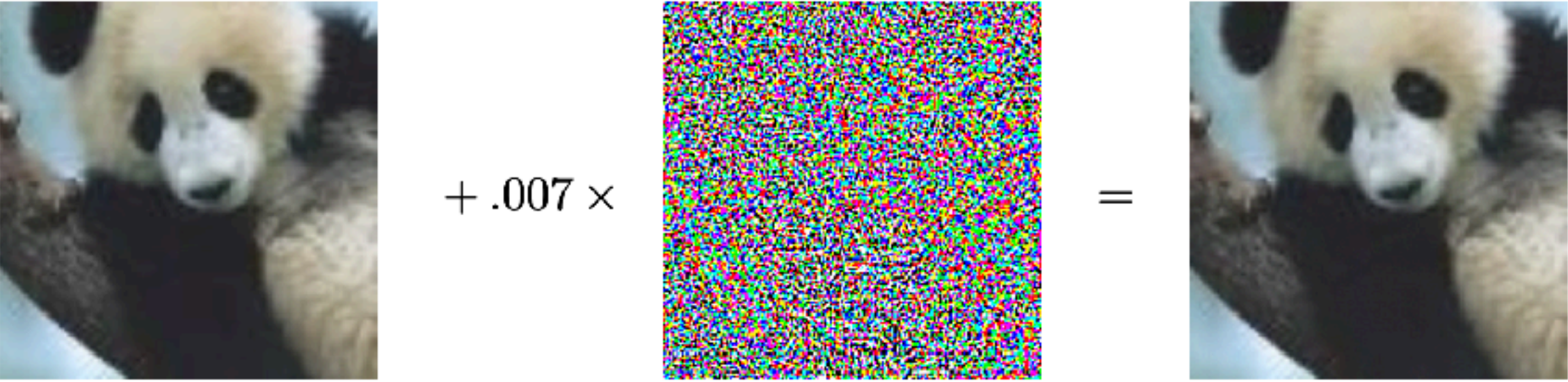


blur

# Adversarial Training

## Adversarial example

- Small (and human-imperceptible) changes in inputs can produce different outputs.



$x$   
“panda”  
57.7% confidence

$+ .007 \times$

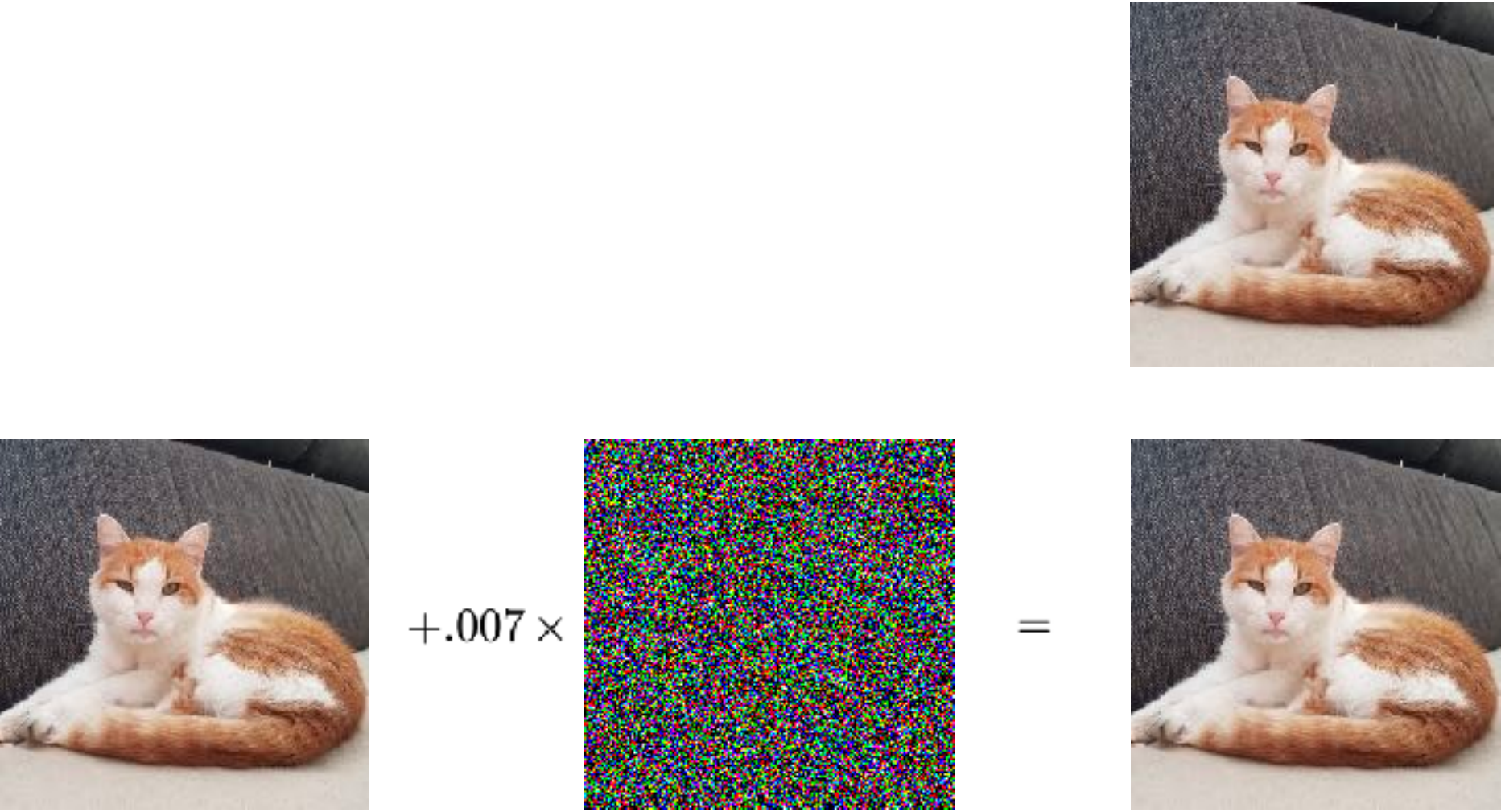
$\text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f_{\theta}(\mathbf{x}), y))$   
“nematode”  
8.2% confidence

$=$

$x + \epsilon \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f_{\theta}(\mathbf{x}), y))$   
“gibbon”  
99.3 % confidence

## Adversarial training

- During training, seek for adversarial examples (i.e., examples  $\mathbf{x}'$  nearby a training example  $\mathbf{x}$  where  $y' \neq y$ ).
- Train with input  $\mathbf{x}'$  and target  $y$ .



$\mathbf{x}$

$+ .007 \times$

$\text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f_{\theta}(\mathbf{x}), y))$

$=$

$\mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(f_{\theta}(\mathbf{x}), y))$

“cat”

“cat”

use both during training

# (5) Further Techniques

---

## Label smoothing

- Training labels are also often noisy  $\rightarrow$  We can assume that a training label is correct with probability  $1 - \epsilon$
- Simple implementation: Replace  $\{0,1\}$  targets for  $k$ -softmax output with  $\left\{ \frac{\epsilon}{k-1}, 1 - \epsilon \right\}$  targets.

## Semi-supervised learning

- Use unlabeled data to obtain good representation of examples.
- Use labeled examples for classification.

## Multi-task learning (auxiliary training)

- Pooling examples out of several tasks (e.g., train lower layers on several tasks, upper layers are task-specific)

## Parameter sharing

- Some parameters can be constrained to have the same value.
- *(more on Convolutional Neural Networks...)*

# Today

---

- ☑ Regularization
  - ☑ Parameter Norm Penalties
  - ☑ Early Stopping
  - ☑ Dropout
  - ☑ Dataset Augmentation
  - ☑ Further Techniques

## Questions?