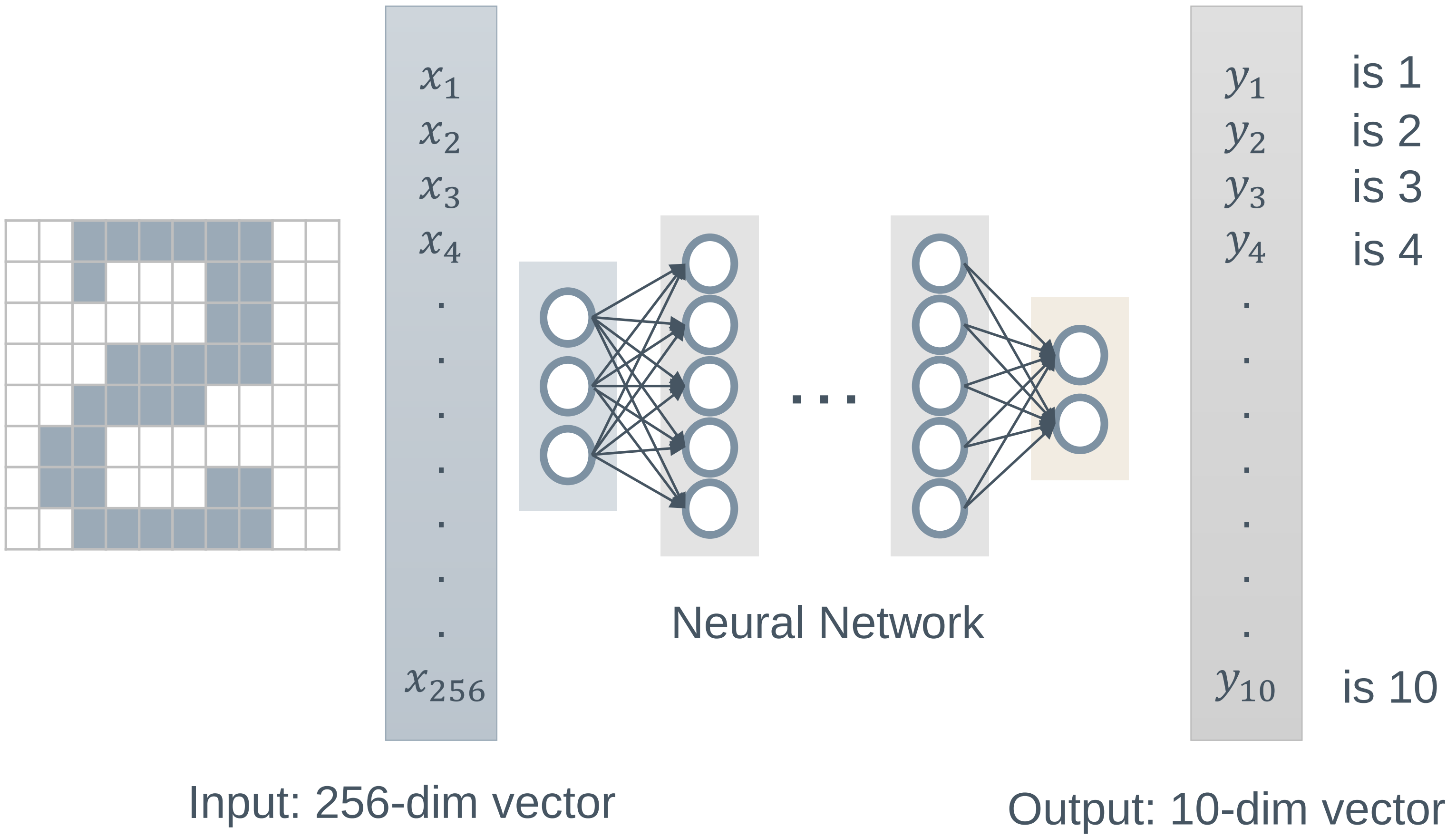
[AI Theory&App]
02 NNBasic

Instructor: Hao-Shang Ma

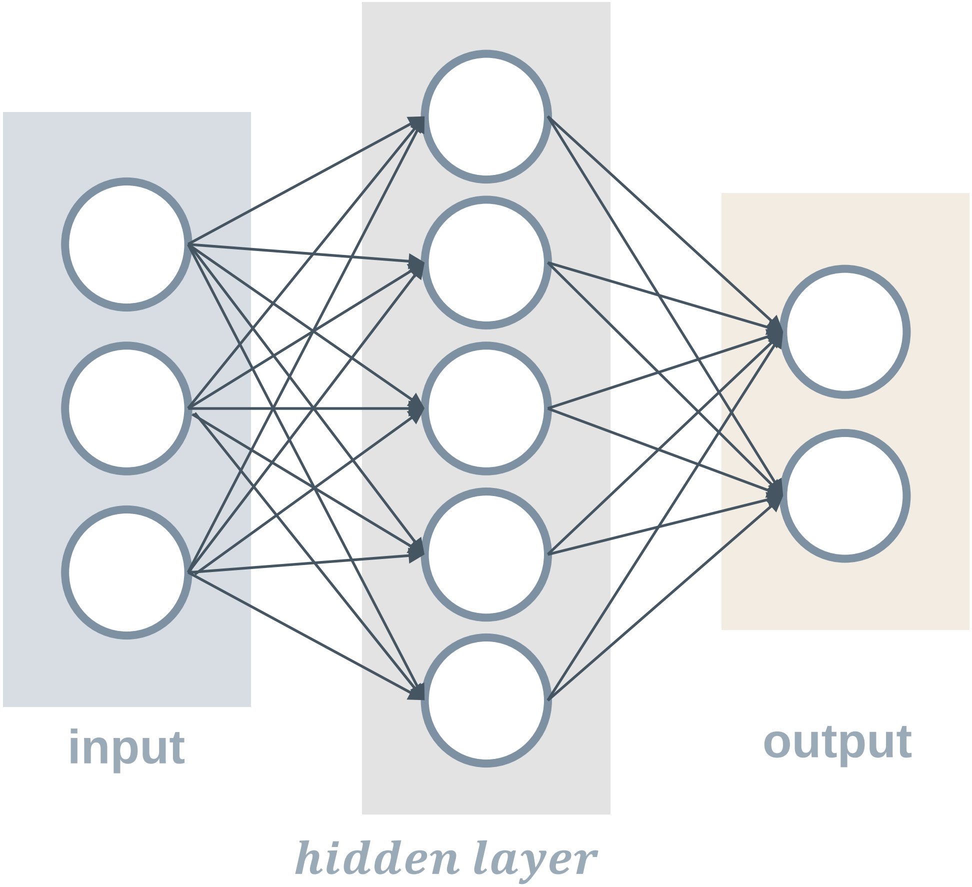
Department Of Computer Science And Information Engineering, NTUST

Introduction to Neural Networks

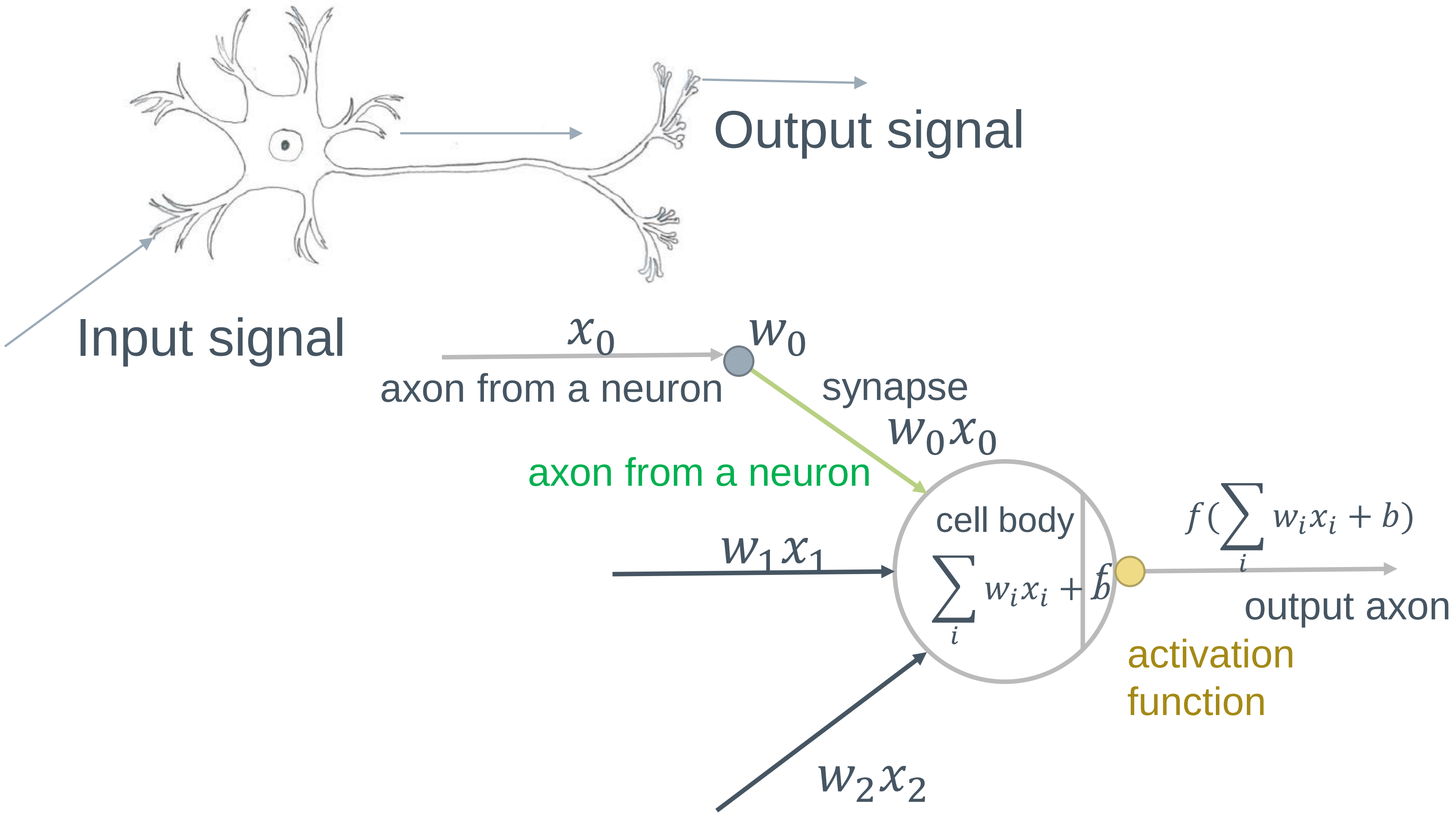
A Brief Example of Neural Networks



Basic Artificial Neural Network

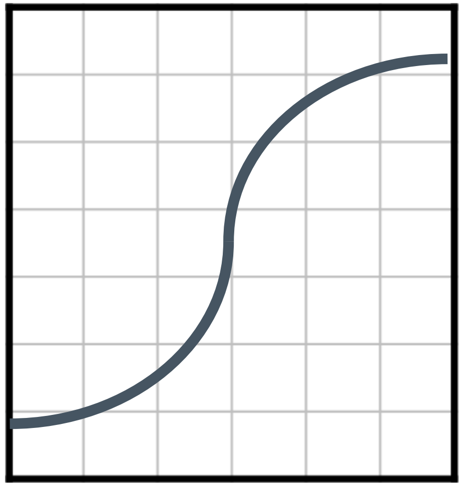


Neuron Design

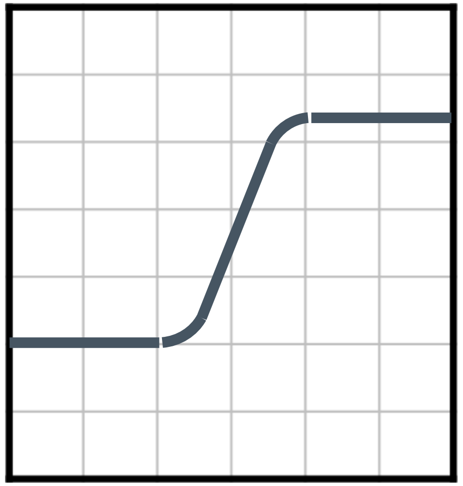


Activation Function for Neurons

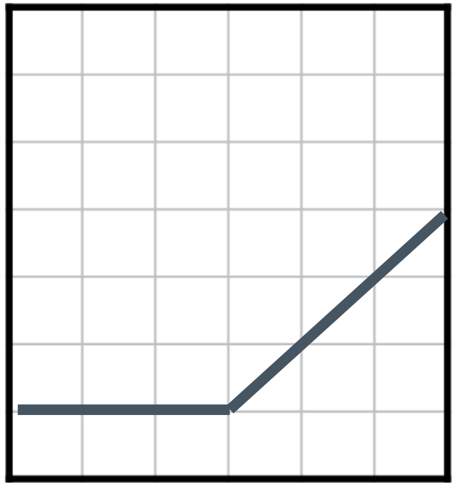
graphs



Sigmoid



tanh



ReLU

equations

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

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
$$= 2\sigma(2x) - 1$$

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

ranges

$0 < \sigma(x) < 1$

$-1 < \tanh(x) < 1$

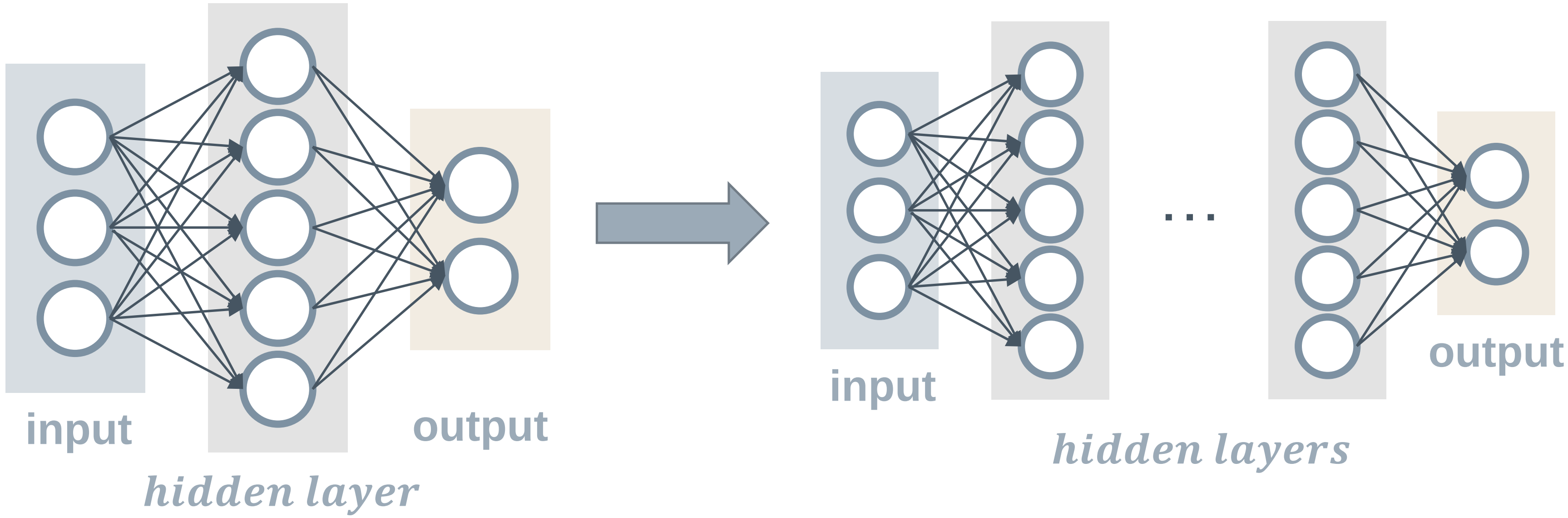
$0 < ReLU(x) < 1$

How to Let NNs Work Better

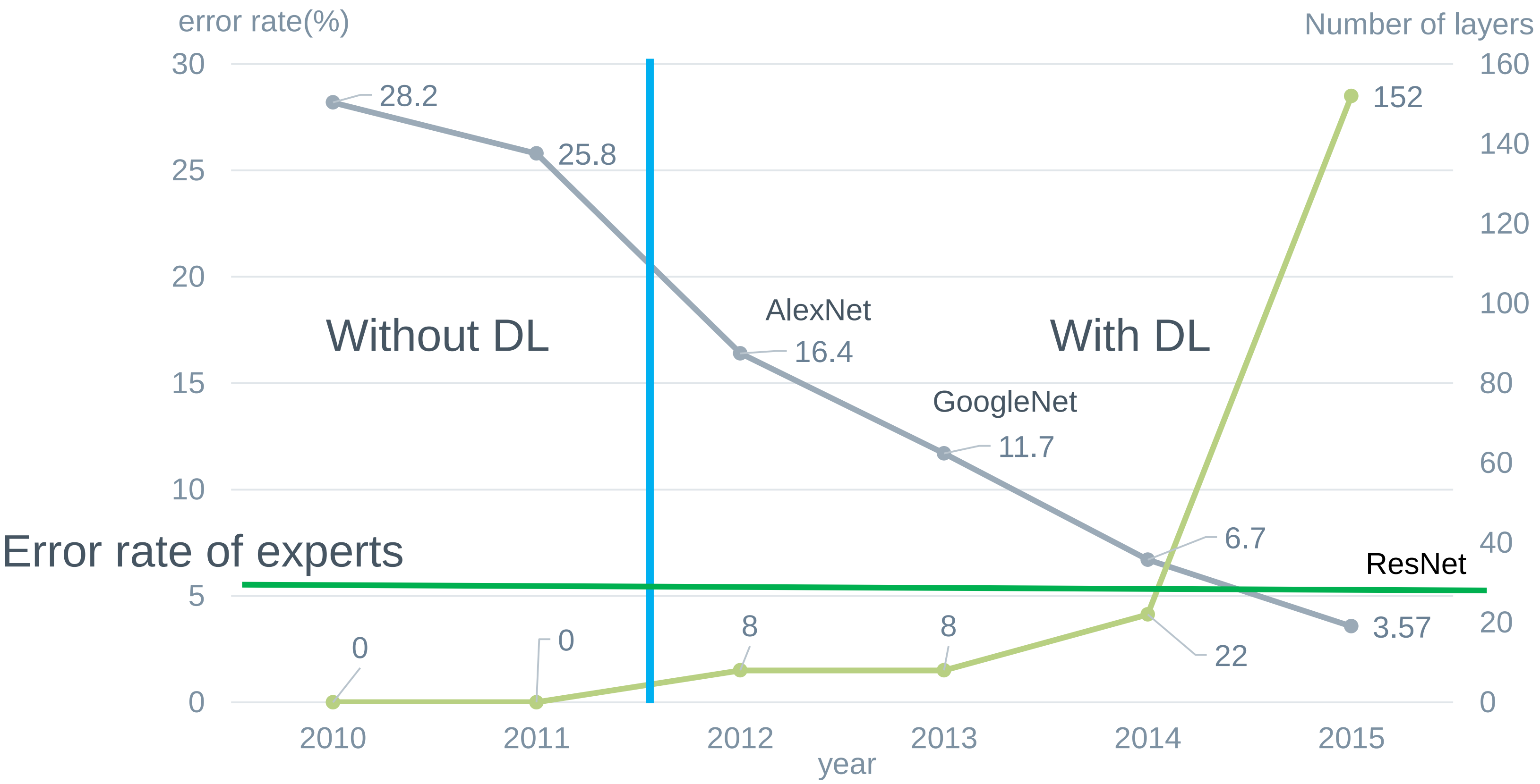
- Something you need to consider first
  - What activation function should I use
  - How many neurons do I need
  - How many layers should I need
- More advanced
  - What kind of network structure should I use (CNN, RNN, GAN, etc.)

Deep Neural Network

- Deep: more hidden layers



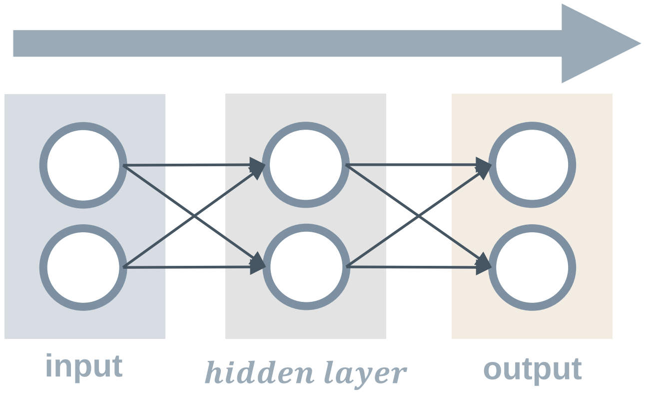
The Deeper the Better?



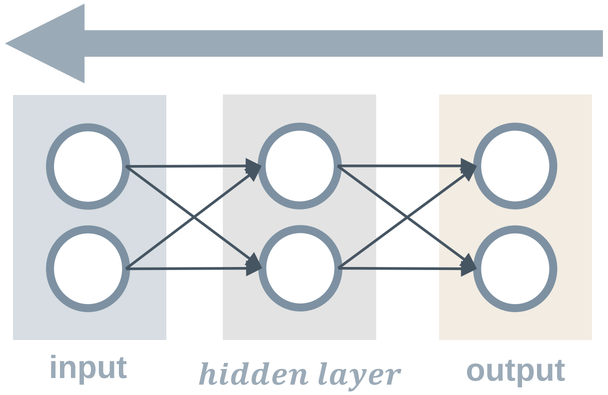
Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei.
ImageNet Large Scale Visual Recognition Challenge. *IJCV,* 2015.

How to Train the NNs

- Input some examples
- Calculate the output
  - Forward propagation

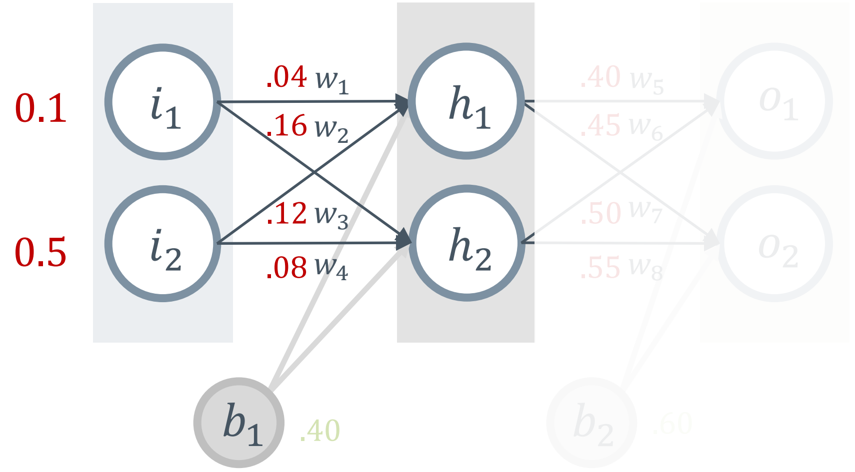


- Measure the errors between the outputs and answers
- Update the weights in NN
  - Back propagation



Forward Propagation

$$\sum_i w_i x_i + b$$



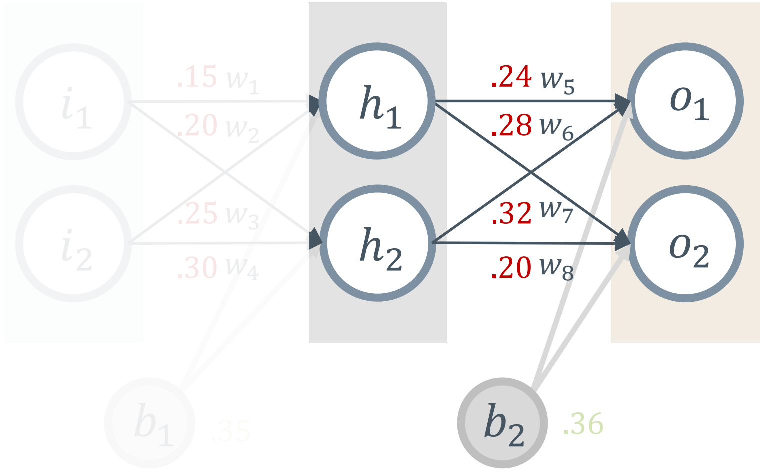
$$\begin{aligned} net_{h_1} &= w_1 * i_1 + w_2 * i_2 + b_1 * 1 \\ &= 0.04 * 0.1 + 0.12 * 0.5 + 0.40 * 1 \\ &= 0.464 \end{aligned}$$

$$\begin{aligned} out_{h_1} &= \frac{1}{1+e^{-net_{h_1}}} = \frac{1}{1+e^{-0.464}} \\ &= 0.613962657 \end{aligned}$$

$$out_{h_2} = 0.611114647$$

Forward Propagation

$$\sum_i w_i x_i + b$$



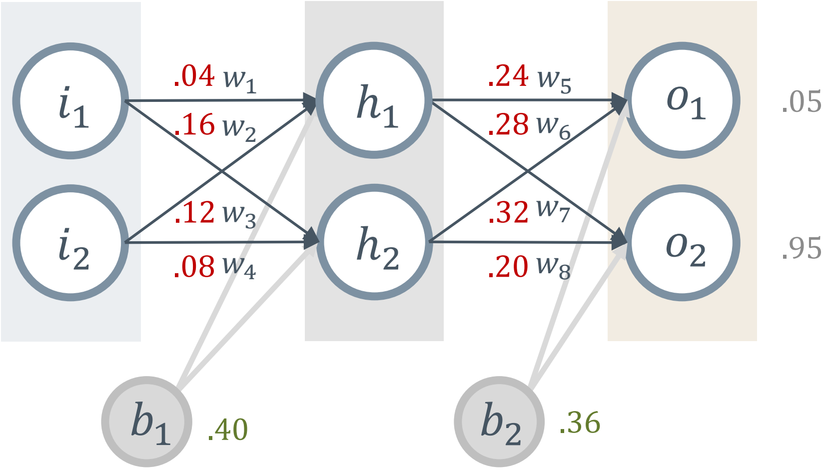
$$net_{o_1} = w_5 * out_{h_1} + w_6 * out_{h_2} + b_2 * 1$$
$$= 0.24 * 0.613962657 + 0.32 * 0.611114647 + 0.36 * 1$$
$$= 0.702907725$$

$$out_{o_1} = \frac{1}{1+e^{-net_{o_1}}} = \frac{1}{1+e^{-0.702907725}}$$
$$= 0.668832137$$

$$out_{o_2} = 0.657941101$$

The Errors of Outputs

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$



$$\begin{aligned} E_{o_1} &= \frac{1}{2}(target - output)^2 \\ &= \frac{1}{2}(0.05 - 0.668832137)^2 \\ &= 0.191476607 \end{aligned}$$

The total error for the neural network is the sum of these errors:

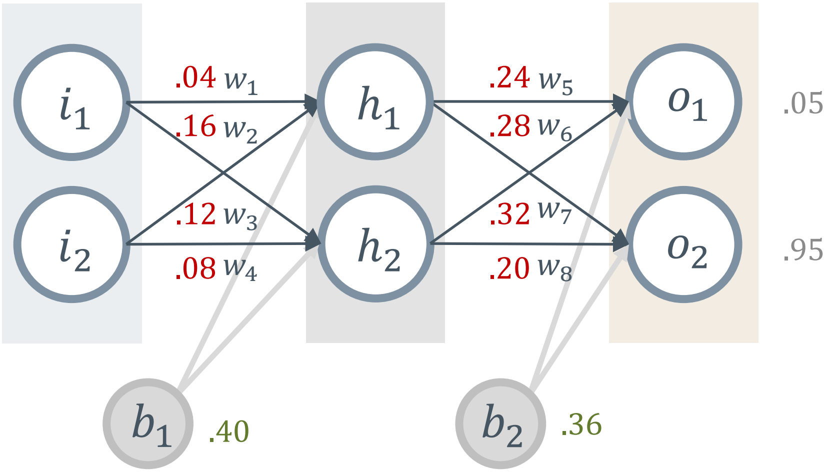
$$E_{total} = E_{o_1} + E_{o_2} = 0.191476607 + 0.042649200 = 0.234125807$$

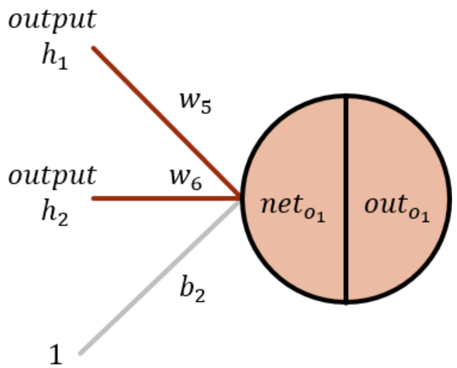
Updating Weights

- The only layer with the answer is the output layer
  - The only layer we can know the errors
- We need to update the weights from the output layer to hidden layers
- Solution: Back-propagation

Backward Propagation to Update w5

- Output layer





loss activate weight

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_5}$$

$$E_{total} = \frac{1}{2}(target - out_{o_1})^2 + \frac{1}{2}(target - out_{o_2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o_1}} = 2 * \frac{1}{2}\left(target_{o_1} - out_{o_1}\right)^{2-1} * (-1) + 0$$

$$= -\left(target_{o_1} - out_{o_1}\right)$$

$$= -(0.05 - 0.668832137)$$

$$= 0.618832137$$

- **Chain rule:**
  - $y = f(x), z = g(y),\ \ z = g(f(x)) = (g \circ f)(x)$

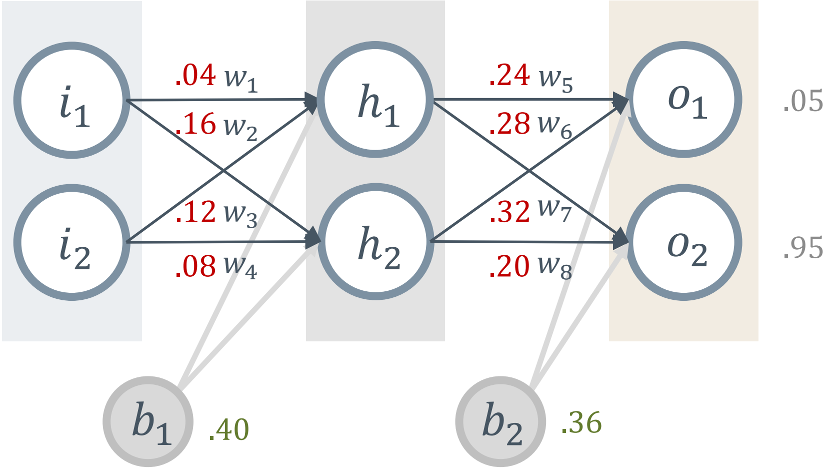
$$(g \circ f)'(x) = \frac{dg}{dx} = \frac{dg}{df}\frac{df}{dx}$$

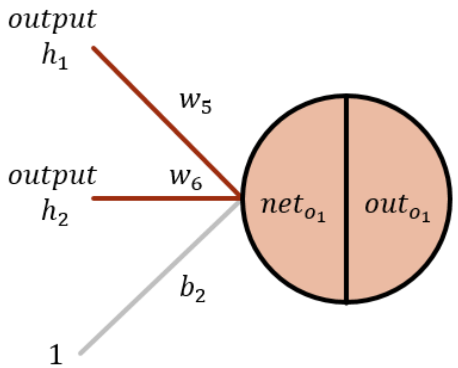
  - $z = f(x, y)$, where $x = g(t), y = h(t)$

$$\frac{df}{dt} = \frac{\partial f}{\partial g}\frac{dg}{dt} + \frac{\partial f}{\partial h}\frac{dh}{dt}$$

Backward Propagation to Update w5

- Output layer





$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_5}$$

$$out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}}$$

$$\frac{\partial out_{o_1}}{\partial net_{o_1}} = out_{o_1}(1 - out_{o_1})$$

$$= 0.668832137(1 - 0.668832137)$$

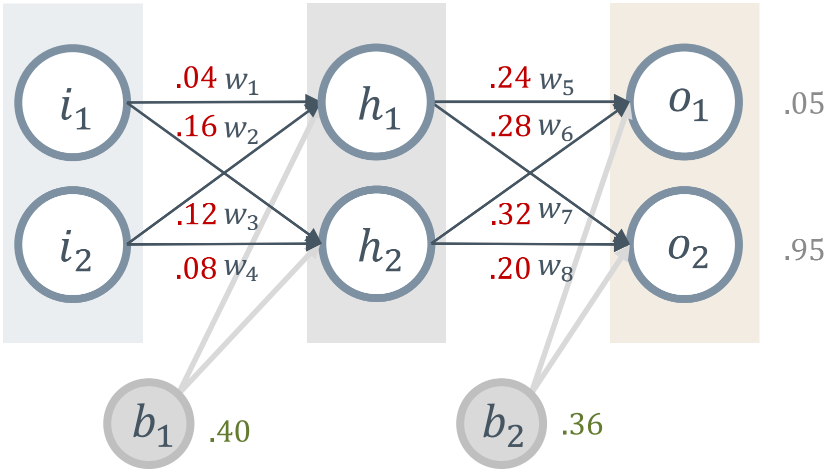
$$= 0.221495709$$

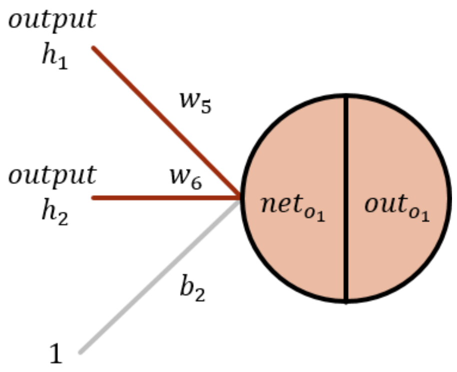
- **Derivative of sigmoid function:**

$$\sigma(x)' = \frac{d}{dx}(1+e^{-x})^{-1}$$
$$= -(1+e^{-x})^{-2}(-e^{-x})$$
$$= \frac{e^{-x}}{(1+e^{-x})^2}$$
$$= \frac{1}{1+e^{-x}} \frac{e^{-x}}{1+e^{-x}}$$
$$= \sigma(x)(1-\sigma(x))$$

Backward Propagation to Update w5

- Output layer





$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_5}$$

$$net_{o_1} = w_5 * out_{h_1} + w_6 * out_{h_2} + b_2 * 1$$

$$\frac{\partial net_{o_1}}{\partial w_5} = 1 * out_{h_1} * w_5^{1-1} + 0 + 0$$

$$= out_{h_1}$$

$$= 0.613962657$$

Backward Propagation to Update w5

- Output layer

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.618832137 * 0.221495709 * 0.613962657 = 0.08415504$$

$$w_5^+ = w_5 - \boxed{\eta} * \frac{\partial E_{total}}{\partial w_5}$$

Learning rate

$$= 0.24 - 0.5 * 0.08415504$$

$$= 0.19792248$$

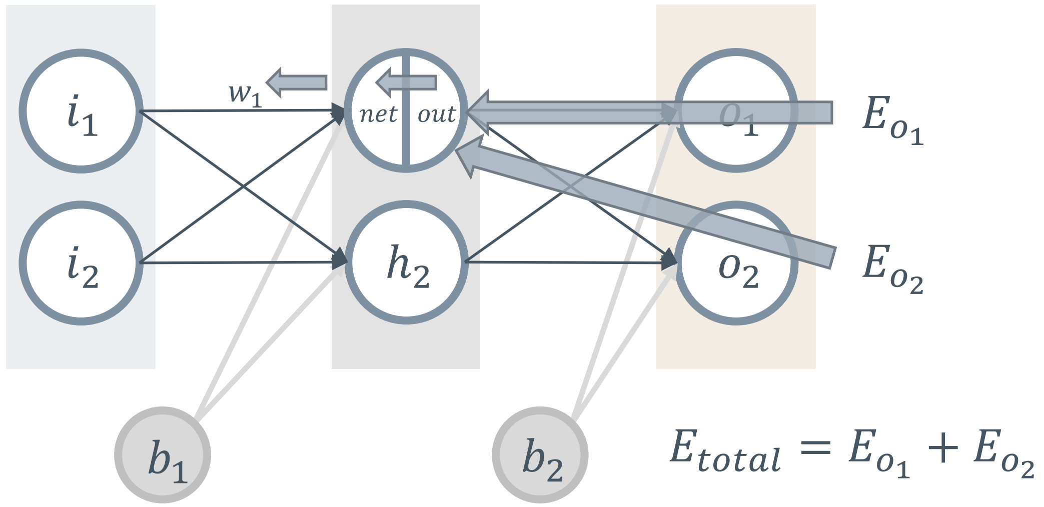
$$w_6^+ = 0.238117666$$

$$w_7^+ = 0.300177638$$

$$w_8^+ = 0.300084039$$

Backward Propagation to Update w1

- Hidden layer



$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_1}$$

Backward Propagation to Update w1

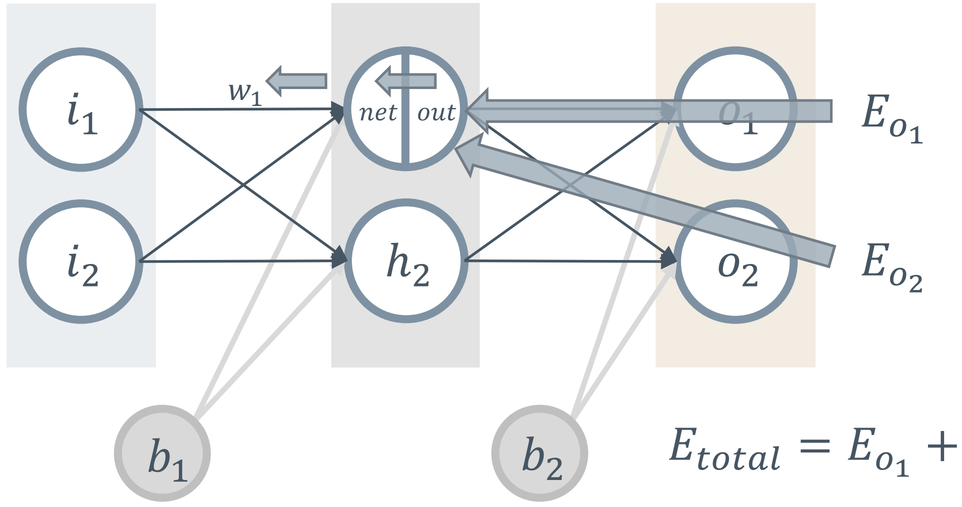
- Hidden layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial net_{h_1}}$$

$$\frac{\partial E_{o_1}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial out_{h_1}}$$

$$\frac{\partial E_{o_1}}{\partial net_{o_1}} = \frac{\partial E_{o_1}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}}$$



$$E_{total} = E_{o_1} + E_{o_2}$$

$$\frac{\partial E_{o_1}}{\partial net_{o_1}} = \frac{\partial E_{o_1}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} = 0.618832137 * 0.337664274 = 0.208957504$$

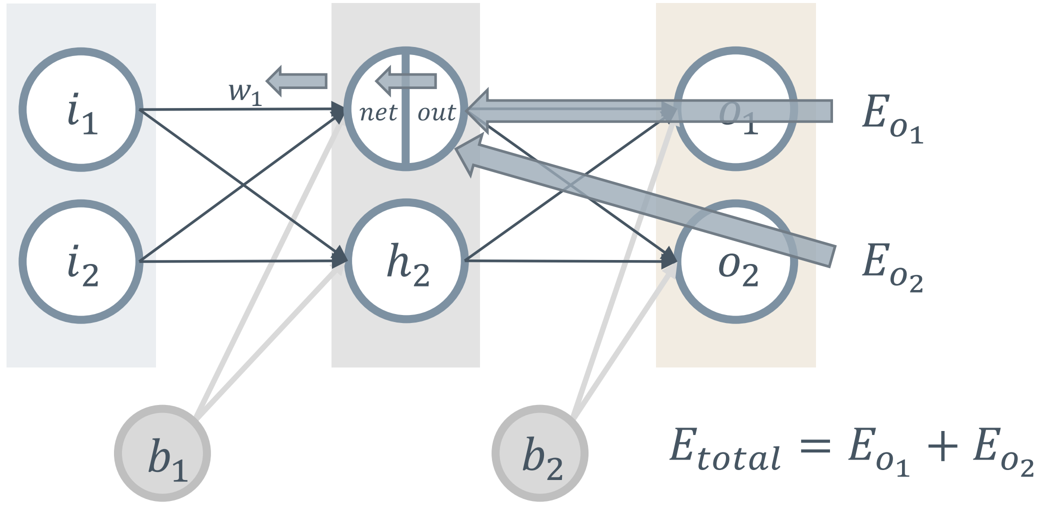
Backward Propagation to Update w1

- Hidden layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial net_{h_1}}$$

$$\frac{\partial E_{o_1}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial out_{h_1}}$$



$$net_{o_1} = w_5 * out_{h_1} + w_6 * out_{h_2} + b_2 * 1$$

$$\frac{\partial net_{o_1}}{\partial out_{h_1}} = w_5 = 0.24$$

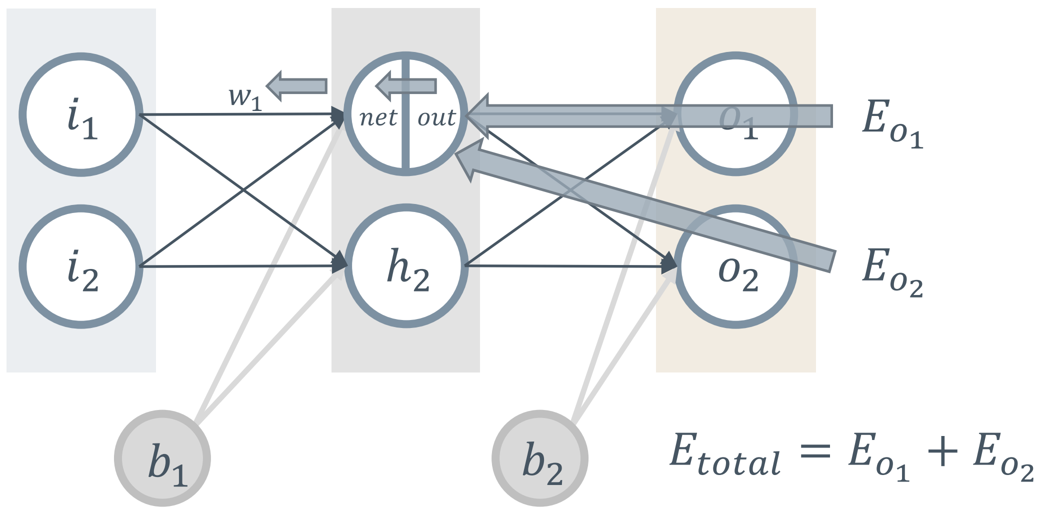
$$\frac{\partial E_{o_1}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial out_{h_1}} = 0.208957504 * 0.24 = 0.050149801$$

Backward Propagation to Update w1

- Hidden layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial net_{h_1}}$$



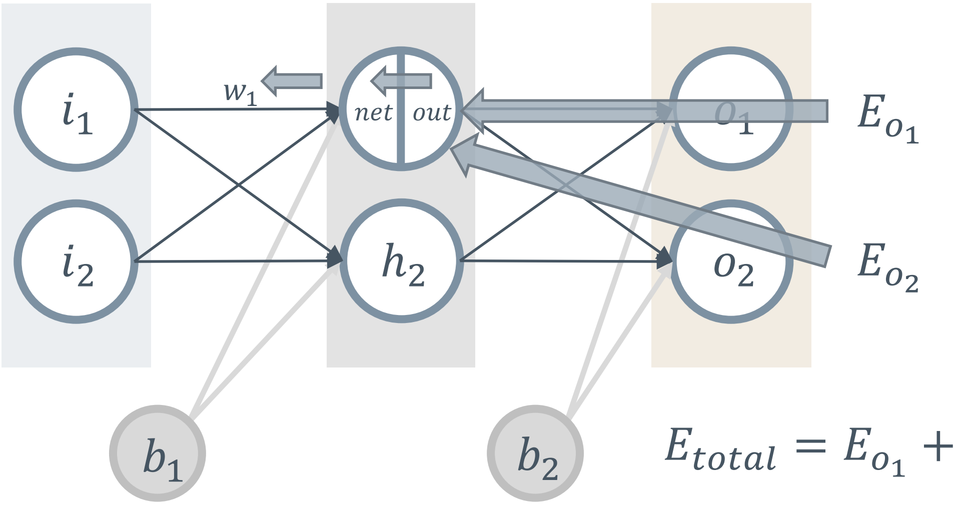
$$\frac{\partial E_{o_1}}{\partial out_{h_1}} = 0.050149801, \qquad \frac{\partial E_{o_2}}{\partial out_{h_1}} = -0.018404176$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial net_{h_1}} = 0.050149801 + (-0.018404176\ ) = 0.031745625$$

Backward Propagation to Update w1

- Hidden layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \boxed{\frac{\partial out_{h_1}}{\partial net_{h_1}}} * \frac{\partial net_{h_1}}{\partial w_1}$$



$$E_{total} = E_{o_1} + E_{o_2}$$

$$out_{h_1} = \frac{1}{1 + e^{-net_{h_1}}}$$

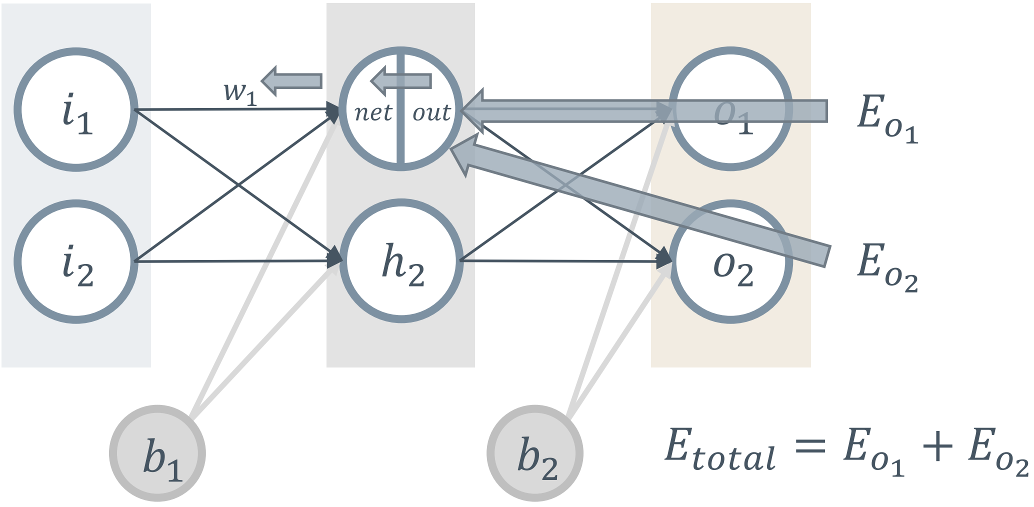
$$\frac{\partial out_{h_1}}{\partial net_{h_1}} = out_{h_1}(1 - out_{h_1}) = 0.613962657 * (1 - 0.613962657\ )$$

$$= 0.237012513$$

Backward Propagation to Update w1

- Hidden layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_1}$$



$$net_{h_1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h_1}}{\partial w_1} = i_1 = 0.1$$

Backward Propagation to Update w1

- Hidden layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.031745625 * 0.237012513 * 0.1 = 0.000752411$$

$$\begin{aligned} w_1^+ &= w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} \\ &= 0.04 - 0.5 * 0.000752411 \\ &= 0.039623795 \end{aligned}$$

$$w_2^+ = 0.118118973$$

$$w_3^+ = 0.159635010$$

$$w_4^+ = 0.078175051$$

Gradient Vanishing

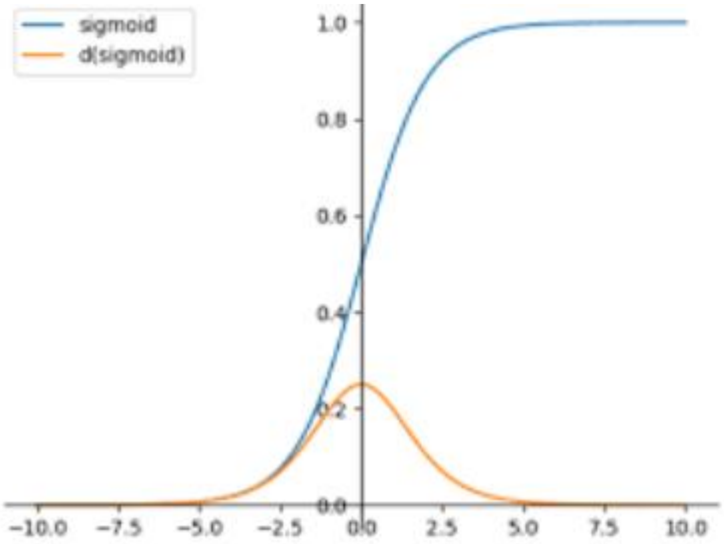
› Update weight

$$w_5^+ = w_5 - \eta * \boxed{\frac{\partial E_{total}}{\partial w_5}}$$

Gradient

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} * \boxed{\frac{\partial out_{o_1}}{\partial net_{o_1}}} * \frac{\partial net_{o_1}}{\partial w_5}$$

activation



› Gradient Vanishing

– Derivative of sigmoid is lower than 0.25

Gradient Vanishing

- Use Relu as Activation function (after Derivation only 0 or 1)
- Small Layers
- Redesign the network structure
  - ResNet or others
- Do not use deep learning

How to Design A Good Neural Network Model

The Hyper Parameters

- Dimensions
  - The number of neurons in each layer
  - Can be different in each layer
- Numbers of layers
  - How depth is your model
- Activation function
  - Linear: ReLu
  - Non-linear: Sigmoid, tanh
- The bias in each layer

How Many Random Variables in Neural Networks

- Consider a neural network
  - 10 layers
  - 100 nodes in each layer
  - 1 bias in each layer
- 1 layer has 10100 parameters
  - 100*100+100
- 10 layers has 101000 parameters
  - 10100*10

The More Random Variables The Better?

- More random variables can represent more latent information
- Too many random variables will lead overfitting
  - Too fit to some special cases

How to Prevent Overfitting

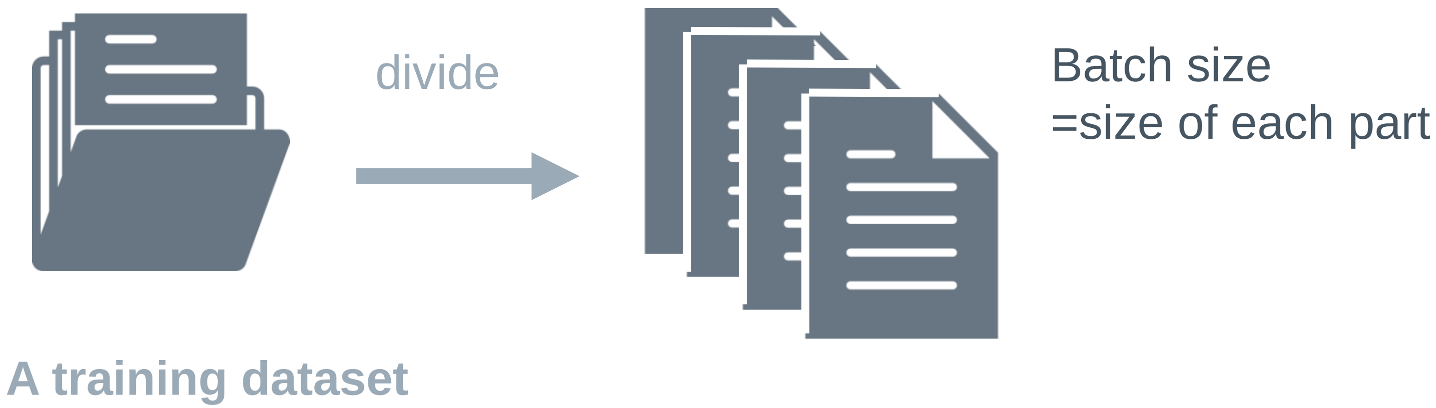
- Decrease your random variables
  - Decrease your dimensions or layers
  - May incur some errors
- Increase your training data
  - Very difficult in practice
- Dropout some variables
  - Let some variables not be trained in the training phase
  - Still in use in the testing phase

How Much Training Data We Need

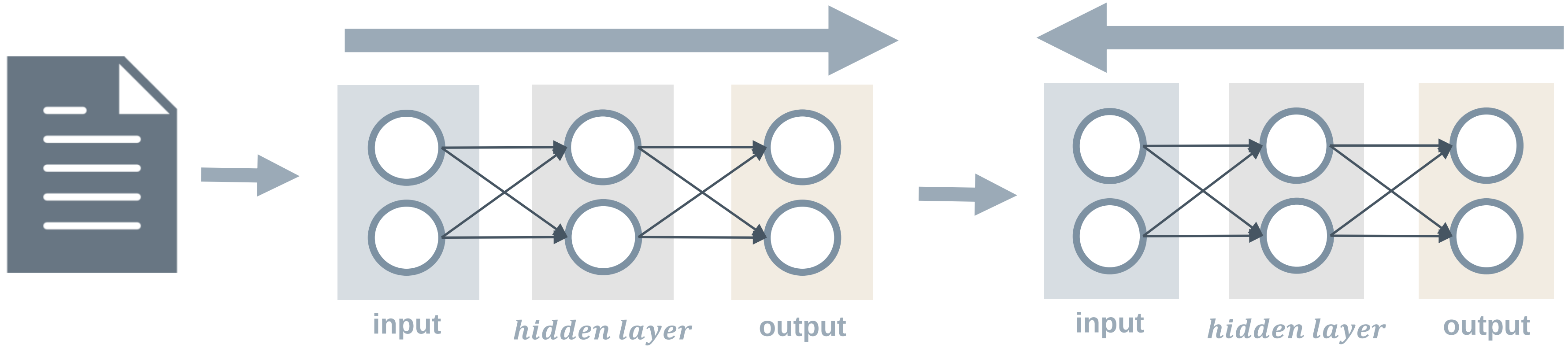
- 10~30 times data to train random variables
  - we need 1010000 ~ 3030000 data to train 101000 variables
- Few data may not be able to train a good model
  - Some variables may not be trained well

More Hyper Parameters

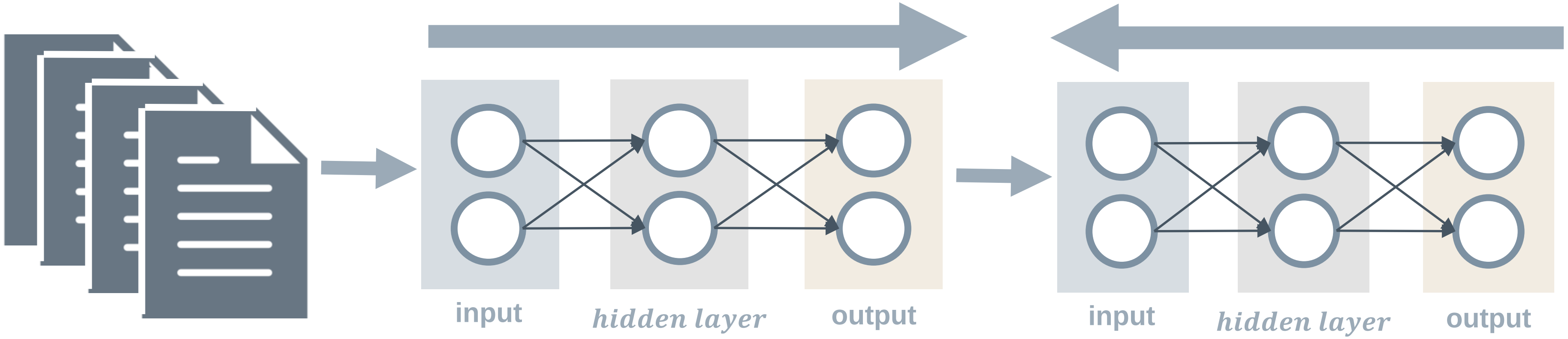
› Learning rate
  – How many updates of the weight via gradients
› Batch size
  – How many input data in each iteration
› Epoch [epək]
  – How many times of passing the training data in the model



The Difference Between Iteration and Epoch



One iteration



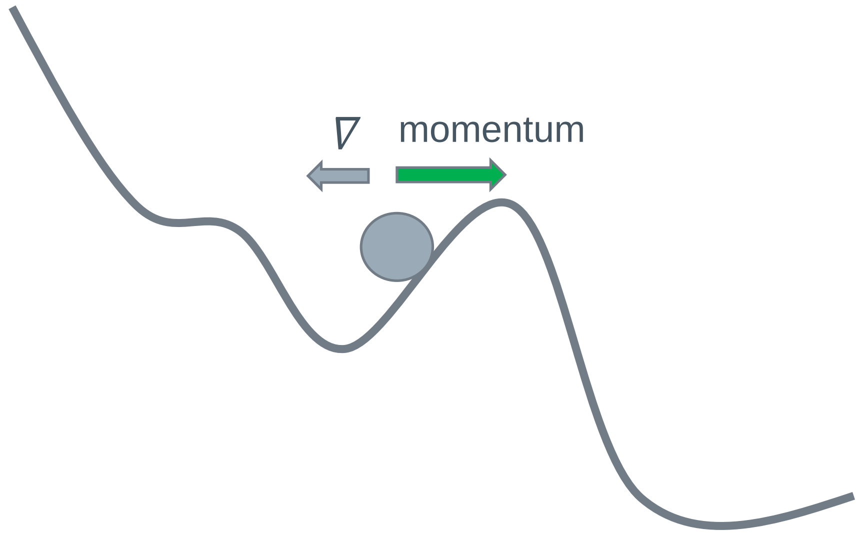
One epoch

Hyper Parameters

- Data set size = Iteration * Batch size （1 Epoch）
- Iteration = （Data set size / Batch size） * Epoch
  - Batch size is large, it considers more data, resulting in more accurate corrections. However, each iteration takes longer, leading to fewer advancements towards the minima.
  - A smaller batch size, it considers only local data, causing directional biases in corrections. Yet, because each iteration involves less data, there is a chance to make more corrections within the same time frame.

Different Way to Optimize Neural Networks

- Stochastic Gradient Descent (SGD)
  - Update the weights at each input example instead of update the weight after each epoch
- Add momentums on gradient descent
  - $v_{t+1} = \lambda v_t - (1-\lambda) * \frac{\partial E_{total}}{\partial w_1}$
  - $w_1^+ = w_1 + \eta * v_{t+1}$



Another Research Area on Neural Networks

- Transfer Learning
  - Share the Neural Networks in similar domain
    - Weight transferring
    - Teacher Networks
- Neural Network reasoning
  - Explain the reasons of each parameter

The Advanced Neural Networks

- Convolutional Neural Networks (CNNs)
  - Image processing or image recognition
- Recurrent Neural Networks (RNNs)
  - Time series prediction or language models
  - Long Short Term Memories (LSTMs)
    - The advanced RNNs to record long term data
- AutoEncoders
  - Objects representation or object embedding
- Generative Adversarial Networks (GANs)
  - Creativity tasks