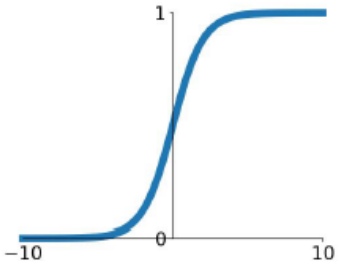
Training Neural Networks

Activation function

Activation functions

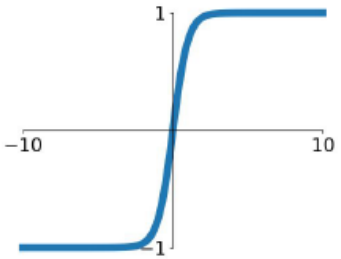
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



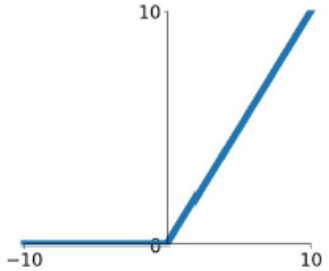
tanh

$$\tanh(x)$$



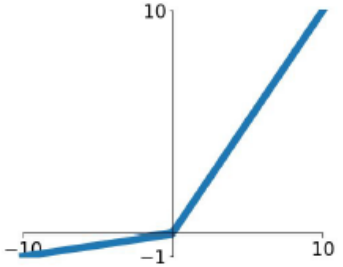
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

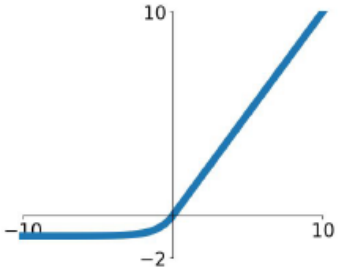


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation functions

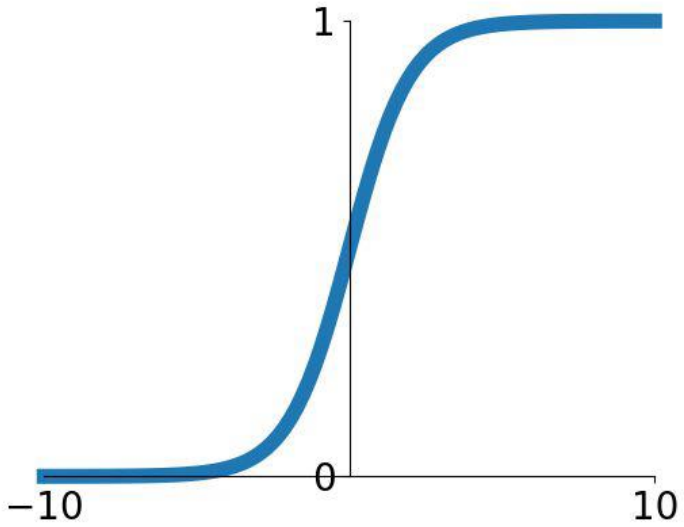
- Squashes numbers to range [0,1]

- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

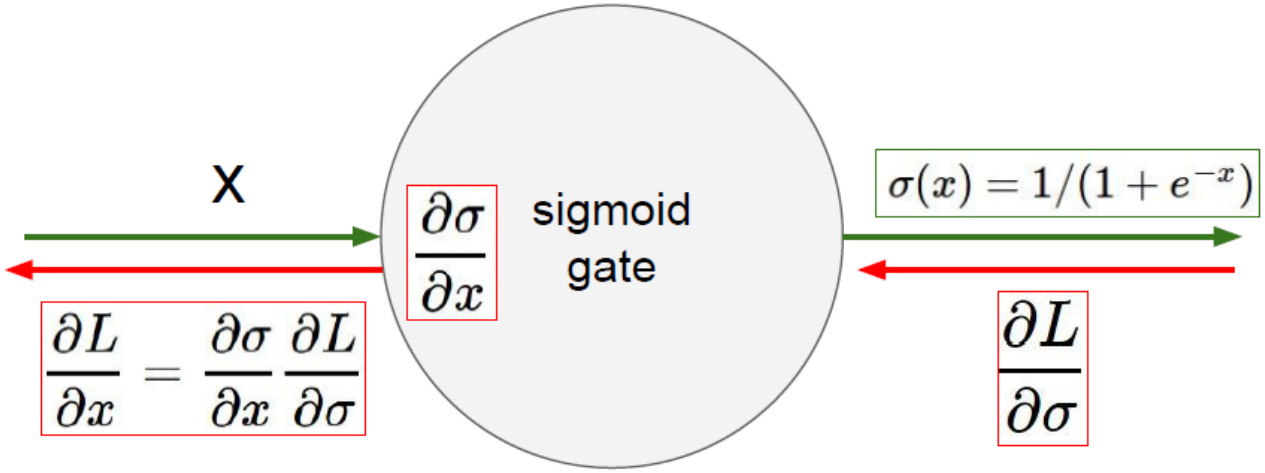
- 3 problems:

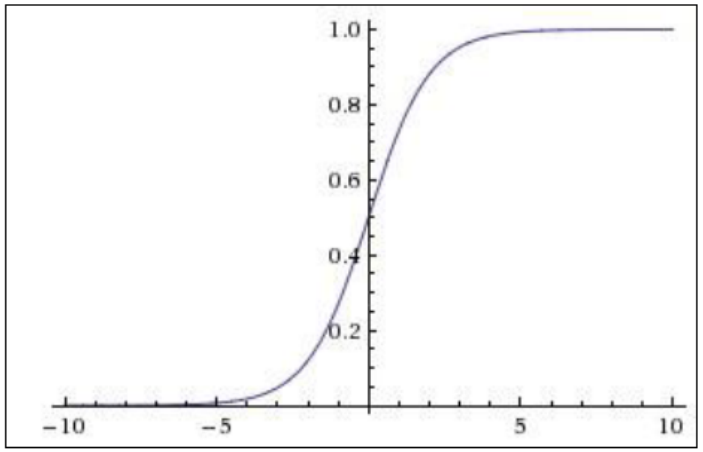
1. Saturated neurons “kill” the gradients

$$\sigma(x) = 1/(1+e^{-x})$$



Sigmoid



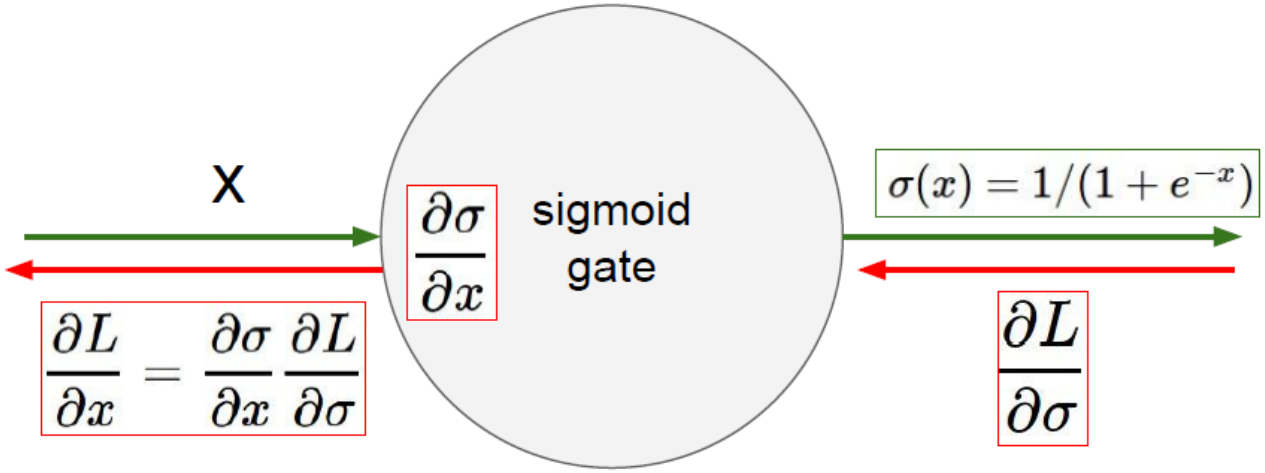


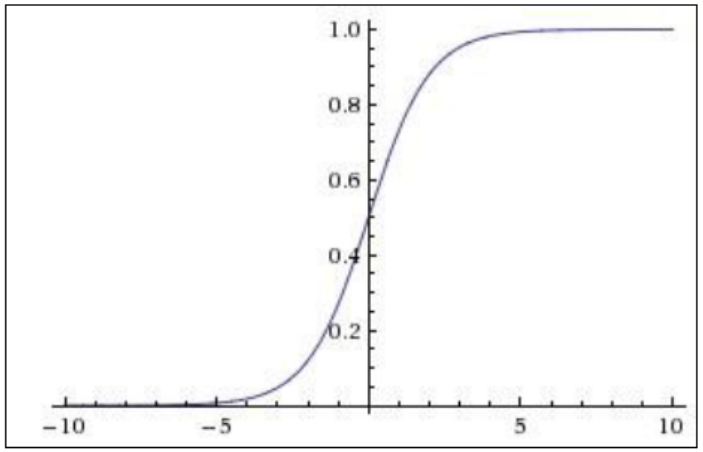
$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x))$$

What happens when x = -10?

$$\sigma(x) = \sim 0$$

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x)) = 0(1 - 0) = 0$$

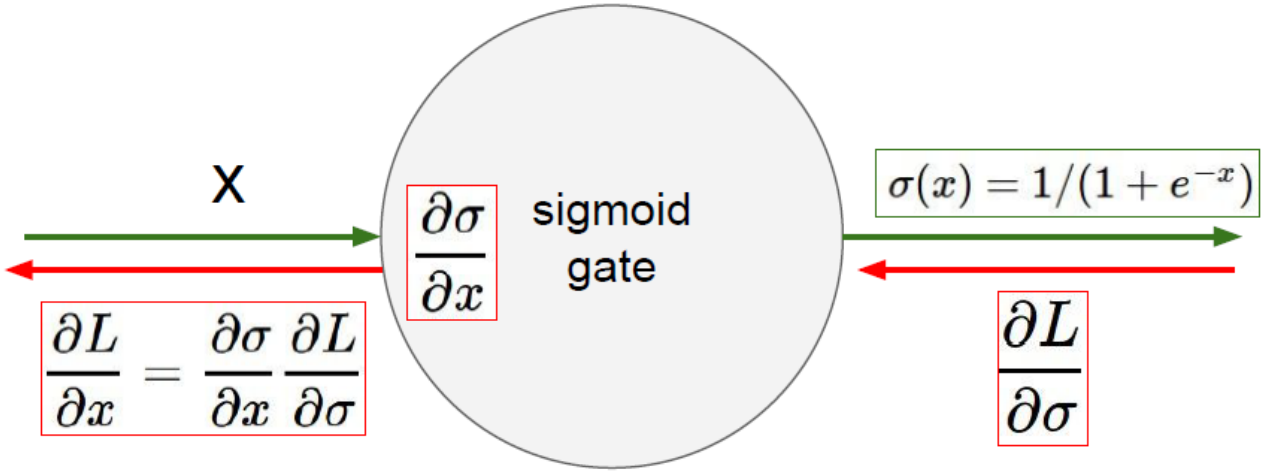


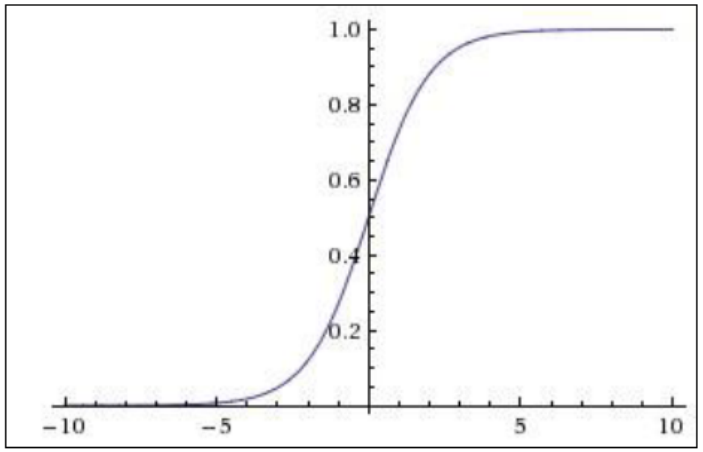


$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x))$$

What happens when x = -10?

What happens when x = 0?



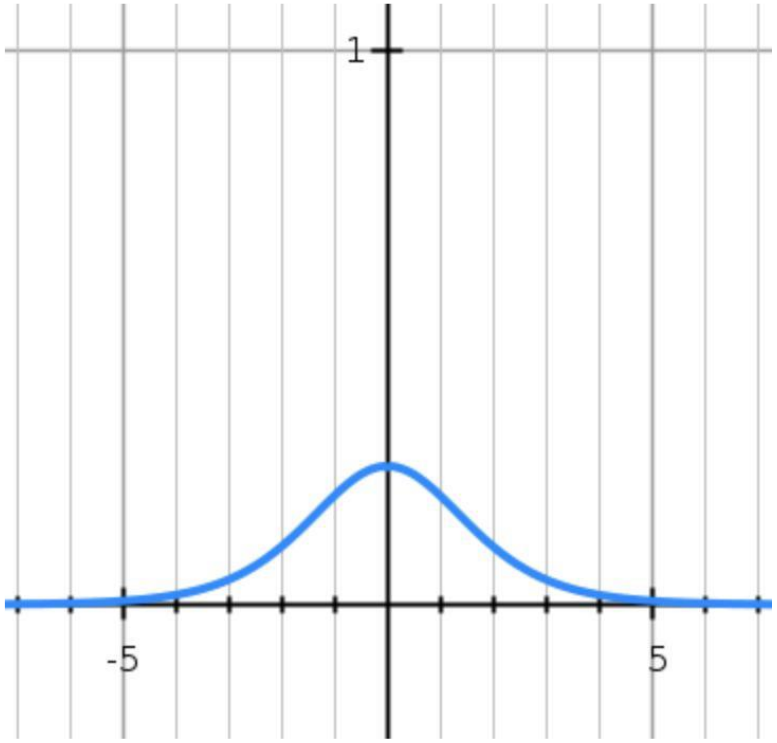


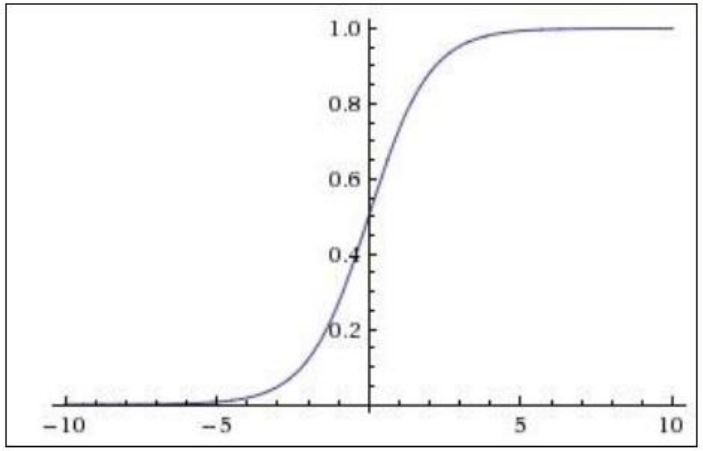
$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x))$$

What happens when x = -10?
What happens when x = 0?
What happens when x = 10?

$$\sigma(x) = \sim 1$$

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x)) = 1(1-1) = 0$$



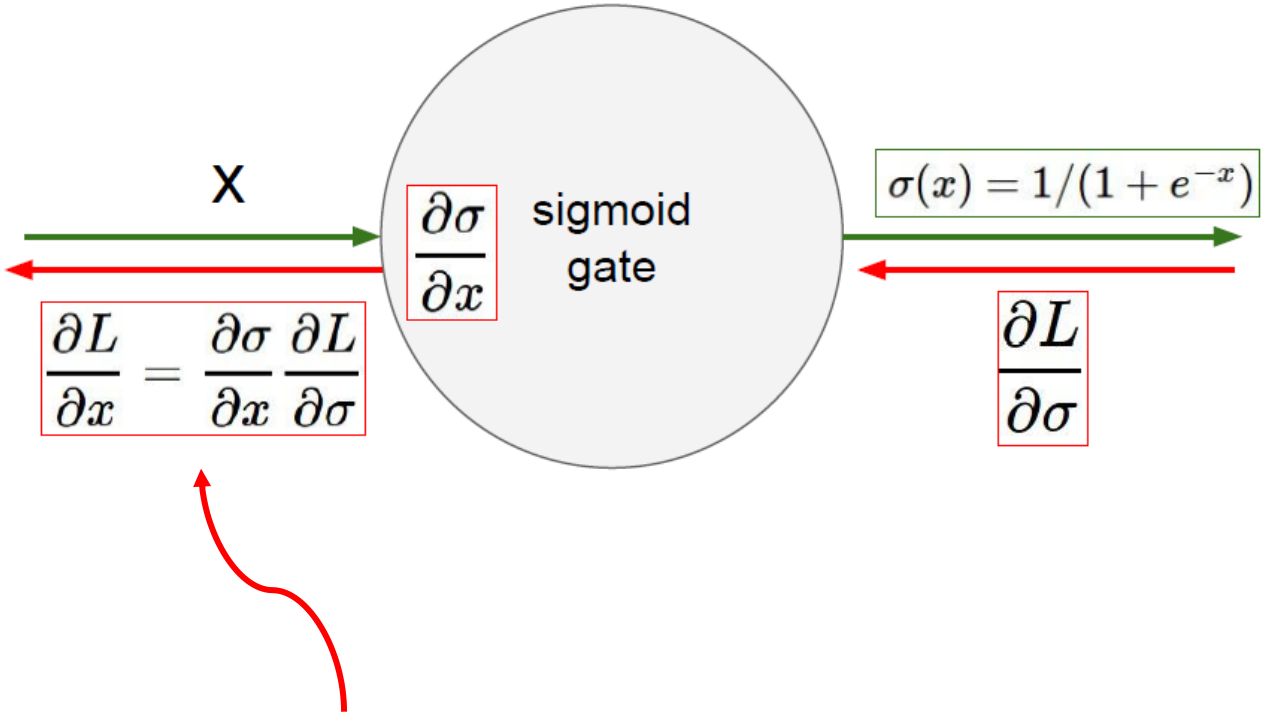


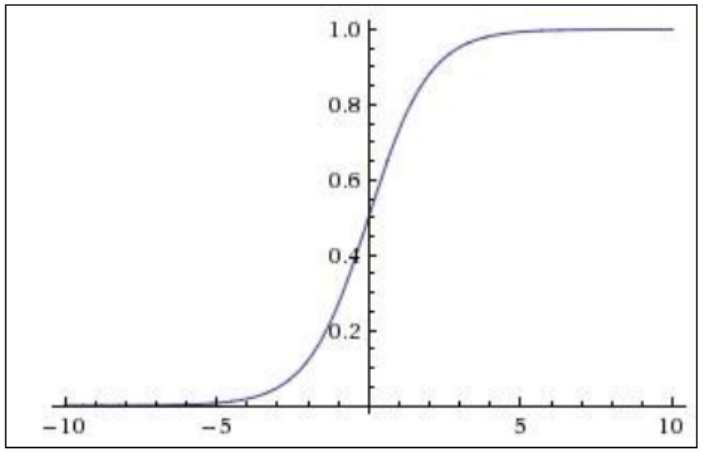
$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x))$$

What happens when x = -10?
What happens when x = 0?
What happens when x = 10?

$$\sigma(x) = \sim 1$$

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x)) = 1(1 - 1) = 0$$





$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x))$$

Why is this a problem?

If all the gradients flowing back will be zero and weights will never change

Activation functions

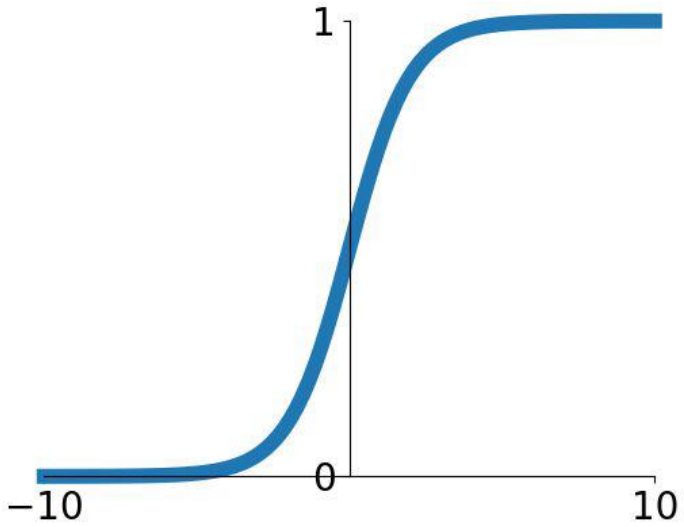
- Squashes numbers to range [0,1]

- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

- 3 problems:

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered

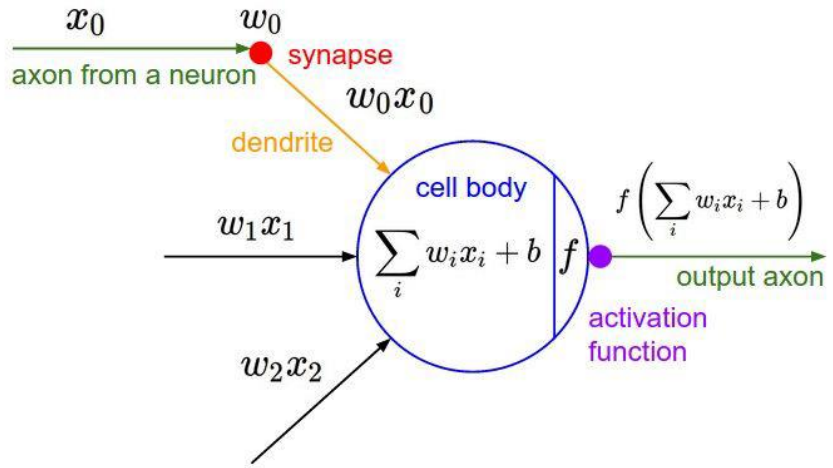
$$\sigma(x) = 1/(1+e^{-x})$$



Sigmoid

Consider what happens when the input to a neuron is always positive...

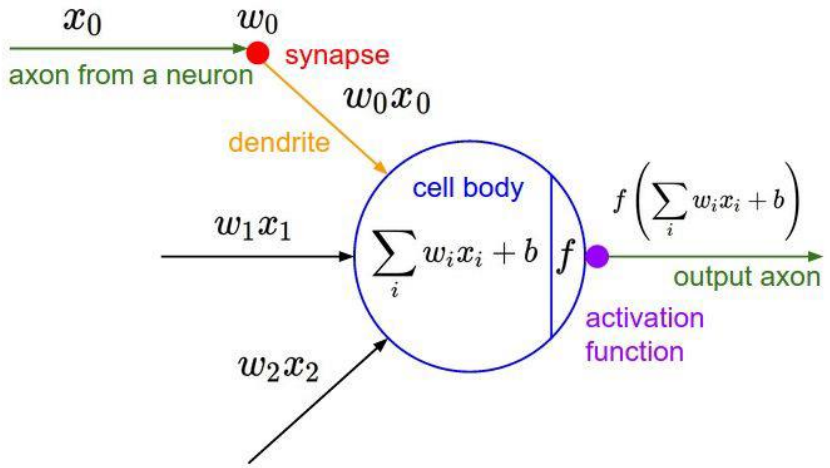
$$f\left(\sum_i w_i x_i + b\right)$$



- If the input to a neuron is always positive, the sigmoid activation will consistently saturate for positive values.
- So, the gradients on the weights may consistently be positive.
- If the gradients are consistently positive, it could lead to a situation where all weight updates are consistently in the same direction, causing convergence issues.

Consider what happens when the input to a neuron is always positive...

$$f\left(\sum_i w_i x_i + b\right)$$



the gradients on **w** ?

Always all positive or all negative :(

Activation functions

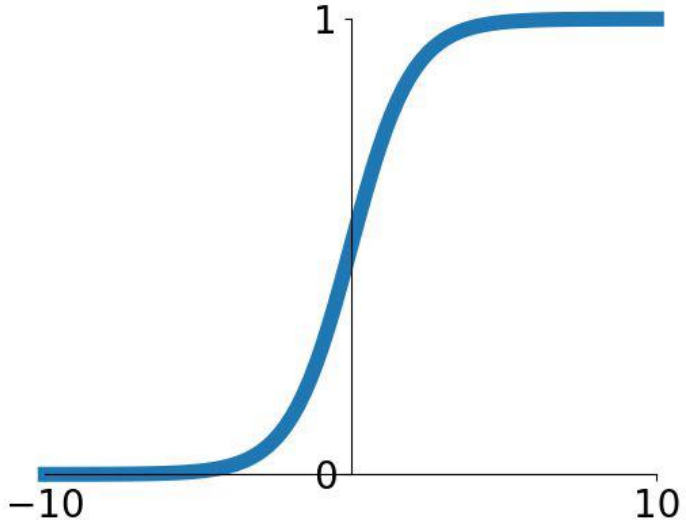
- Squashes numbers to range [0,1]

- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

- 3 problems:

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

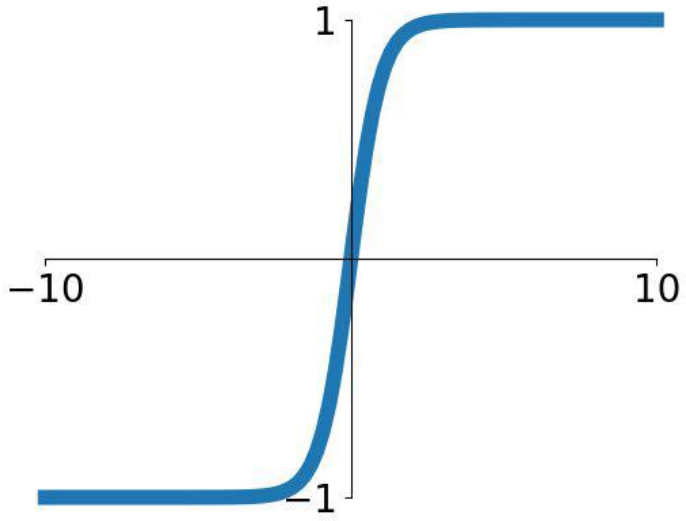
$$\sigma(x) = 1/(1+e^{-x})$$



Sigmoid

Activation functions

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

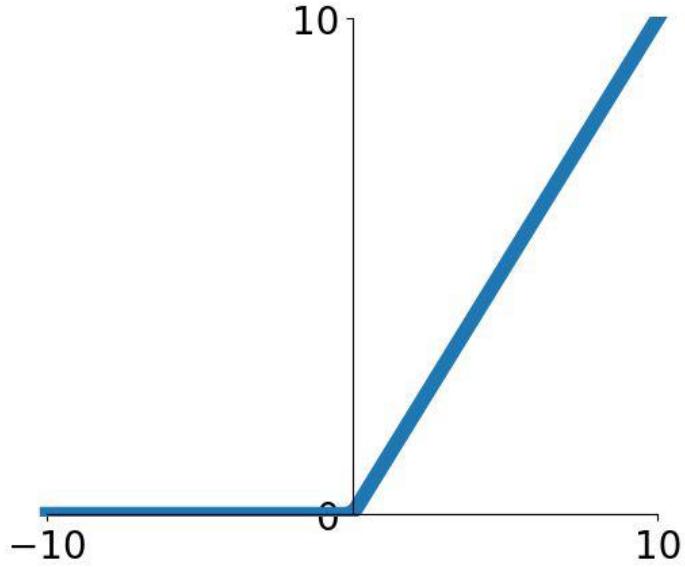


tanh(x)

Activation functions

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

$$f(x) = \max(0, x)$$



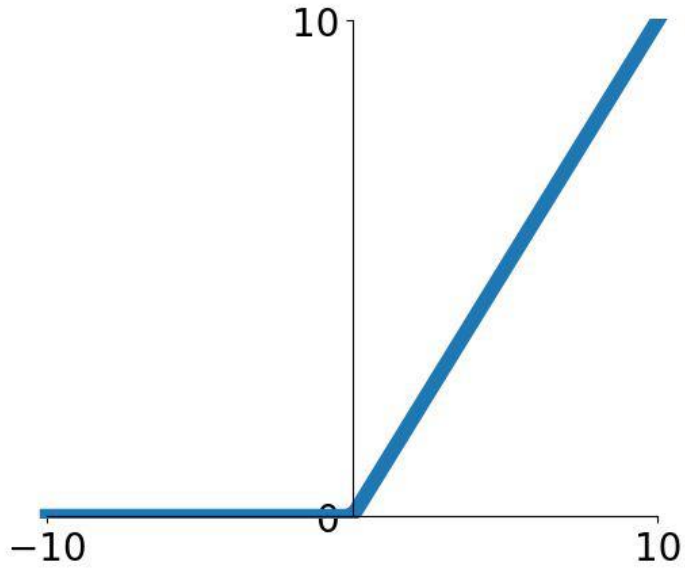
ReLU
(Rectified Linear Unit)

Activation functions

$$f(x) = \max(0, x)$$

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

- Not zero-centered output
- ReLU neuron can become "dead"

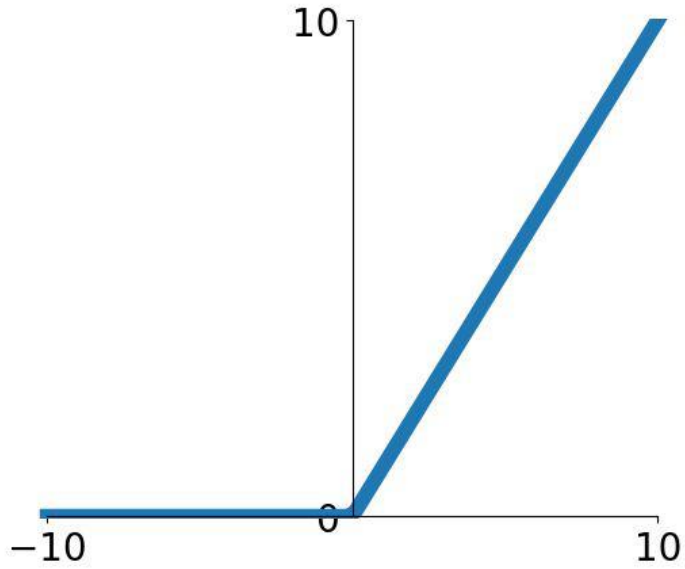


ReLU
(Rectified Linear Unit)

Activation functions

- The term "dead ReLU" refers to a situation where a ReLU neuron always outputs zero (never activates) for any input.
- Since the gradient of the ReLU function is zero for negative inputs, the gradient of the loss with respect to the weights becomes zero, and the weights will not be updated during backpropagation.
- This issue can occur during training, especially if the learning rate is too high or if there's an initialization problem with the weights.

$$f(x) = \max(0, x)$$

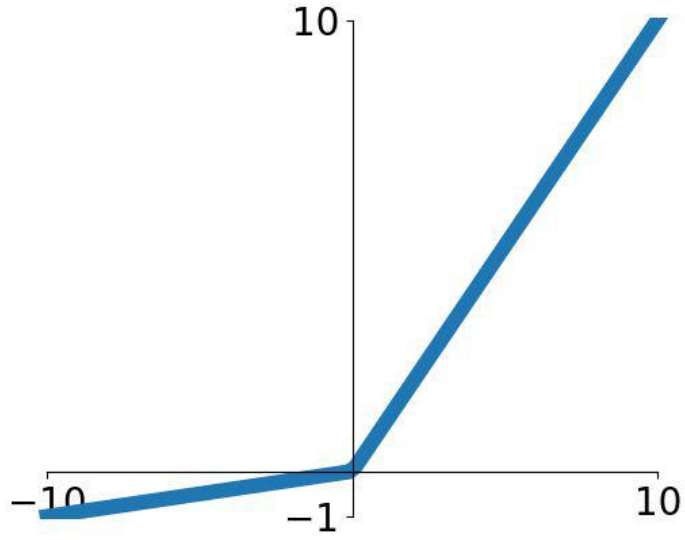


ReLU
(Rectified Linear Unit)

Activation functions

- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not “die”.**

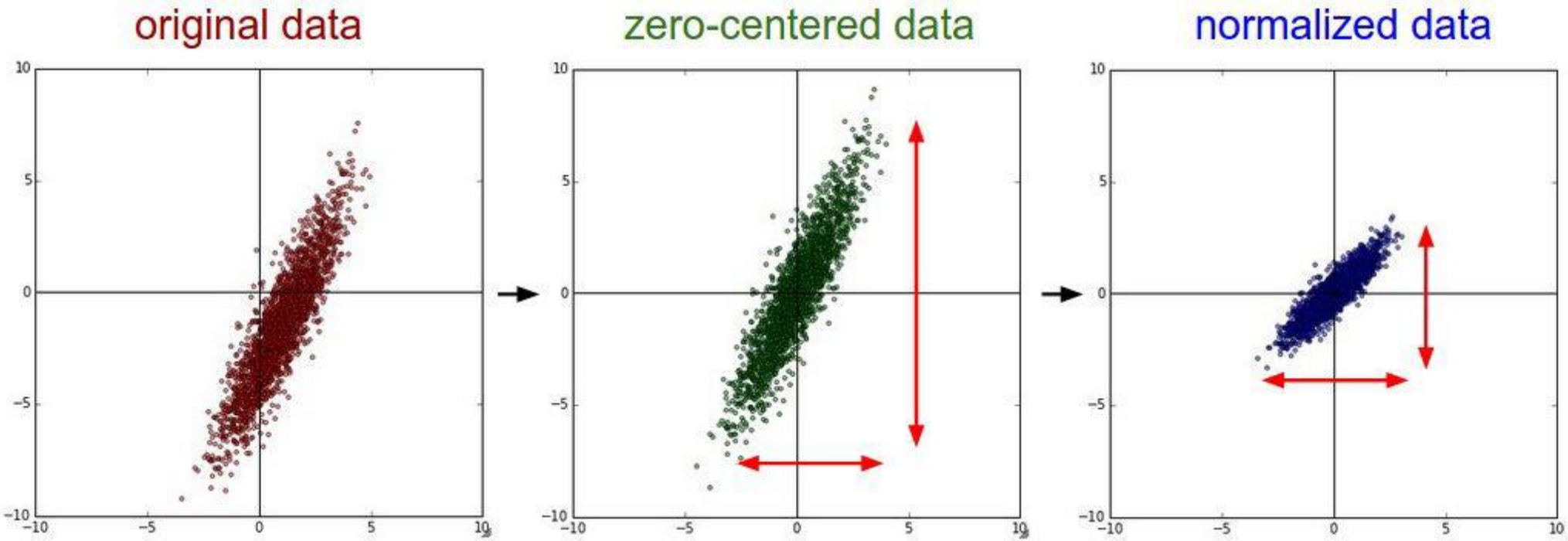
$$f(x) = \max(0.01x, x)$$



Leaky ReLU
(Rectified Linear Unit)

Data Preprocessing

Data preprocessing



Data preprocessing

- Subtract the mean image (e.g. AlexNet)

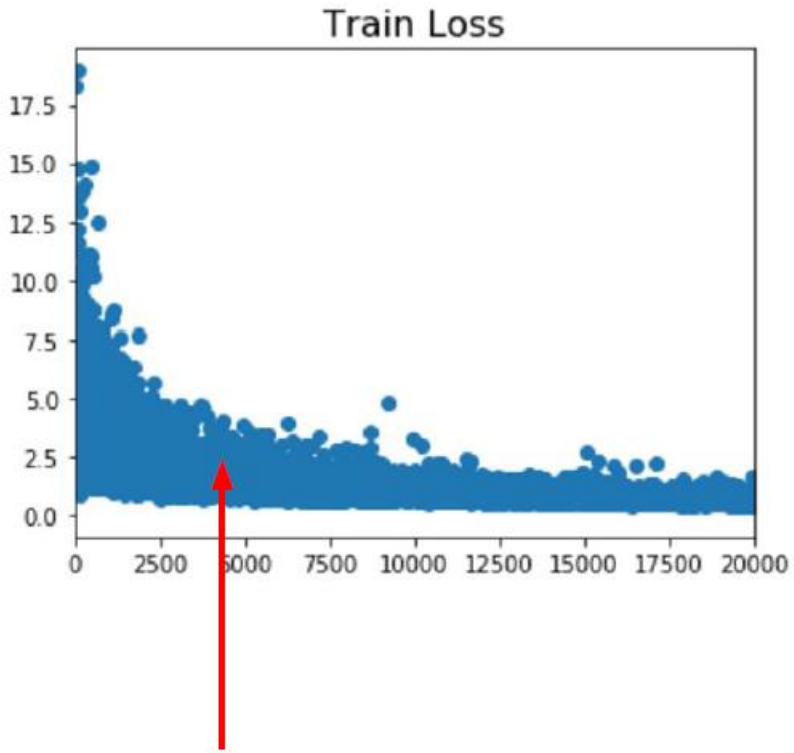
  (mean image = [32,32,3] array)
- Subtract per-channel mean (e.g. VGGNet)

  (mean along each channel = 3 numbers)
- Subtract per-channel mean and Divide by per-channel std (e.g. ResNet)

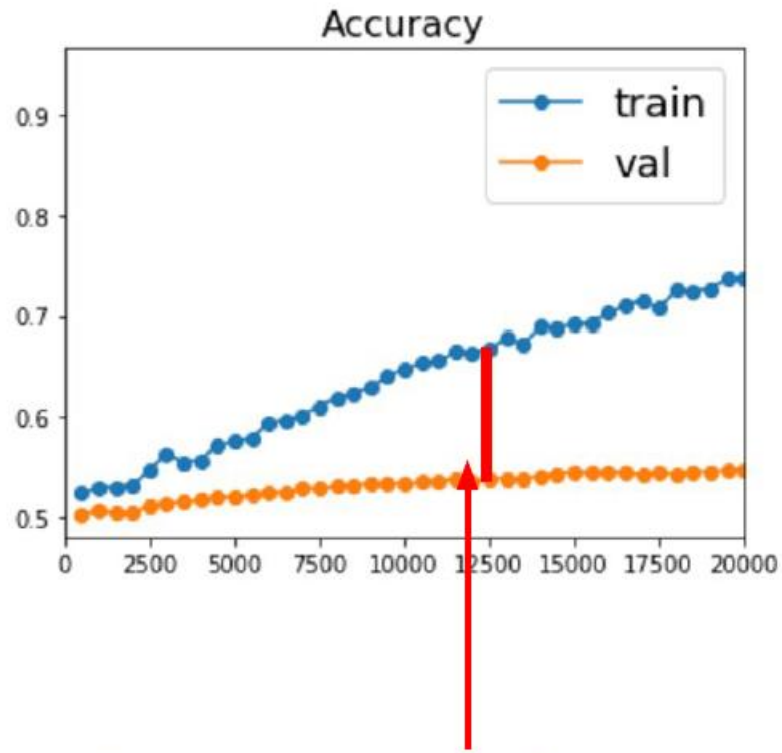
  (mean along each channel = 3 numbers)

Training vs. Testing Error

Training vs Testing Error

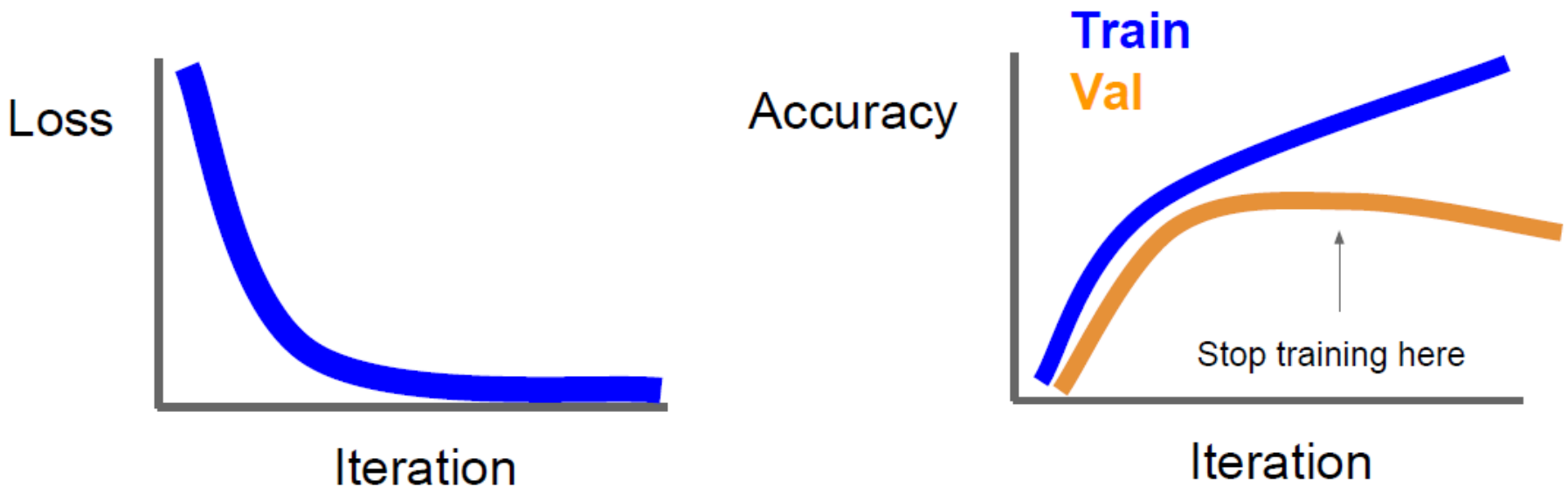


Better optimization algorithms help reduce training loss



But we really care about error on new data - how to reduce the gap?

Early stopping



Stop training the model when accuracy on the validation set decreases
Or train for a long time, but always keep track of the model snapshot that worked best on val

Model Ensembles

1. Train multiple independent models

2. At test time average their results

(Take average of predicted probability distributions, then choose argmax)

- Enjoy 2% extra performance

- How to improve single-model performance and stop overfitting?

Regularization

Regularization: Add term to loss

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

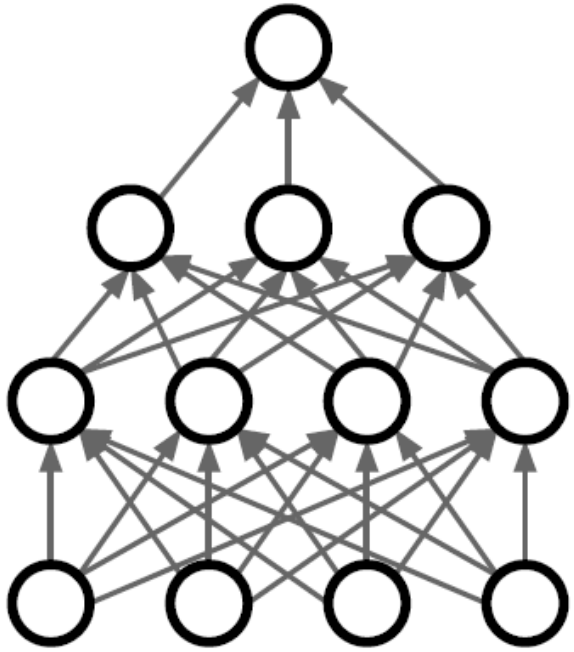
L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$ (Weight decay)

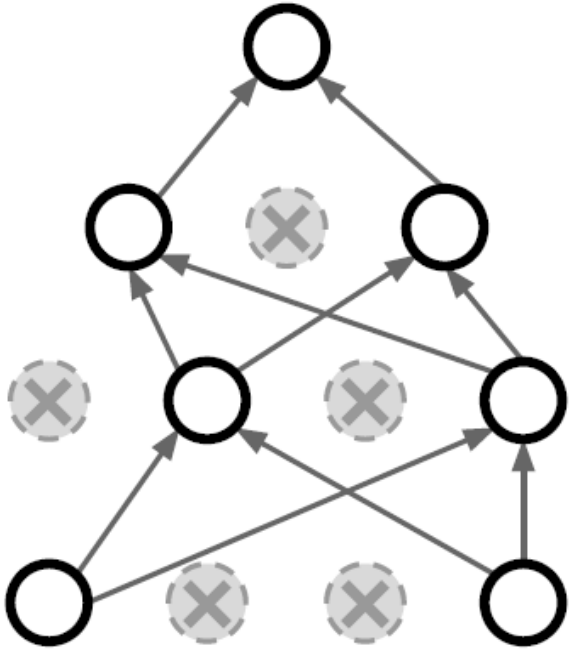
L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

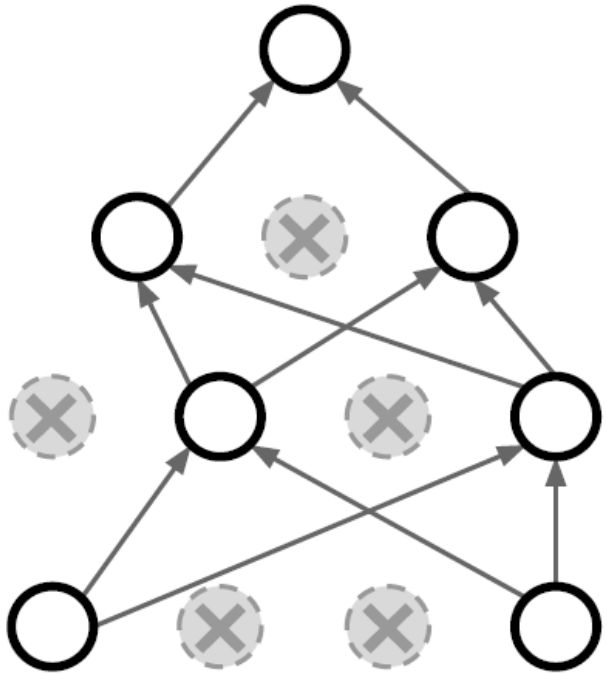
Regularization: Dropout

- In each forward pass, randomly set some neurons to zero Probability of dropping is a hyperparameter (0.5 is common)

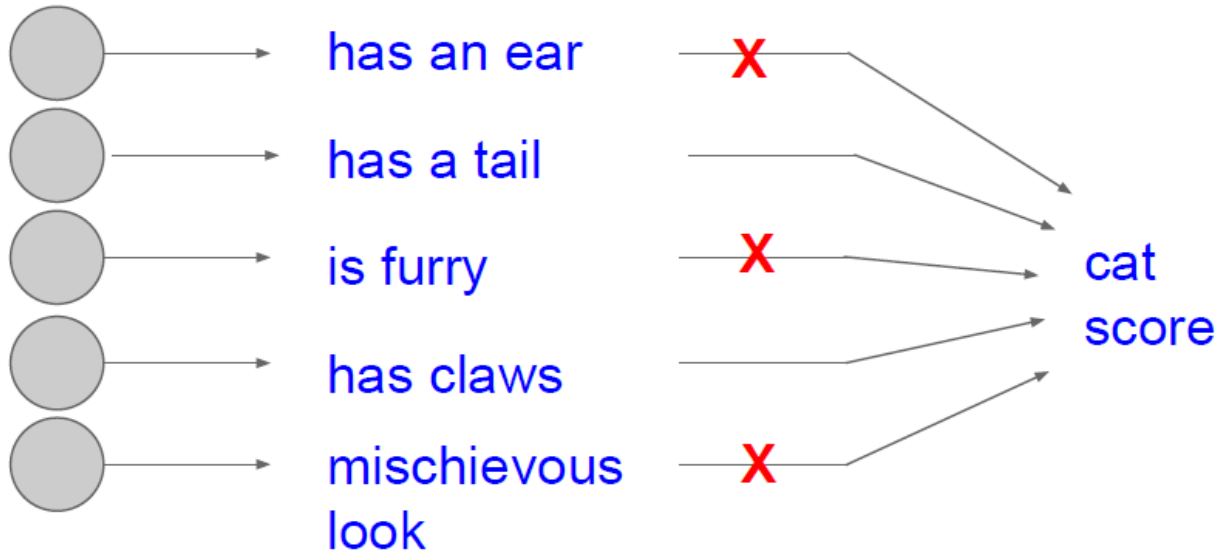




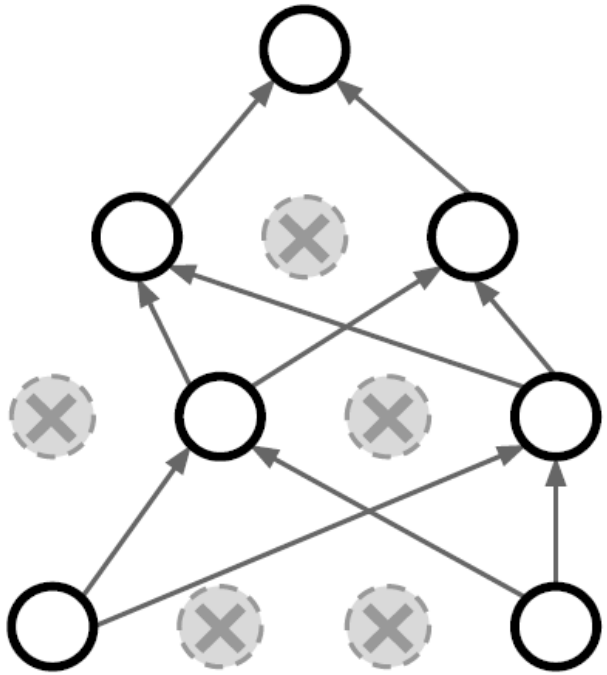
Regularization: Dropout



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



Regularization: Dropout



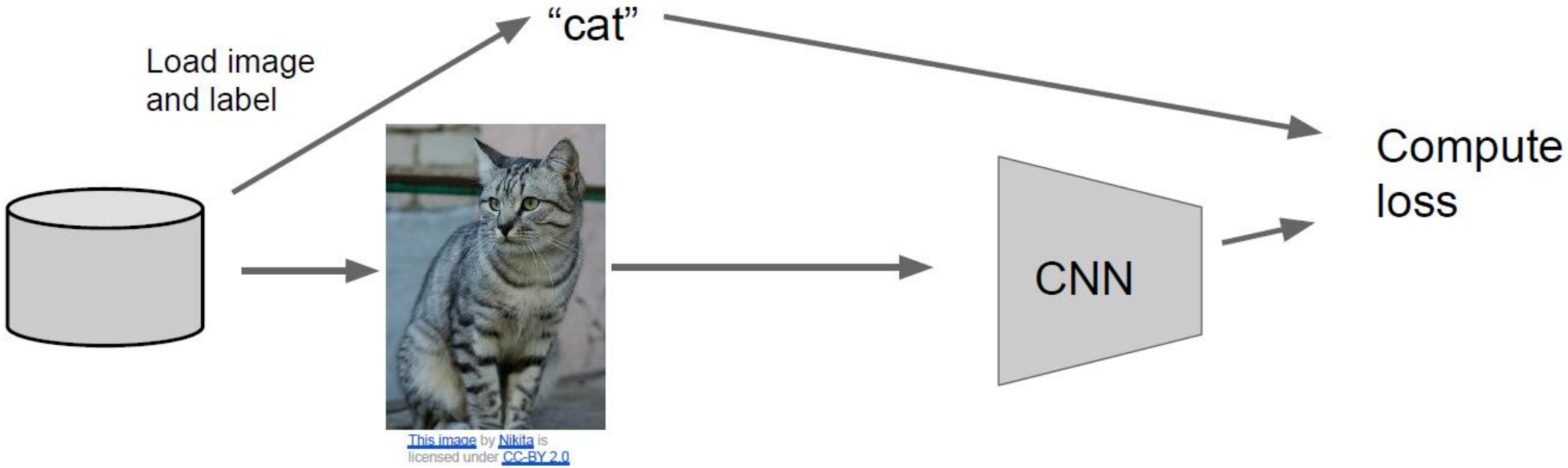
Another interpretation:

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

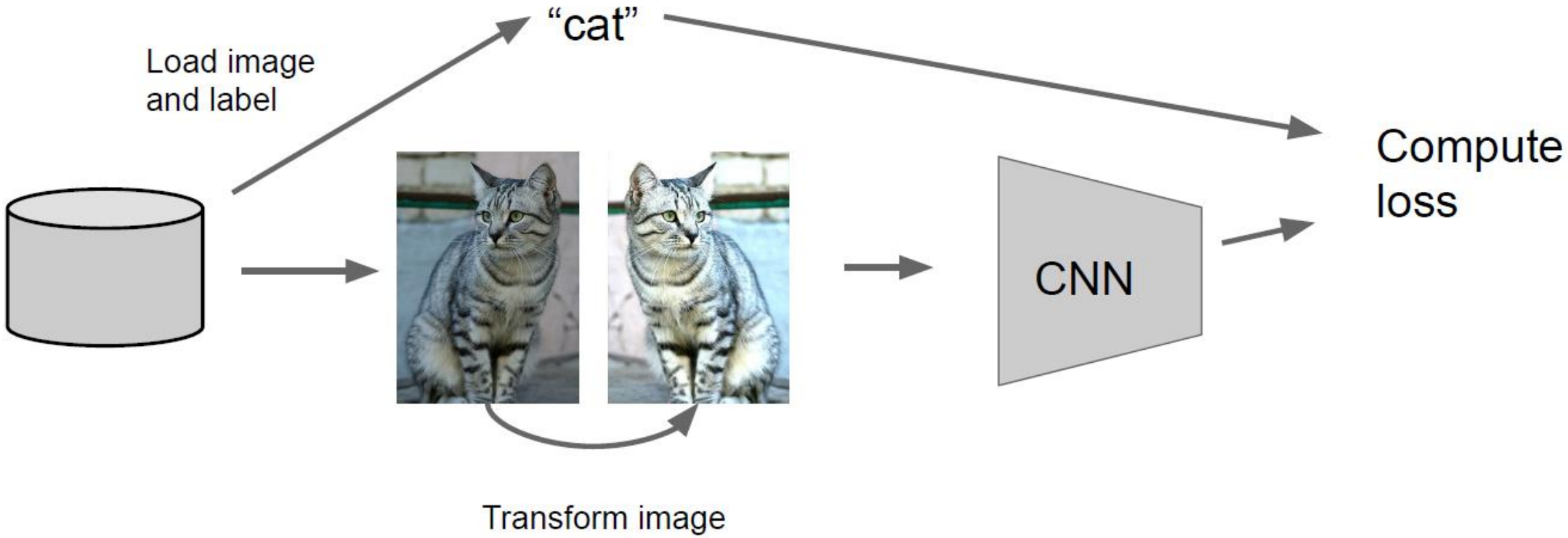
Each binary mask is one model

An FC layer with 4096 units has
$2^{4096} \sim 10^{1233}$ possible masks!
Only $\sim 10^{82}$ atoms in the universe...

Regularization: Data Augmentation

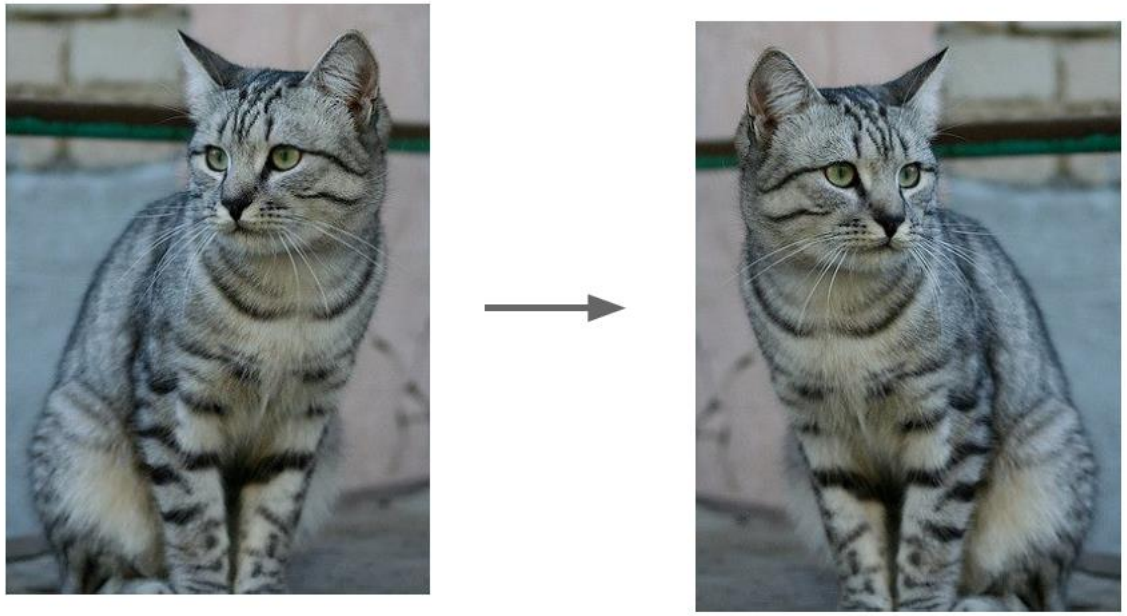


Regularization: Data Augmentation



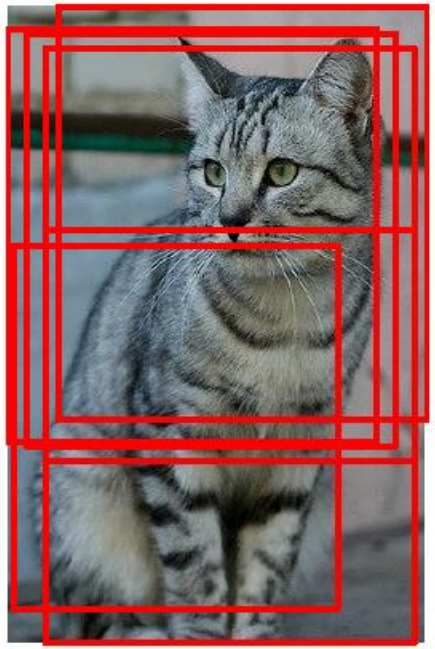
Data Augmentation

Horizontal Flips



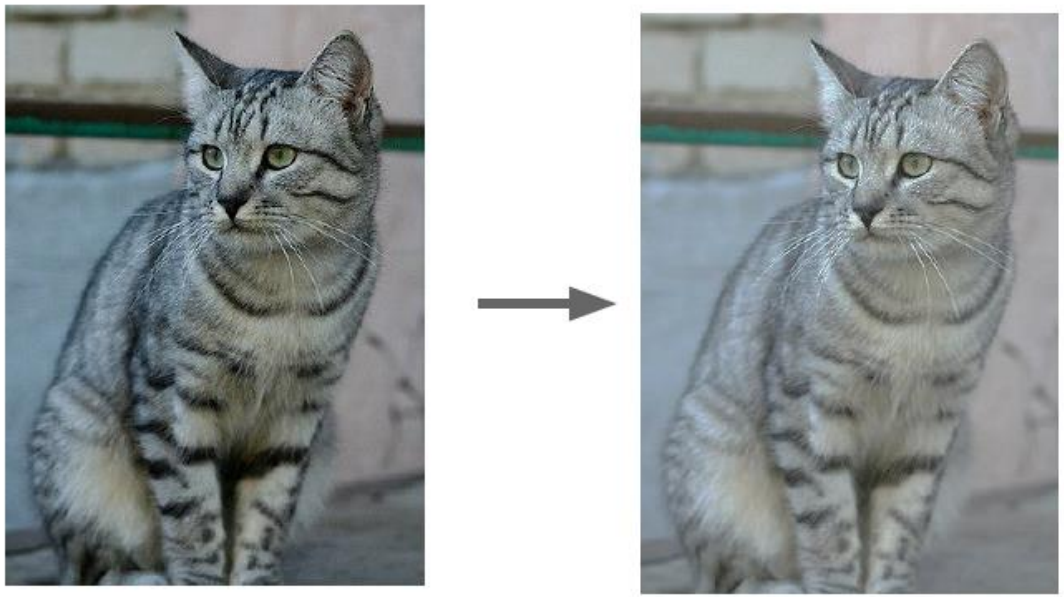
Data Augmentation

sample random crops / scales



Data Augmentation

contrast and brightness



Data Augmentation

- Examples of data augmentations:

- translation
- rotation
- stretching
- Shearing
- …

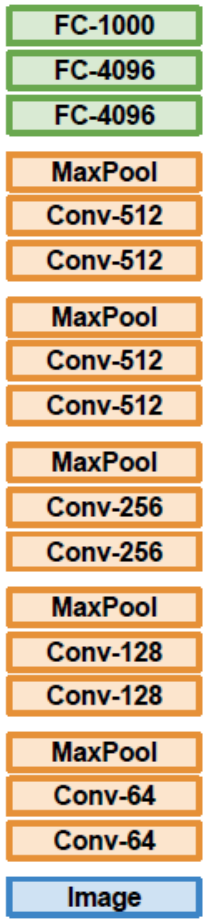
Transfer learning

Transfer learning

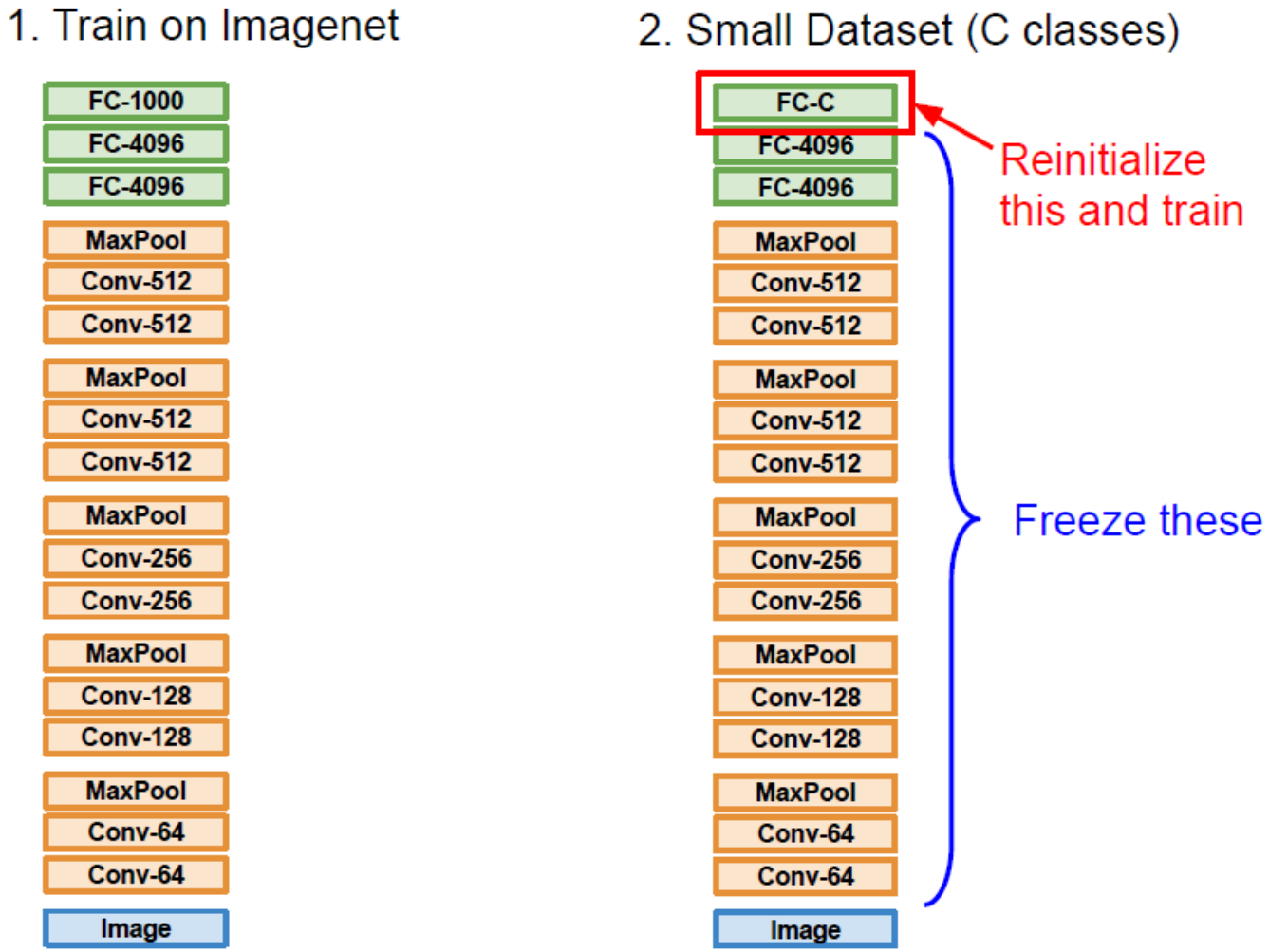
- You need a lot of a data if you want to train/use CNNs?

Transfer learning

1. Train on Imagenet



Transfer learning



Transfer learning