CS 540 Introduction to Artificial Intelligence
Neural Networks Review

University of Wisconsin-Madison
Spring 2026 Sections 1 & 2

Midterm Information

- **Time: March 24th 5:45-7:15 PM**
- **Location (by section **):**
  - Section 001 (Tuesday/Thursday 11-12:15PM): 6210 Social Sciences Bldg
  - Section002 (Tuesday/Thursday 2:30-3:45PM): B10 Ingraham Hall
  - Students with McBurney accommodations should have received an email with additional information.
  - Students who cannot take the exam on the specified time should contact their instructor if they have not done it yet.
- **Topics: Topics covered up to and including Week 9**
- **Exclusion List (questions regarding the following topics will NOT appear on the midterm):**
  - **Logic (covered in sections 1 and 2)**
  - **SVM + Kernel Trick (covered in section 3)**
- **Format**: MCQ
- **Cheat sheet**: a handwritten single piece of paper, front and back
- **Calculator**: optional, if it doesn't have an Internet connection
- **Bring**: your WISC ID, pencil (No 2 or softer), your 1-sheet notes.
- **Past exam questions**: on Canvas →Files → Past Exams

Neural Networks

How to classify

Cats vs. dogs?

Neural networks can also be used for regression.

- Typically, no activation on outputs, mean squared error loss function.

Single-layer Perceptron

↓

Multi-layer Perceptron

↓

Training of neural networks

↓

Convolutional neural networks

Review: Perceptron

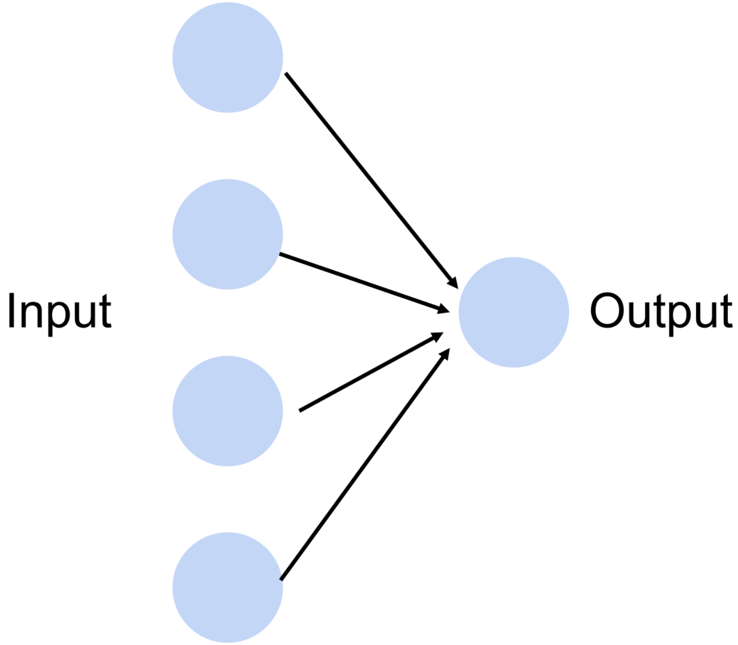
- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b , perceptron outputs:

$$o = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Activation function

Cats vs. dogs?

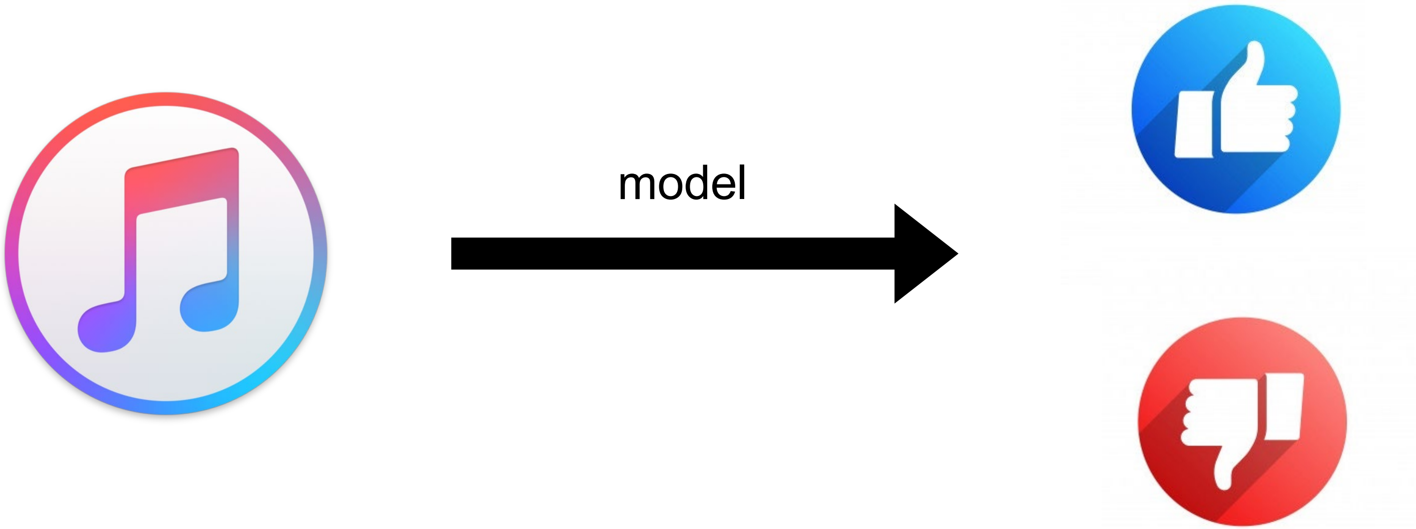


The Perceptron Learning Rule

Perceptron Learning Algorithm
<u>Input:</u> dataset (X, y) number of steps T, step size η
1. Initialize w_0, b_0 2. For $t = 1, 2, \dots, T$ 3. Pick random (x_i, y_i) 4. Predict $\hat{y}_i \leftarrow \sigma(\langle w_{t-1}, x_i \rangle + b_{t-1})$ 5. If $\hat{y}_i \neq y_\mathrm{i}$: 6. $w_t \leftarrow w_{t-1} + \eta(y_i - \hat{y}_i)x_i$ 7. $b_t \leftarrow b_{t-1} + \eta(y_i - \hat{y}_i)$ 8. Return w_T

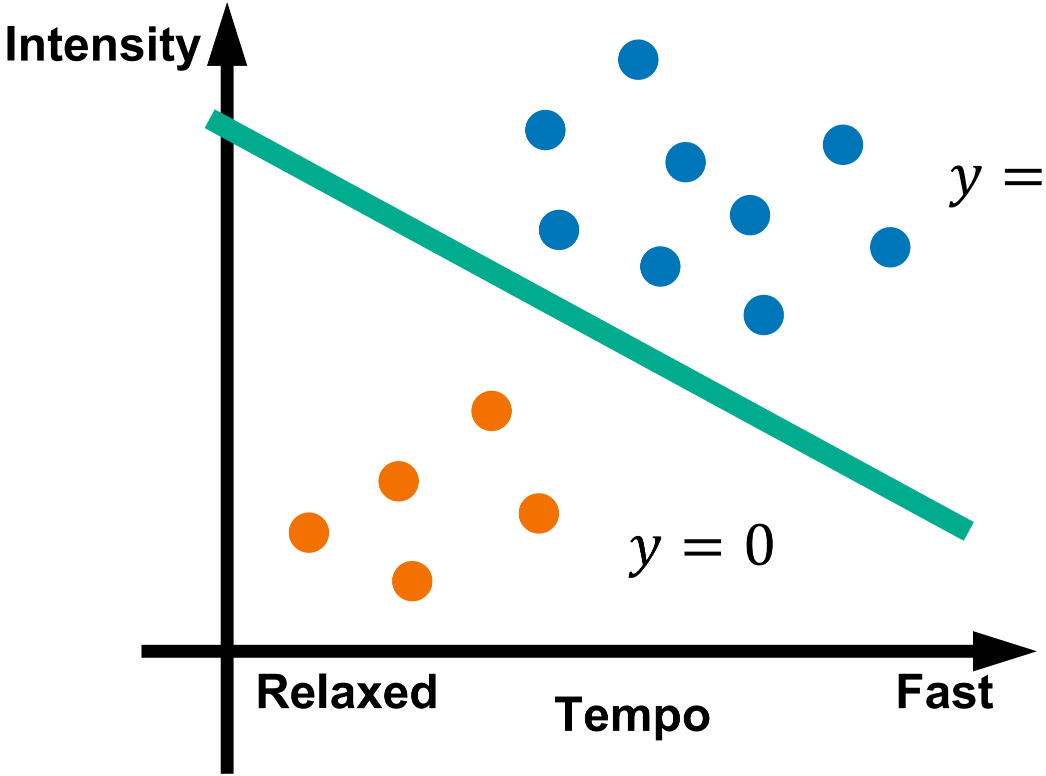
Gradient Descent
<u>Input:</u> dataset (X, y), loss function L, number of steps T, step size η
1. Initialize w_0 2. For $t = 1, 2, \dots, T$ 3. Calculate $g_t = \nabla L(w_{t-1}; X, y)$ 4. Update $w_t \leftarrow w_{t-1} - \eta g_t$ 5. Return w_T

Example 2: Predict whether a user likes a song or not



Example 2: Predict whether a user likes a song or not using Perceptron

User Sharon



- DisLike
- Like

Learning logic functions using perceptron

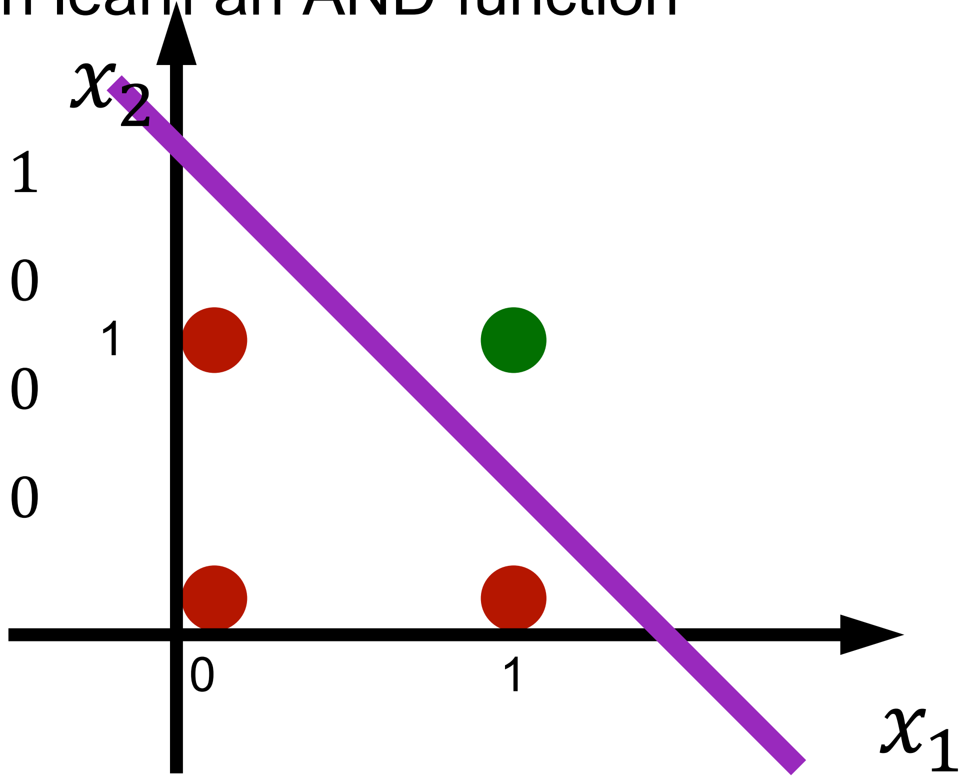
The perceptron can learn an AND function

$x_1 = 1, x_2 = 1, y = 1$

$x_1 = 1, x_2 = 0, y = 0$

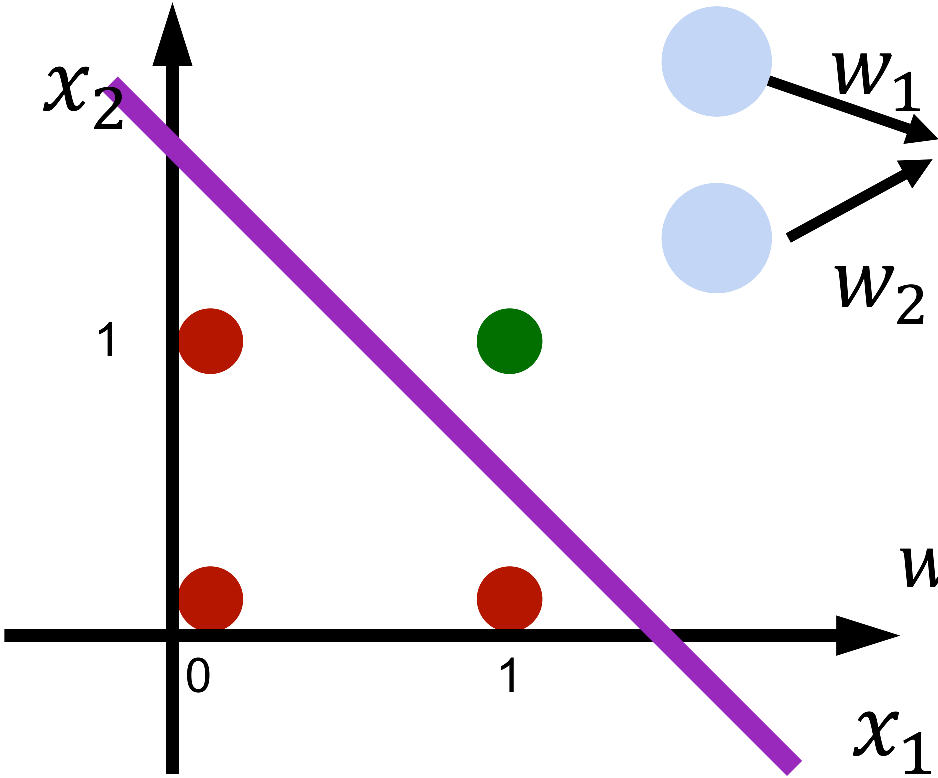
$x_1 = 0, x_2 = 1, y = 0$

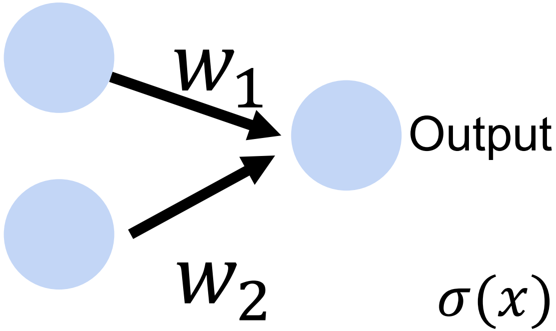
$x_1 = 0, x_2 = 0, y = 0$



Learning logic functions using perceptron

The perceptron can learn an AND function





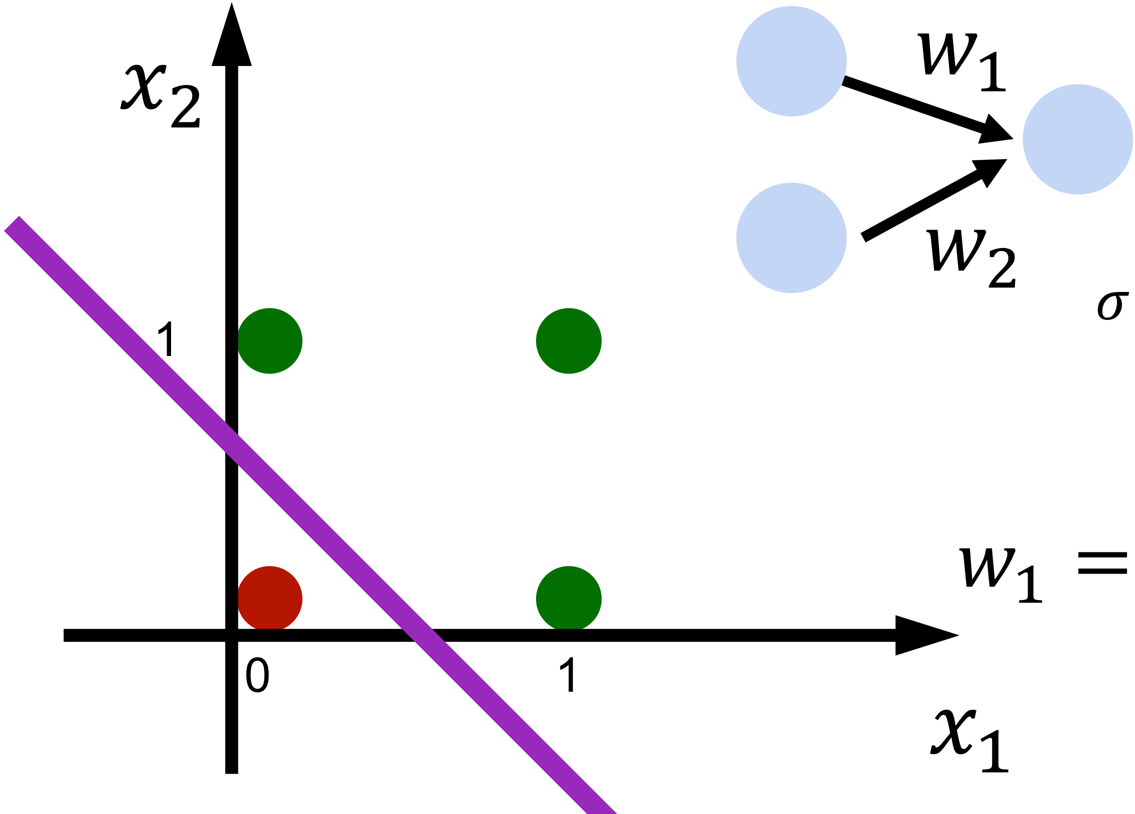
$$\sigma(x_1 w_1 + x_2 w_2 + b)$$

$$\sigma(x) = \begin{cases} 1 & if \;\; x > 0 \\ 0 & otherwise \end{cases}$$

$$w_1 = 1, w_2 = 1, b = -1.5$$

Learning OR function using perceptron

The perceptron can learn an OR function



Output $\sigma(x_1w_1 + x_2w_2 + b)$

$$\sigma(x) = \begin{cases} 1 & if\, x > 0 \\ 0 & otherwise \end{cases}$$

$$w_1 = 1, w_2 = 1, b = -0.5$$

XOR Problem (Minsky & Papert, 1969)

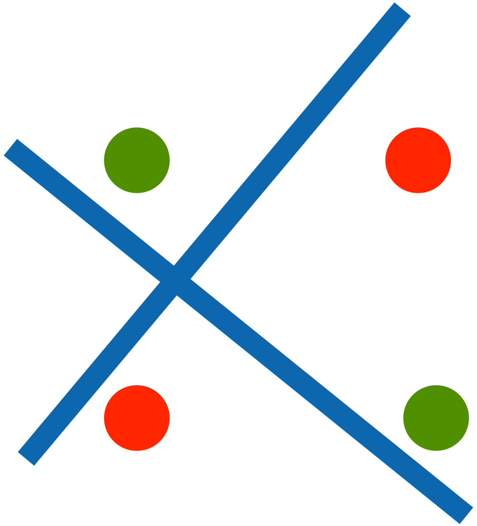
The perceptron cannot learn an XOR function
(neurons can only generate linear separators)

$x_1 = 1, x_2 = 1, y = 0$

$x_1 = 1, x_2 = 0, y = 1$

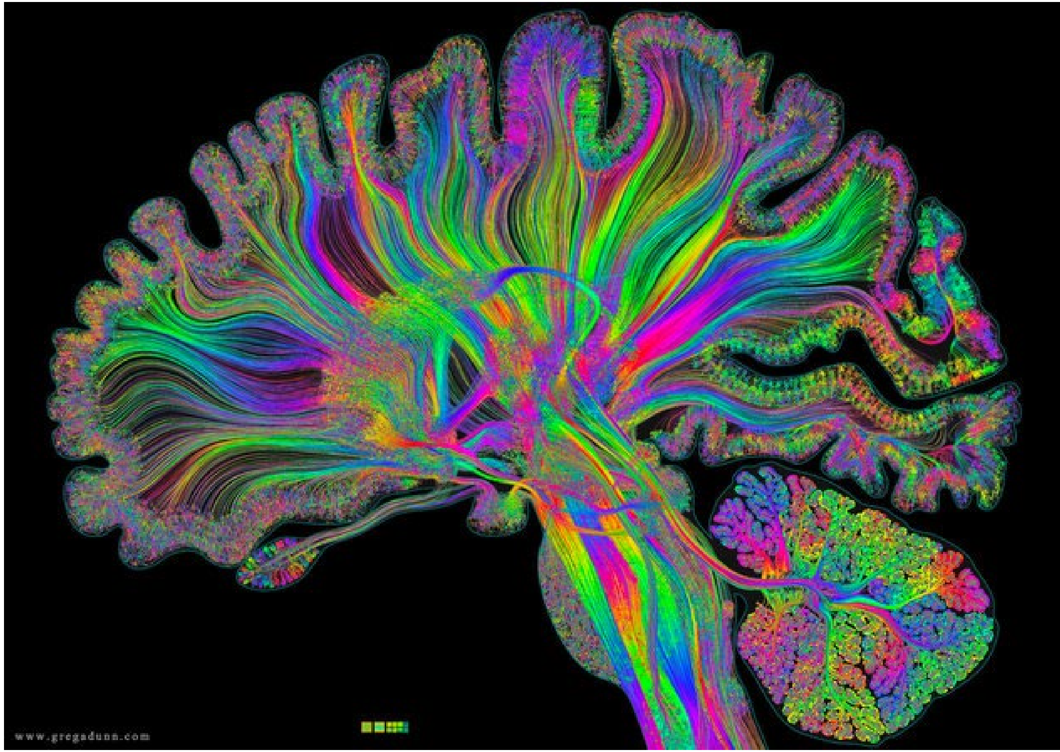
$x_1 = 0, x_2 = 1, y = 1$

$x_1 = 0, x_2 = 0, y = 0$



This contributed to the first AI winter

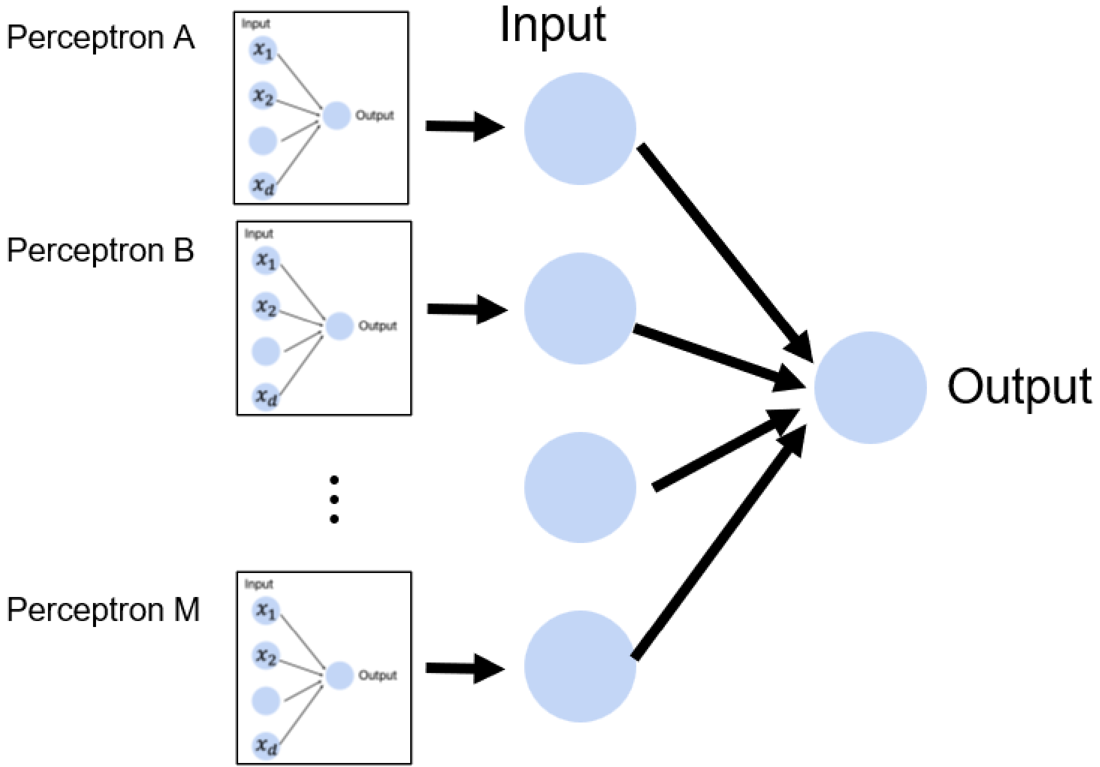
Multilayer Perceptron



Beyond one layer

Big Idea: take our inputs from other perceptrons!

Cats vs. dogs?

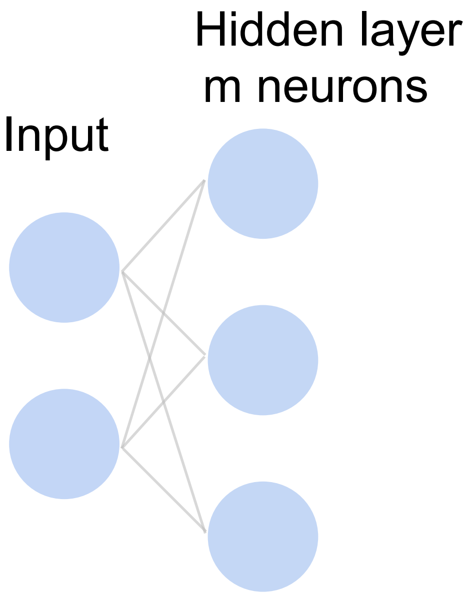


Single Hidden Layer

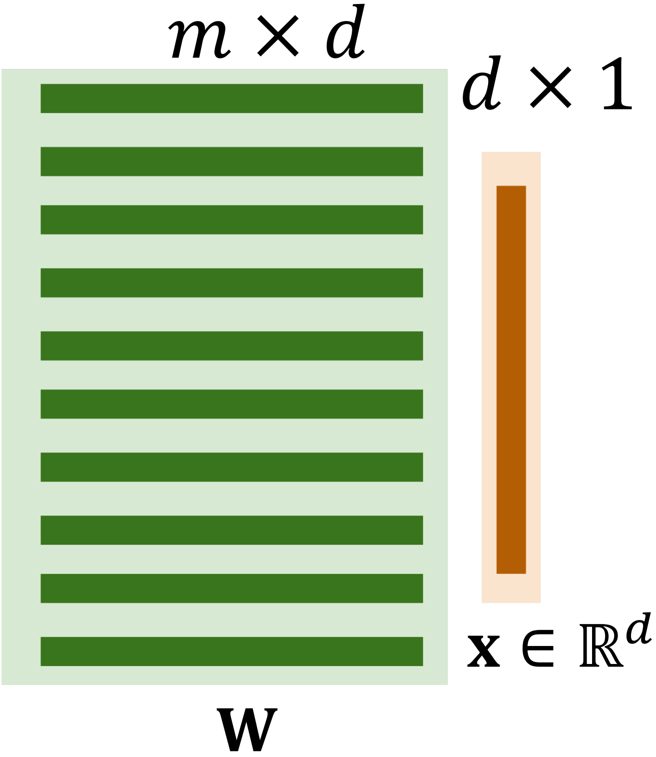
- Input $\mathbf{x} \in \mathbb{R}^d$
- Hidden $\mathbf{W} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m$
- Intermediate output

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

σ is an element-wise activation function

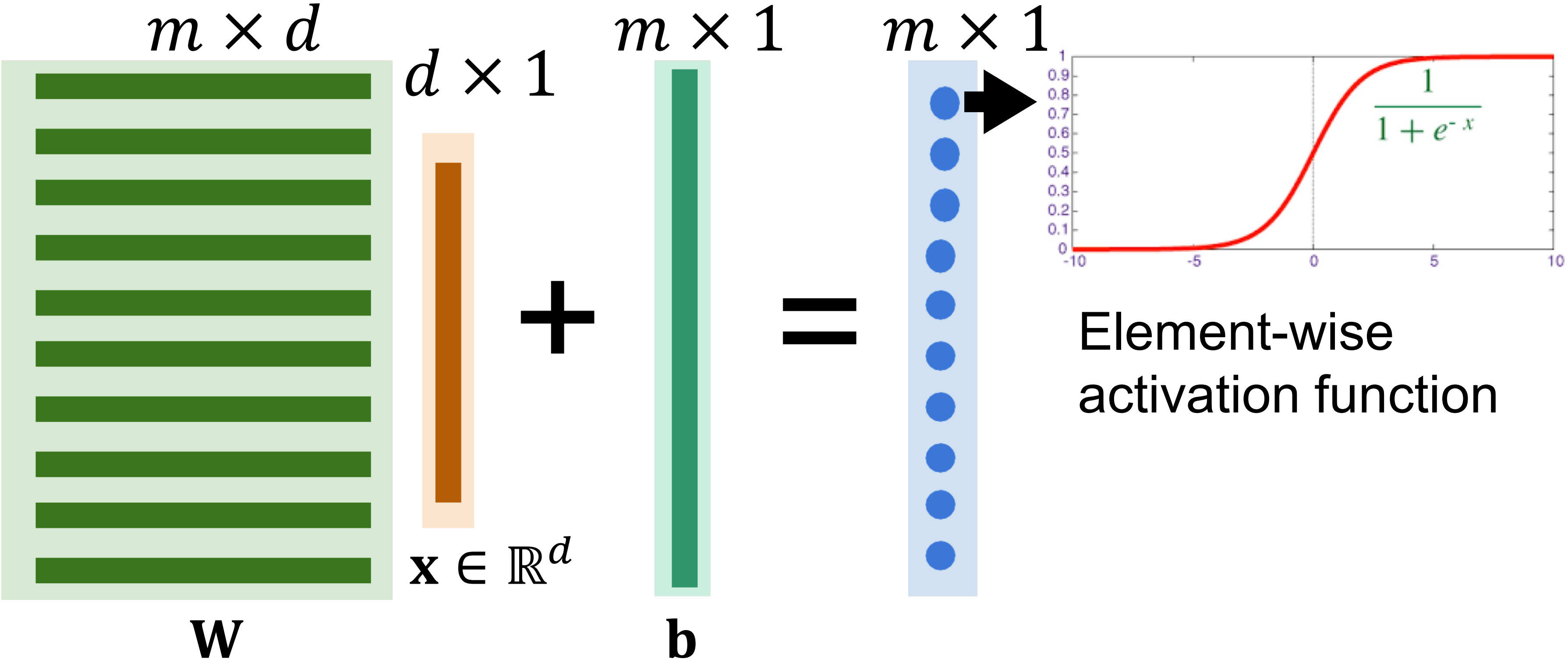


Neural networks with one hidden layer



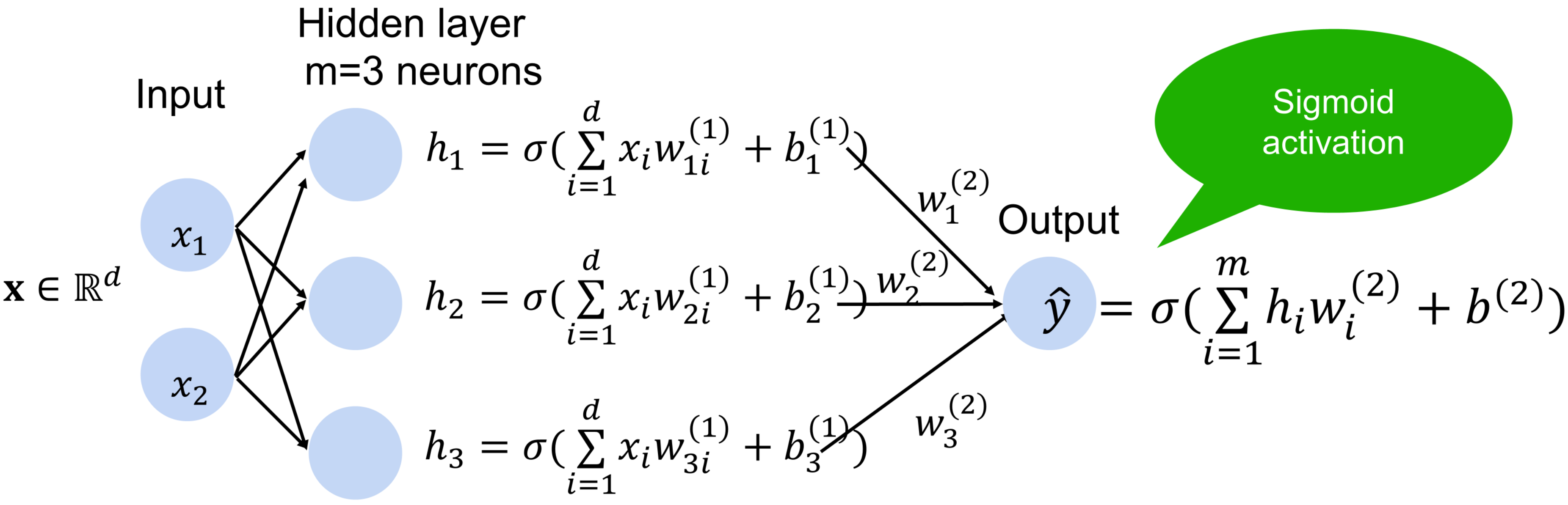
Neural networks with one hidden layer

Key elements: linear operations + Nonlinear activations



Multi-layer perceptron: Example

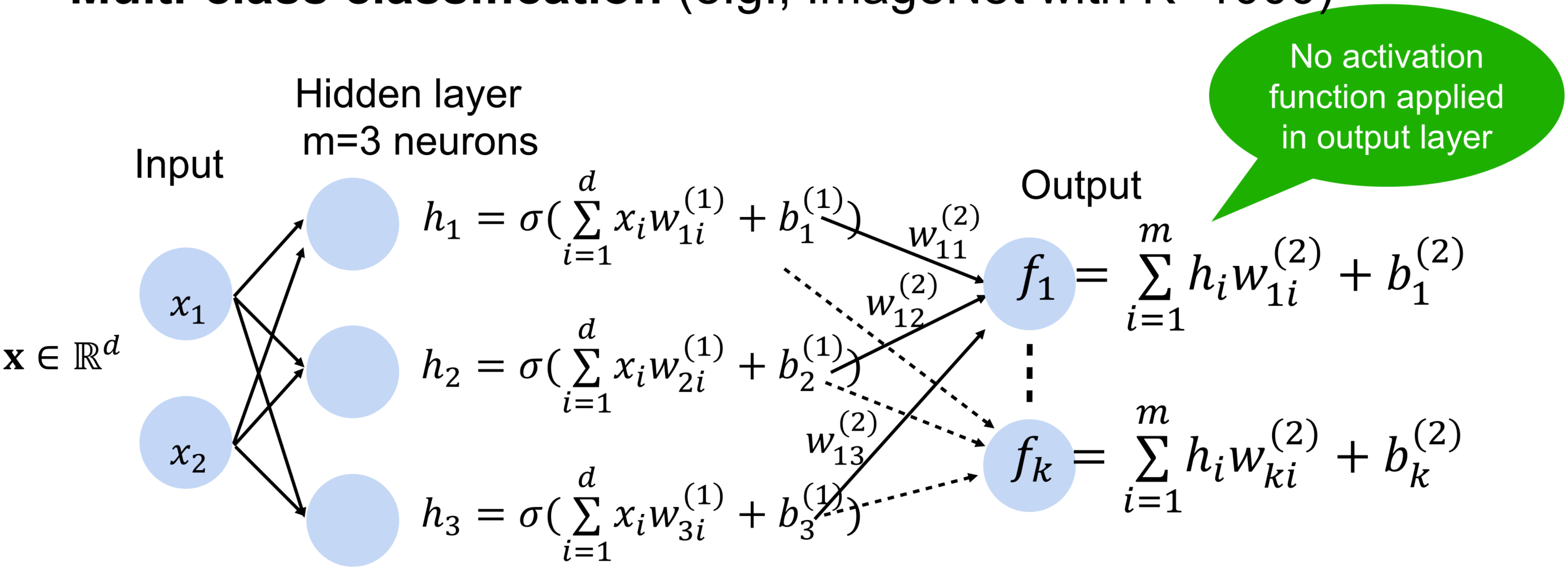
- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer, depth = 2



Neural network for K-way classification

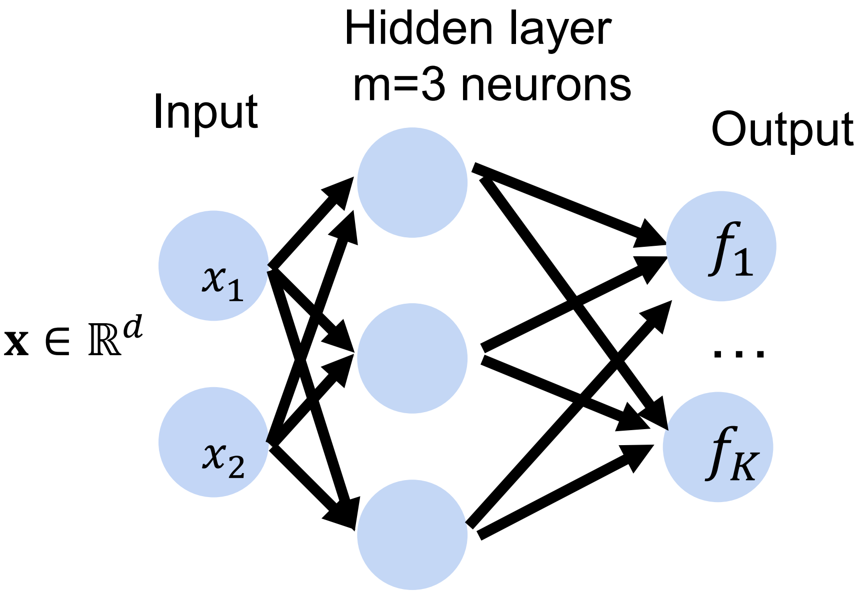
- K outputs in the final layer

Multi-class classification (e.g., ImageNet with K=1000)



Softmax

Turns outputs f into probabilities (sum up to 1 across K classes)



$$p(y|\mathbf{x}) = softmax(f) = \frac{\exp(f_y(x))}{\sum_{k=1}^{K} \exp(f_k(x))}$$

More complicated neural networks: multiple hidden layers

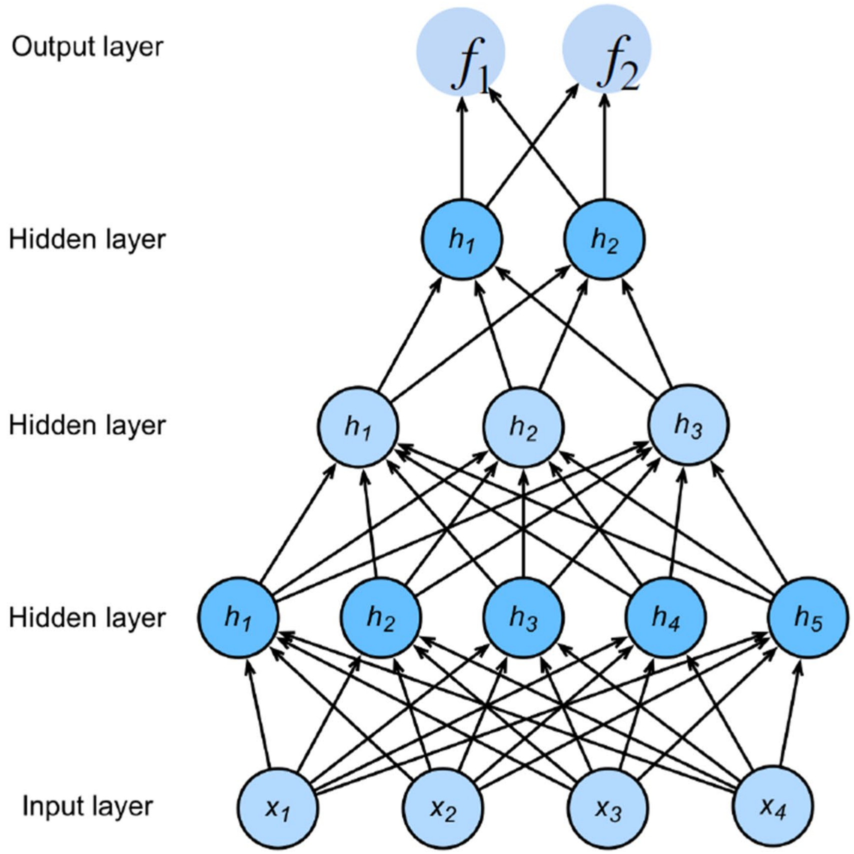
$$\mathbf{h}_1 = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{h}_2 = \sigma(\mathbf{W}^{(2)}\mathbf{h}_1 + \mathbf{b}^{(2)})$$

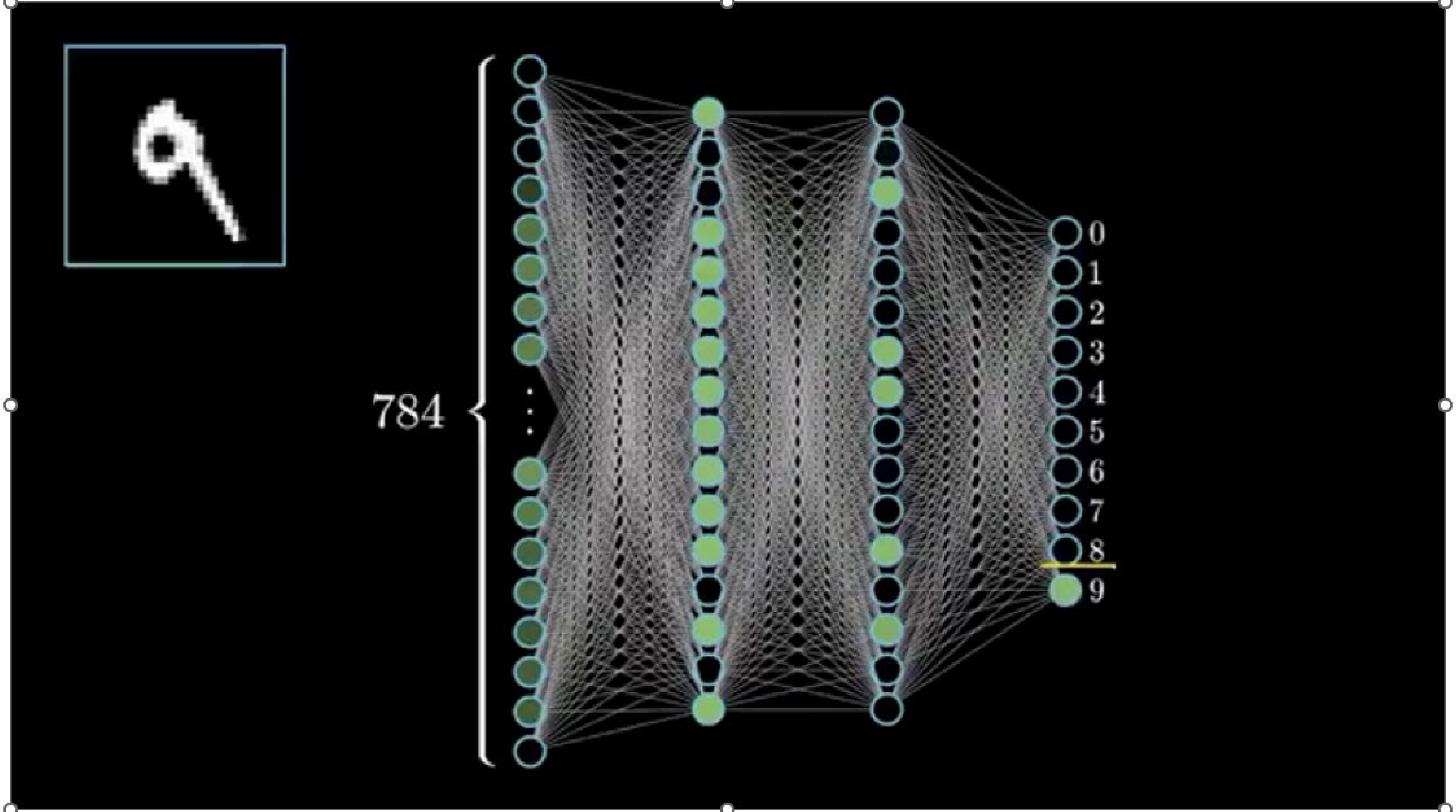
$$\mathbf{h}_3 = \sigma(\mathbf{W}^{(3)}\mathbf{h}_2 + \mathbf{b}^{(3)})$$

$$\mathbf{f} = \mathbf{W}^{(4)}\mathbf{h}_3 + \mathbf{b}^{(4)}$$

$$\mathbf{p} = \text{softmax}(\mathbf{f})$$



Classify MNIST handwritten digits



Quiz Break

Suppose you are given a 3-layer multilayer perceptron (2 hidden layers h1 and h2 and 1 output layer). All activation functions are sigmoids, and the output layer uses a softmax function. Suppose h1 has 1024 units and h2 has 512 units. Given a dataset with 2 input features and 3 unique class labels, how many learnable parameters does the perceptron have in total?

Quiz Break

Suppose you are given a 3-layer multilayer perceptron (2 hidden layers h1 and h2 and 1 output layer). All activation functions are sigmoids, and the output layer uses a softmax function. Suppose h1 has 1024 units and h2 has 512 units. Given a dataset with 2 input features and 3 unique class labels, how many learnable parameters does the perceptron have in total?

$1024 * 2 + 1024 + 512 * 1024 + 512 + 512 * 3 + 3 = 529411$

Break & Quiz

We want to design a neural network for a multi-class classification task. The final layer should be:

- A. ReLU
- B. Sigmoid.
- C. Softmax
- D. Step function.
- E. Tanh.

Break & Quiz

We want to design a neural network for a multi-class classification task. The final layer should be:

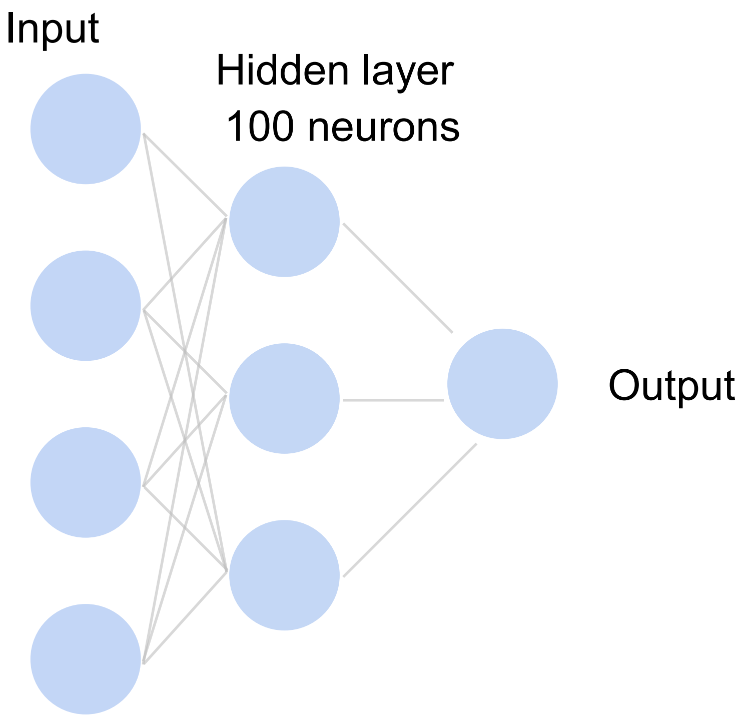
- A. ReLU
- B. Sigmoid
- **C. Softmax**
- D. Step function.
- E. Tanh.

How to train a neural network? Binary classification

Loss function: $\frac{1}{|D|} \sum_{(\mathbf{x},y) \in D} \ell(\mathbf{x}, y)$

Per-sample loss:

$$\ell(\mathbf{x}, y) = -y\log(\hat{y}) - (1 - y)\log(1 - \hat{y})$$



Negative log likelihood
Minimizing NLL is equivalent to Max Likelihood Learning (MLE)
Also known as binary cross-entropy loss

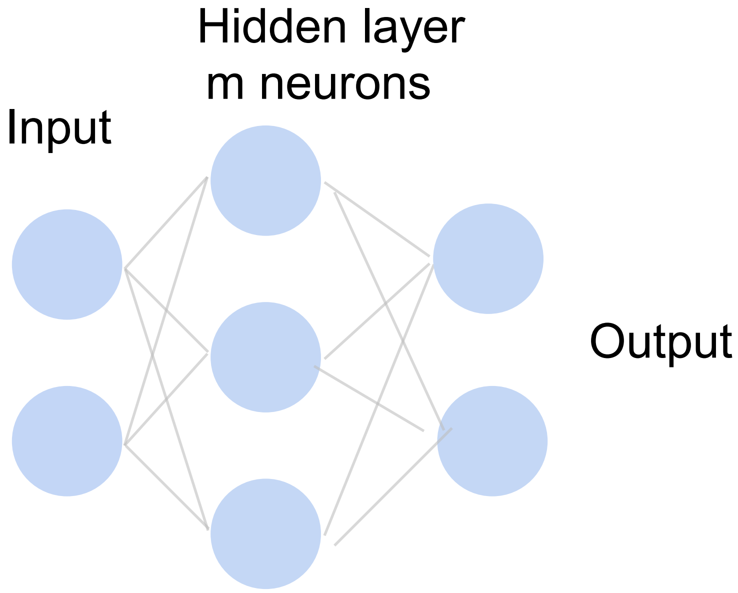
How to train a neural network? Multi-class classification

Loss function: $$\frac{1}{|D|}\sum_i \ell(\mathbf{x}_i, y_i)$$

Per-sample loss:

$$\ell(\mathbf{x}, y) = \sum_{j=1}^{K} -y_j \log p_j$$

Also known as cross-entropy loss or softmax loss

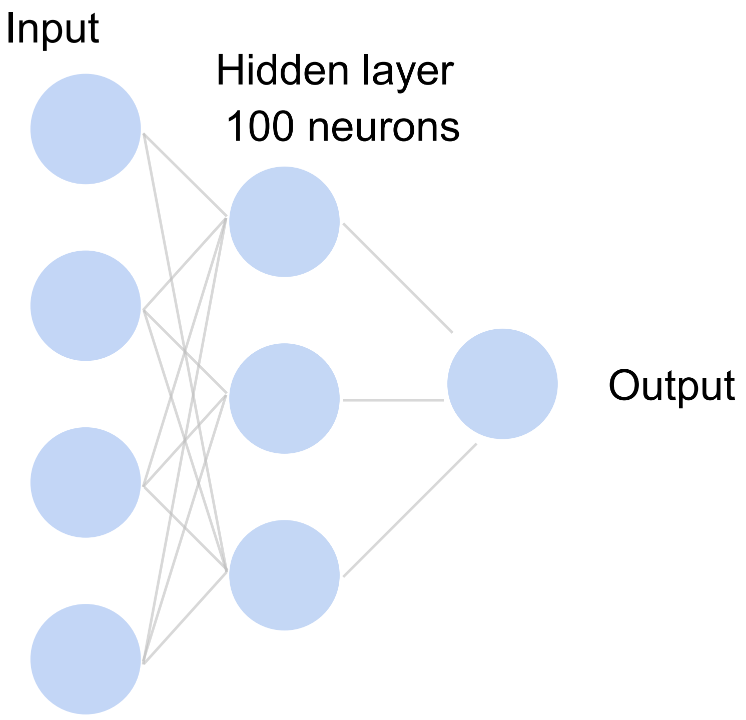


How to train a neural network? Multi-class classification

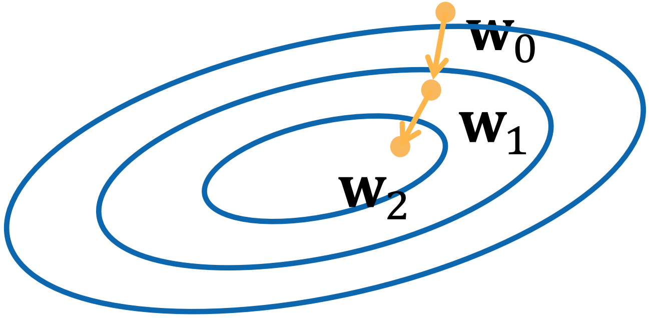
Update the weights W to minimize the loss function

$$L = \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \ell(\mathbf{x}, y)$$

Use gradient descent!



Gradient Descent



- Choose a learning rate $\eta > 0$
- Initialize the model parameters w_0
- For t =1,2,…
  - Update parameters:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\partial L}{\partial \mathbf{w}_{t-1}}$$

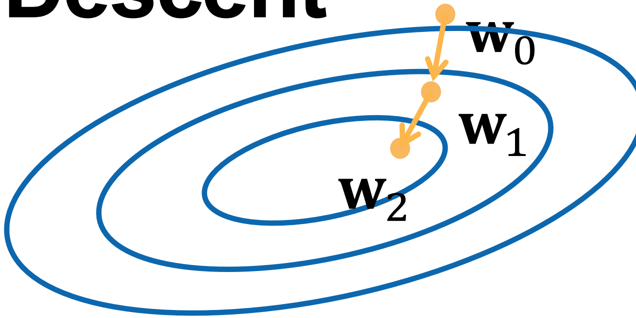
$$= \mathbf{w}_{t-1} - \eta \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \frac{\partial \ell(\mathbf{x}, y)}{\partial \mathbf{w}_{t-1}}$$

D can be very large. Expensive per iteration

The gradient w.r.t. all parameters is obtained by concatenating the partial derivatives w.r.t. each parameter

- Repeat until converges

Minibatch Stochastic Gradient Descent

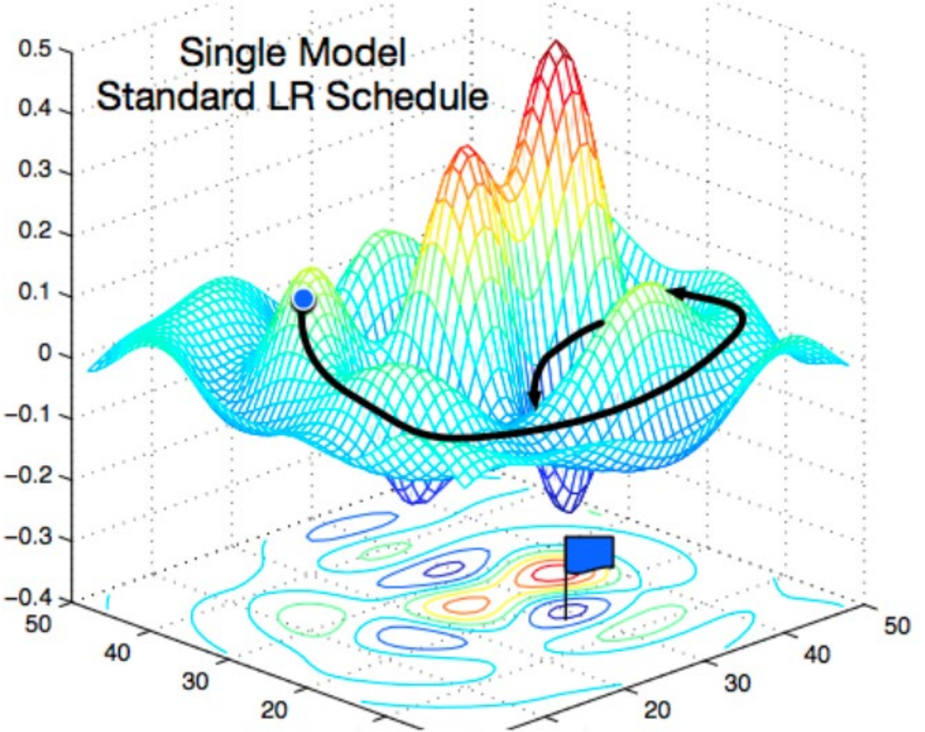


- Choose a learning rate $\eta > 0$
- Initialize the model parameters w_0
- For t =1,2,…
    - **Randomly sample a subset (mini-batch)** $B \subset D$
    - Update parameters:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{1}{|B|} \sum_{(\mathbf{x},y)\in B} \frac{\partial \ell(\mathbf{x}, y)}{\partial \mathbf{w}_{t-1}}$$

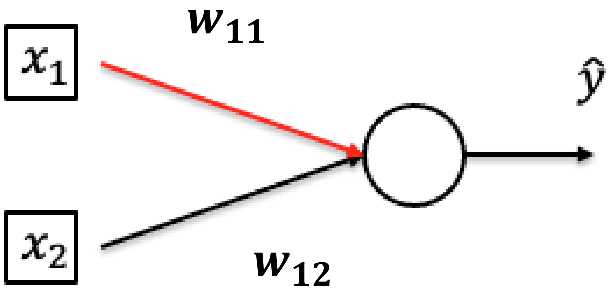
- Repeat until converges

Non-convex Optimization



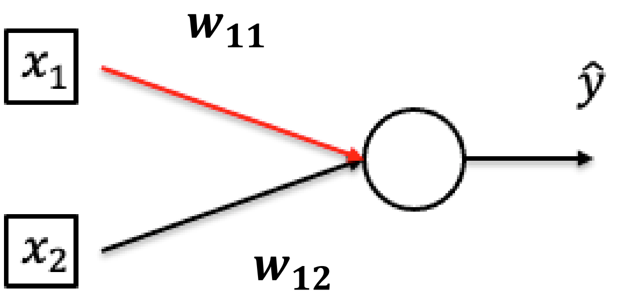
[Gao and Li et al., 2018]

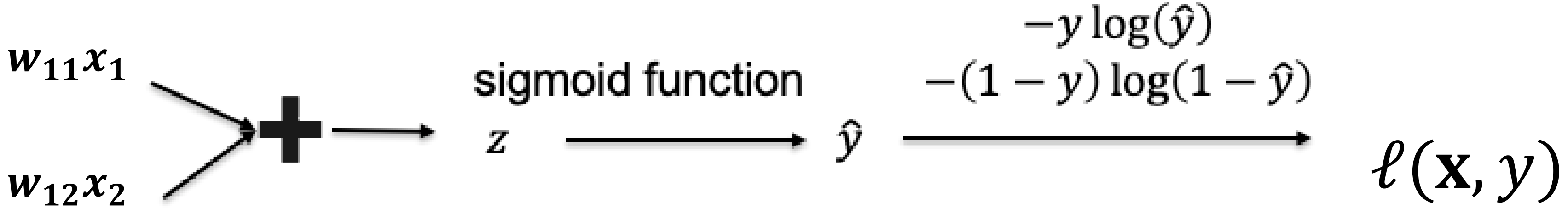
Calculate Gradient (on one data point)



- Want to compute $\frac{\partial \ell(\mathbf{x}, y)}{\partial w_{11}}$
- Data point: $((x_1, x_2), y)$

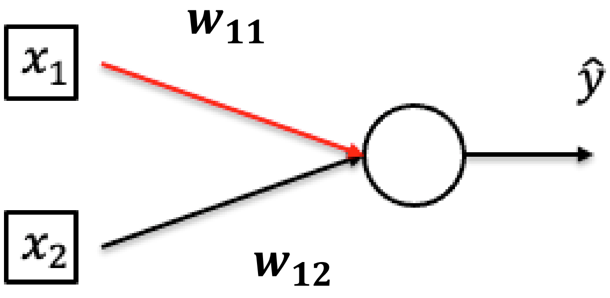
Calculate Gradient (on one data point)

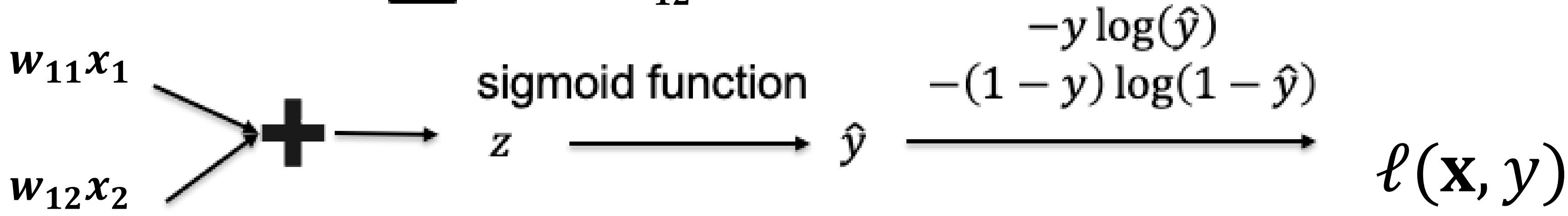




Use chain rule!

Calculate Gradient (on one data point)

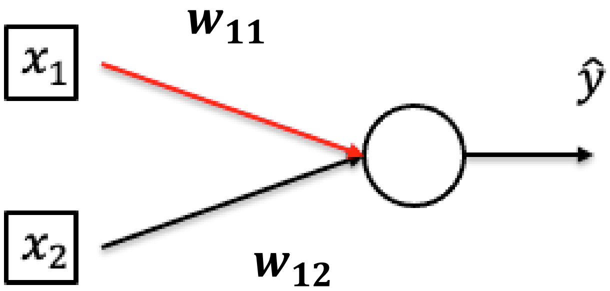


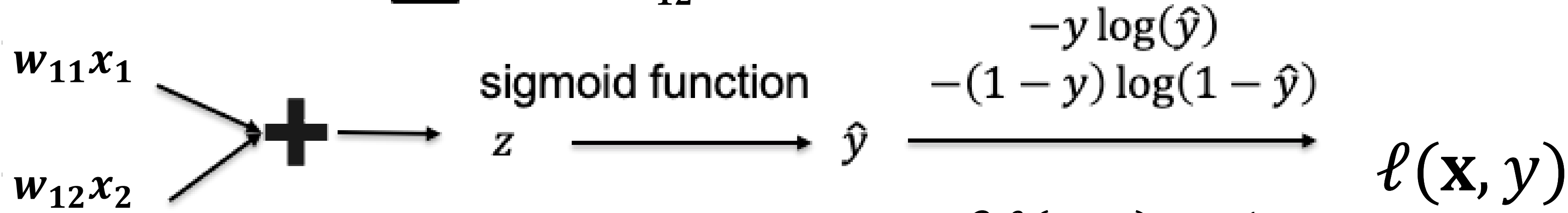


$$\frac{\partial \ell(\mathbf{x}, y)}{\partial \hat{y}} = \frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}$$

- By chain rule: $\frac{\partial l}{\partial w_{11}} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_{11}}$

Calculate Gradient (on one data point)



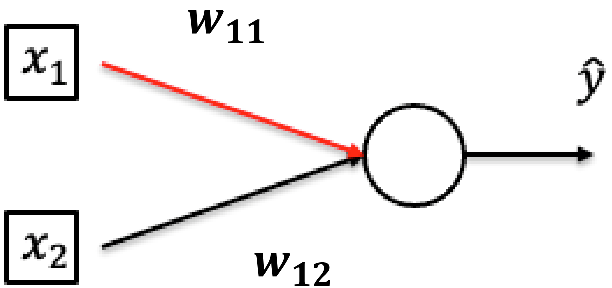


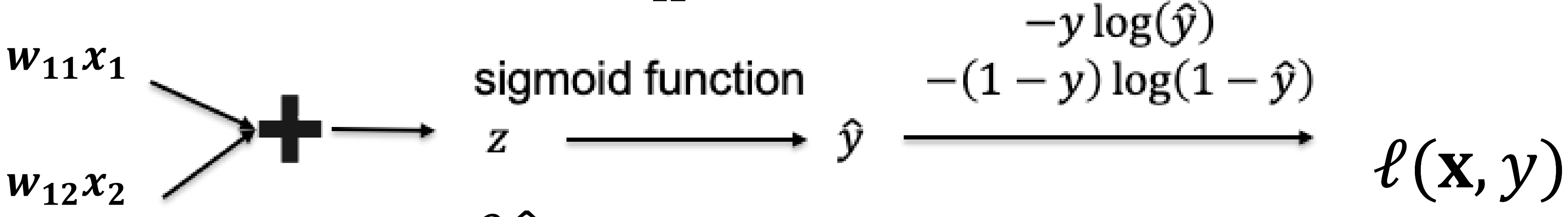
$$\frac{\partial \ell(\mathbf{x}, y)}{\partial \hat{y}} = \frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}$$

- By chain rule:

$$\frac{\partial l}{\partial w_{11}} = \left(\frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}\right)\frac{\partial \hat{y}}{\partial z}\frac{\partial z}{\partial w_{11}}$$

Calculate Gradient (on one data point)

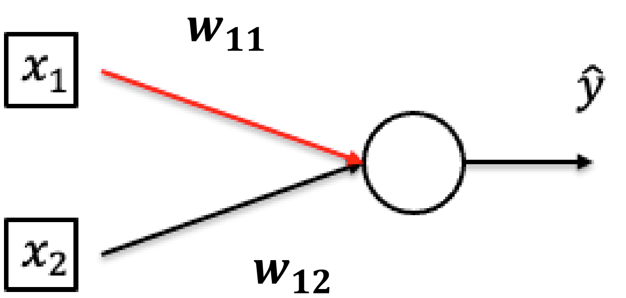


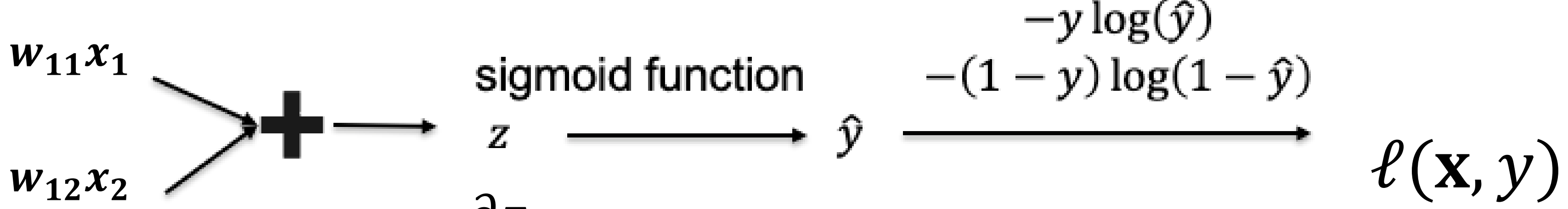


$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

- By chain rule: $\displaystyle \frac{\partial l}{\partial w_{11}} = \left(\frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}\right)\hat{y}(1-\hat{y})\frac{\partial z}{\partial w_{11}}$

Calculate Gradient (on one data point)

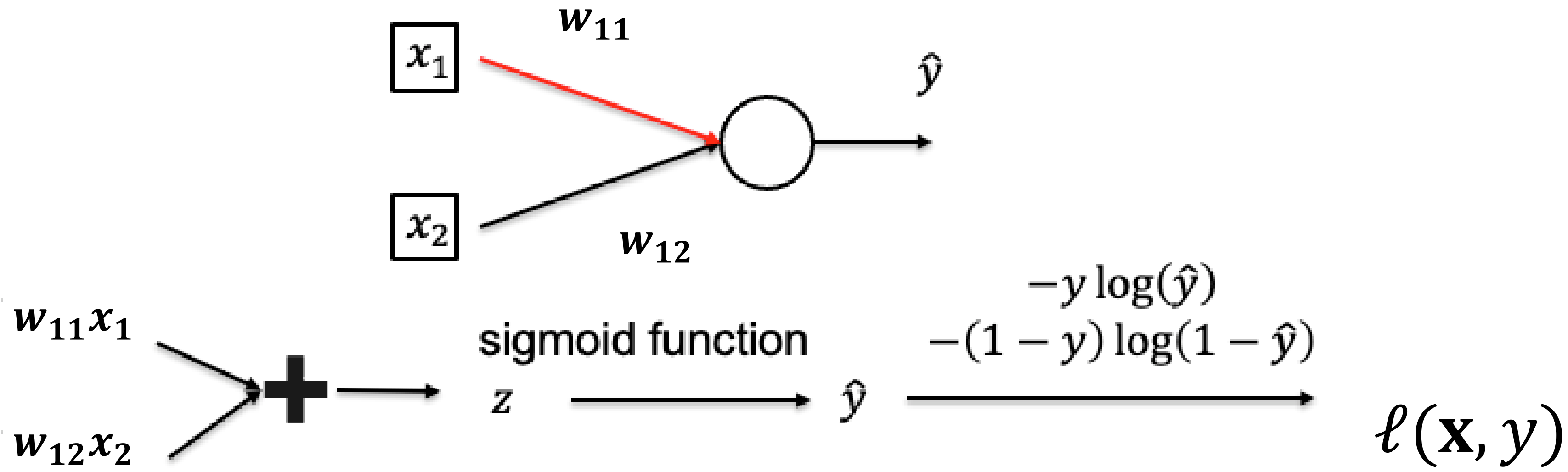




$$\frac{\partial z}{\partial w_{11}} = x_1$$

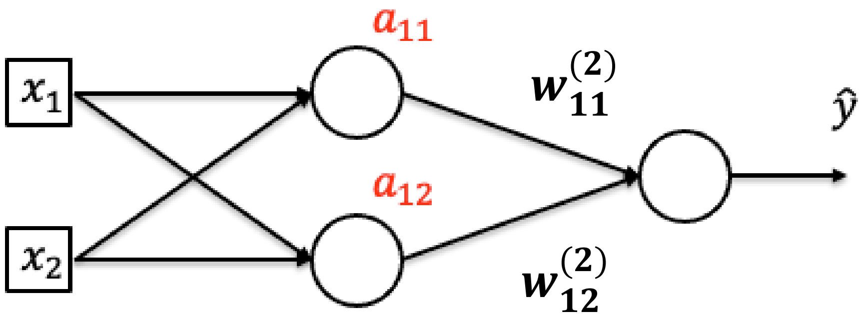
- By chain rule: $$\frac{\partial \boldsymbol{l}}{\partial \boldsymbol{w}_{11}} = \left(\frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}\right)\hat{y}(1-\hat{y})x_1$$

Calculate Gradient (on one data point)

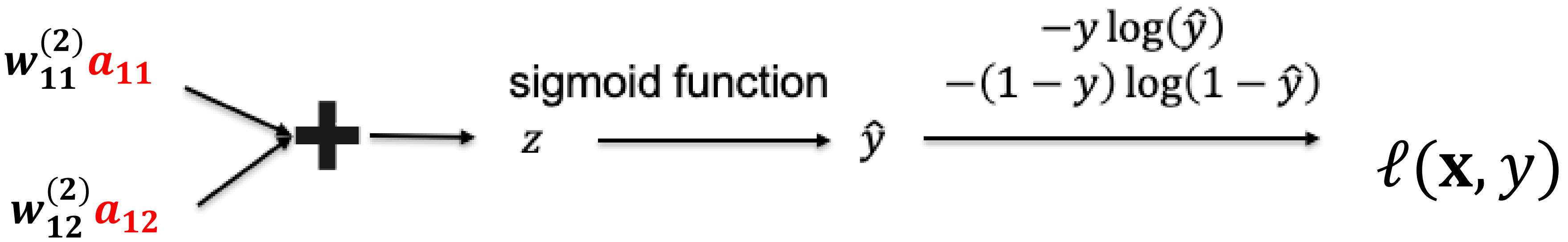


- By chain rule: $$\frac{\partial l}{\partial w_{11}} = (\hat{y} - y)x_1$$

Calculate Gradient (on one data point)

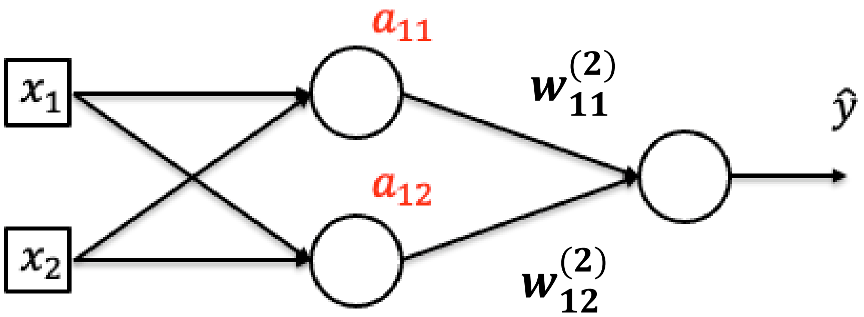


Make it deeper

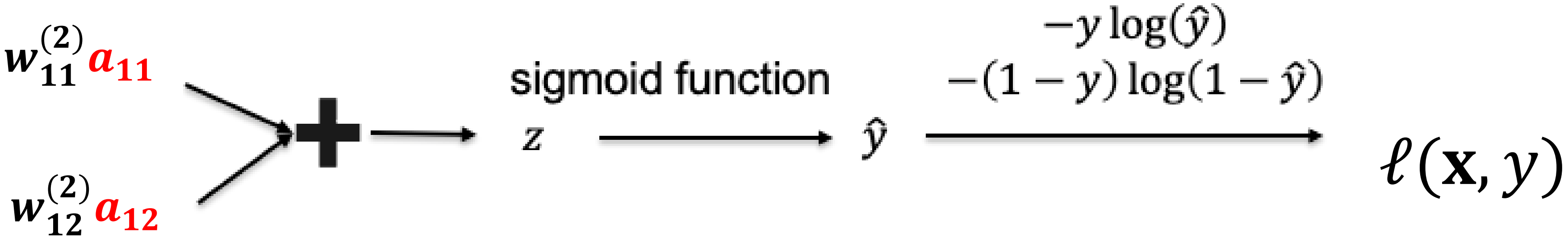


- By chain rule: $\frac{\partial l}{\partial a_{11}} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial a_{11}} = (\hat{y} - y) w_{11}^{(2)}$

Calculate Gradient (on one data point)

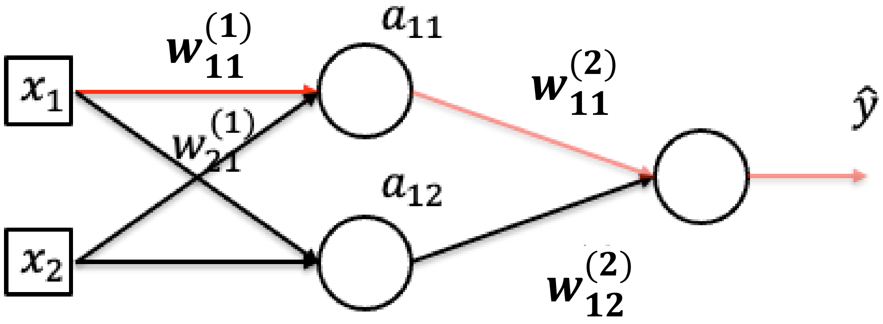


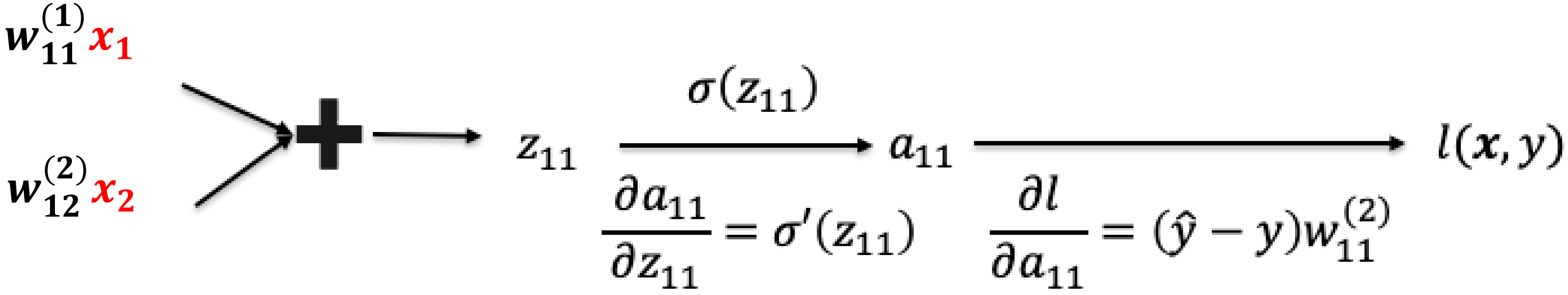
Make it deeper



- By chain rule: $\frac{\partial l}{\partial a_{12}} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial a_{12}} = (\hat{y} - y) w_{12}^{(2)}$

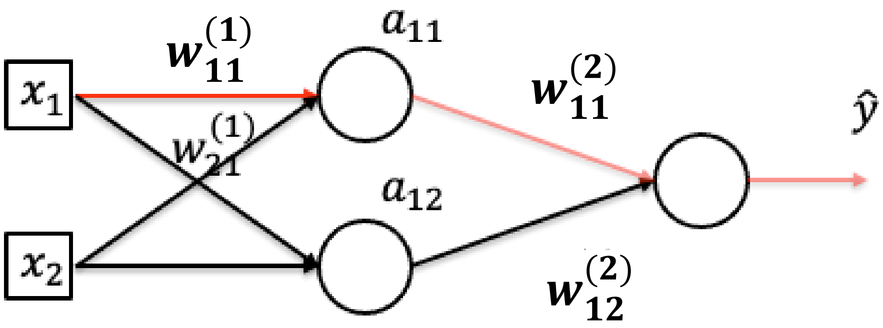
Calculate Gradient (on one data point)

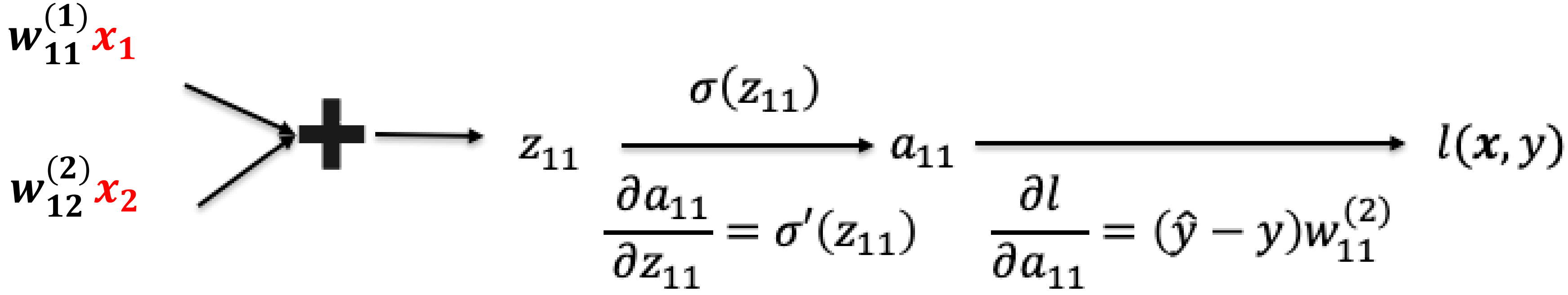




- By chain rule: $\frac{\partial l}{\partial w_{11}^{(1)}} = \frac{\partial l}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} \frac{\partial z_{11}}{\partial w_{11}^{(1)}} = (\hat{y} - y) w_{11}^{(2)} a_{11}(1 - a_{11}) \frac{\partial z_{11}}{\partial w_{11}^{(1)}}$

Calculate Gradient (on one data point)





- By chain rule: $\frac{\partial l}{\partial w_{11}^{(1)}} = \frac{\partial l}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} \frac{\partial z_{11}}{\partial w_{11}^{(1)}} \quad = (\hat{y} - y) w_{11}^{(2)} a_{11} (1 - a_{11}) x_1$

Break & Quiz

Consider a neural network for binary classification using sigmoid activation $\sigma(z) = \frac{1}{1+e^{-z}}$.The network produces output $\hat{y} = \sigma(z)$ where z = 2. Given the binary cross-entropy loss $\ell(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$ and that $\frac{\partial \sigma}{\partial z} = \sigma'(z) = \sigma(z)(1 - \sigma(z))$, what is $\frac{\partial \ell}{\partial z}$ when the true label is y = 1?:

- A. $\sigma(2)$ - 1.
- B. $\sigma(2)$
- C. - $\sigma(2)$
- D. 1-$\sigma(2)$.
- E. -1.

Break & Quiz

Consider a neural network for binary classification using sigmoid activation $\sigma(z) = \frac{1}{1+e^{-z}}$. The network produces output $\hat{y} = \sigma(z)$ where z = 2. Given the binary cross-entropy loss $\ell(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$ and that $\frac{\partial \sigma}{\partial z} = \sigma'(z) = \sigma(z)(1 - \sigma(z))$, what is $\frac{\partial \ell}{\partial z}$ when the true label is y = 1?:

- **A. $\boldsymbol{\sigma(2)}$ - 1.**
- B. $\sigma(2)$
- C. - $\sigma(2)$
- D. 1-$\sigma(2)$.
- E. -1.

$$\frac{\partial l}{\partial z} = \frac{\partial l}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \qquad \frac{\partial \ell(y)}{\partial \hat{y}} = \frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}$$

$$\frac{\partial \hat{y}}{\partial z} = \sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big) = \hat{y}(1 - \hat{y})$$

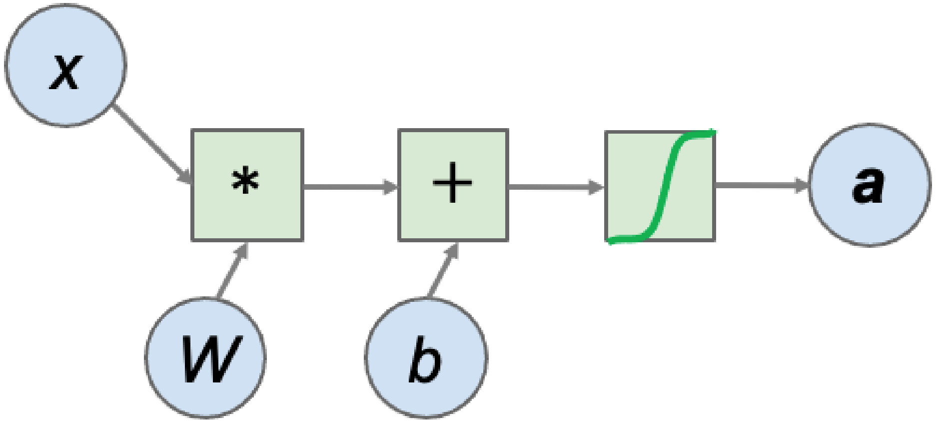
$$\frac{\partial l}{\partial z} = \left(\frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}\right)\hat{y}(1 - \hat{y}) = (\hat{y} - y) = \sigma(2) - 1$$

Computational Graphs

Neural networks as variables + operations

$$\mathbf{a} = sigmoid(\mathbf{Wx} + \mathbf{b})$$

- Can describe with a **computational graph**
- Decompose functions into atomic operations
- Separate data (variables) and computing (operations)



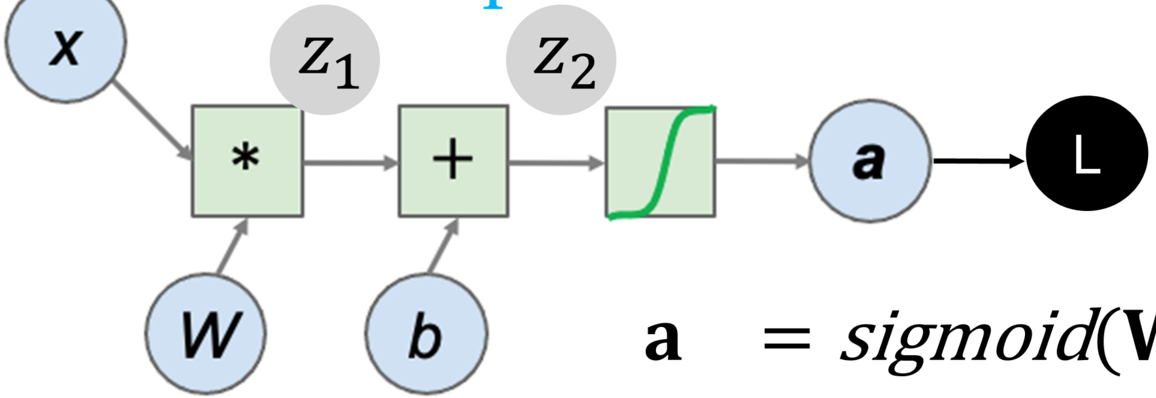
Calculate gradient: backpropagation with chain rule

- Define a loss function L, must compute $\frac{\partial L}{\partial \mathbf{W}}, \frac{\partial L}{\partial b}$ for all weights and biases.

- Gradient to a variable =

gradient on the top x gradient from the current operation

$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{z_1}} \frac{\partial \boldsymbol{z_1}}{\partial \boldsymbol{W}}$$



$$\mathbf{a} = sigmoid(\mathbf{Wx} + \mathbf{b})$$

Break & Quiz

Which of the following specifically refers to the process by which the gradients with respect to the neural network are computed, once the loss value is already computed?

A. Forward pass
B. Gradient vanishing
C. Backpropagation
D. Learning rate adjustment
E. Activation

Break & Quiz

Which of the following specifically refers to the process by which the gradients with respect to the neural network are computed, once the loss value is already computed?

A. Forward pass

B. Gradient vanishing

C. **Backpropagation**

D. Learning rate adjustment

E. Activation

Numerical Stability

Gradients for Neural Networks

Compute the gradient of the loss ℓ w.r.t. $\mathbf{W}_t$

$$\frac{\partial \ell}{\partial \mathbf{W}^t} = \frac{\partial \ell}{\partial \mathbf{h}^d} \frac{\partial \mathbf{h}^d}{\partial \mathbf{h}^{d-1}} \cdots \frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} \frac{\partial \mathbf{h}^t}{\partial \mathbf{W}^t}$$

Multiplication of *many* matrices

Wikipedia

Two Issues for Deep Neural Networks

$$\prod_{i=t}^{d-1} \frac{\partial \mathbf{h}^{i+1}}{\partial \mathbf{h}^{i}}$$

Gradient Exploding

$$1.5^{100} \approx 4 \times 10^{17}$$

Gradient Vanishing

$$0.8^{100} \approx 2 \times 10^{-10}$$

Issues with Gradient Exploding

Value out of range: infinity value (NaN)

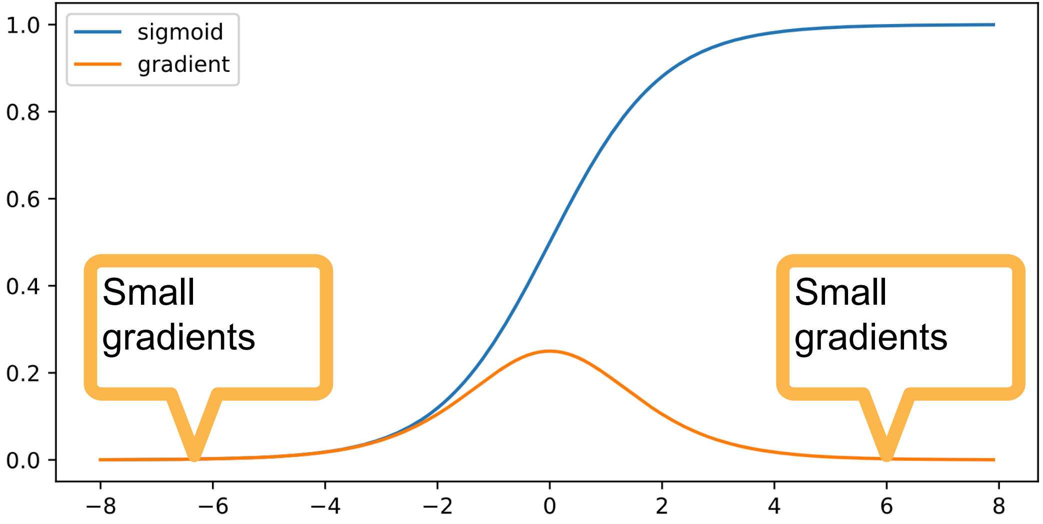
Sensitive to learning rate (LR)

- Not small enough LR → larger gradients
- Too small LR → No progress
- May need to change LR dramatically during training

Gradient Vanishing

Use sigmoid as the activation function

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



Issues with Gradient Vanishing

Gradients with value 0

No progress in training

- No matter how to choose learning rate

Severe with bottom layers (those near the input)

- Only top layers (near output) are well trained
- No benefit to make networks deeper

Break & Quiz

When training a neural network, the vanishing gradient problem is encountered. What is a primary and direct consequence of this phenomenon on the network's parameters?

A. Values get out of range resulting in 'NaN' (Not a Number) values.

B. The model quickly memorizes the training data, leading to a low training error but a very high-test error (overfitting).

C. The parameters in the initial layers of the network (those closer to the input) are not being updated

D. All parameters within a single layer tend to converge to the same value, preventing the layer from learning diverse features.

E. The network's parameters are updated in a high-variance, chaotic manner, causing the optimization process to oscillate and never settle in a good local minimum.

Break & Quiz

When training a neural network, the vanishing gradient problem is encountered. What is a primary and direct consequence of this phenomenon on the network’s parameters?

A. Values get out of range resulting in ‘NaN‘ (Not a Number) values.

B. The model quickly memorizes the training data, leading to a low training error but a very high-test error (overfitting).

C. **The parameters in the initial layers of the network (those closer to the input) are not being updated**

D. All parameters within a single layer tend to converge to the same value, preventing the layer from learning diverse features.

E. The network’s parameters are updated in a high-variance, chaotic manner, causing the optimization process to oscillate and never settle in a good local minimum.

How to classify

Cats vs. dogs?

Dual

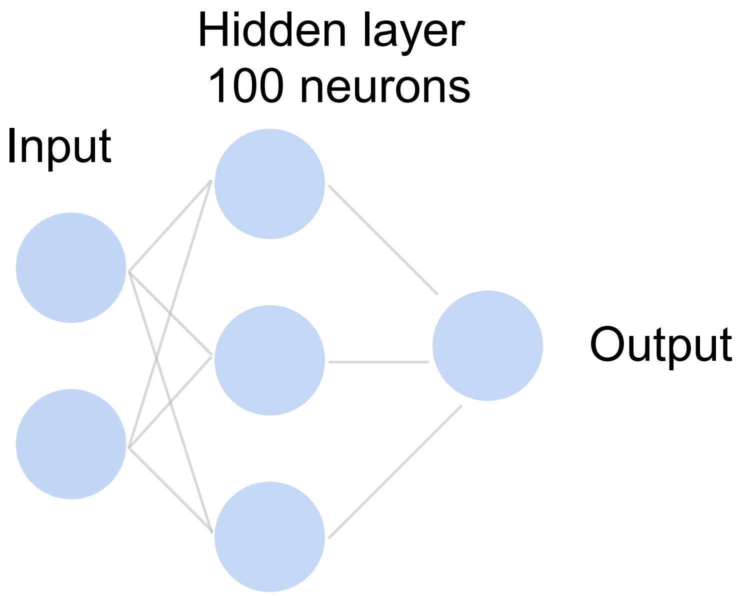
12MP

wide-angle and telephoto cameras

36M floats in a RGB image!

Fully Connected Networks

Cats vs. dogs?

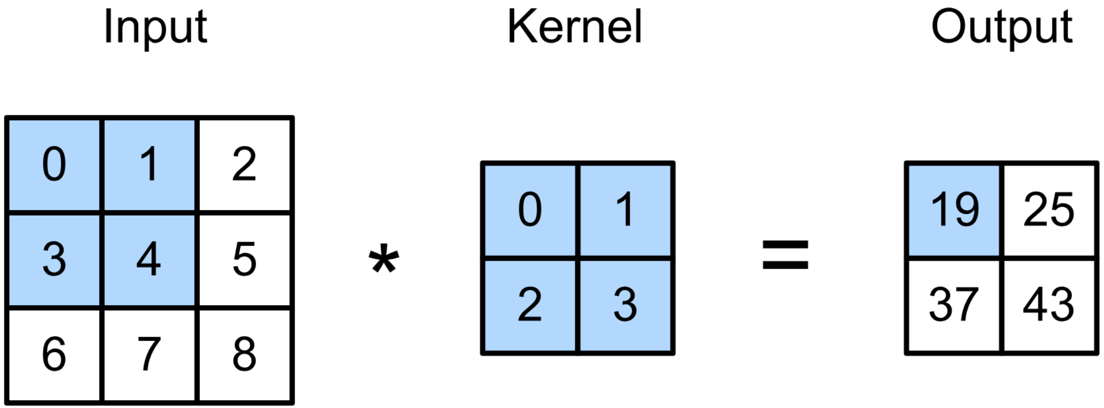


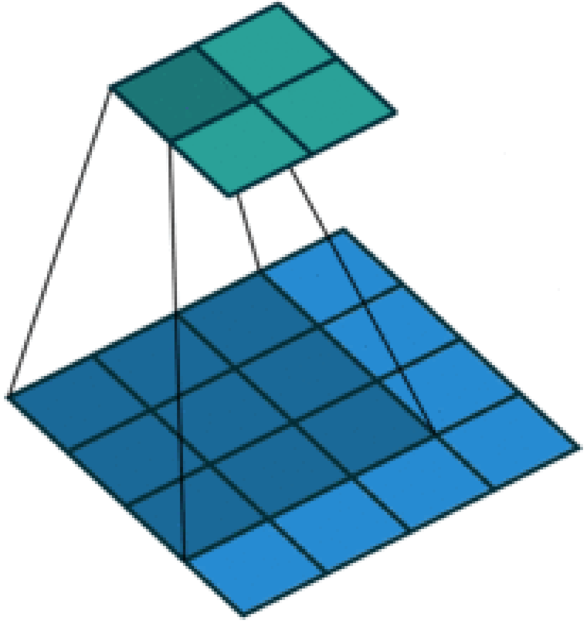
~ 36M elements x 100 = **~3.6B** parameters!

Why Convolution?

- Translation Invariance
- Locality

Review: 2-D Convolution

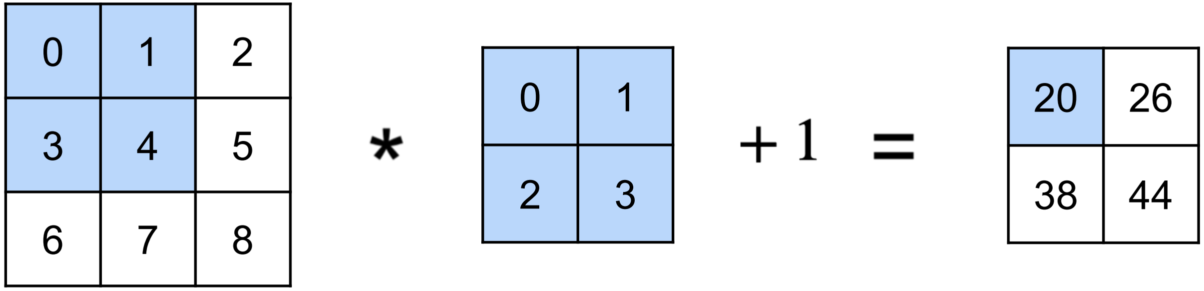




(vdumoulin@ Github)

$$
\begin{aligned}
0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 &= 19, \\
1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 &= 25, \\
3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 &= 37, \\
4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 &= 43.
\end{aligned}
$$

2-D Convolution Layer



X: $\boldsymbol{n_h}$ **x** $\boldsymbol{n_w}$ input matrix

W: $\boldsymbol{k_h}$ **x** $\boldsymbol{k_w}$ kernel matrix

b: scalar bias

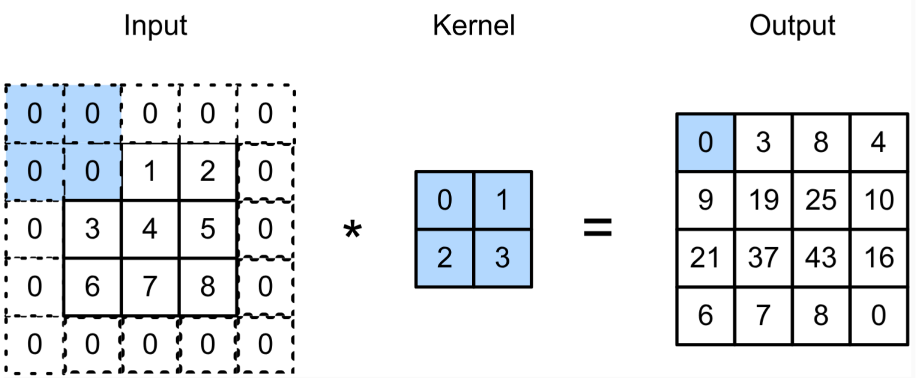
Y: $(n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

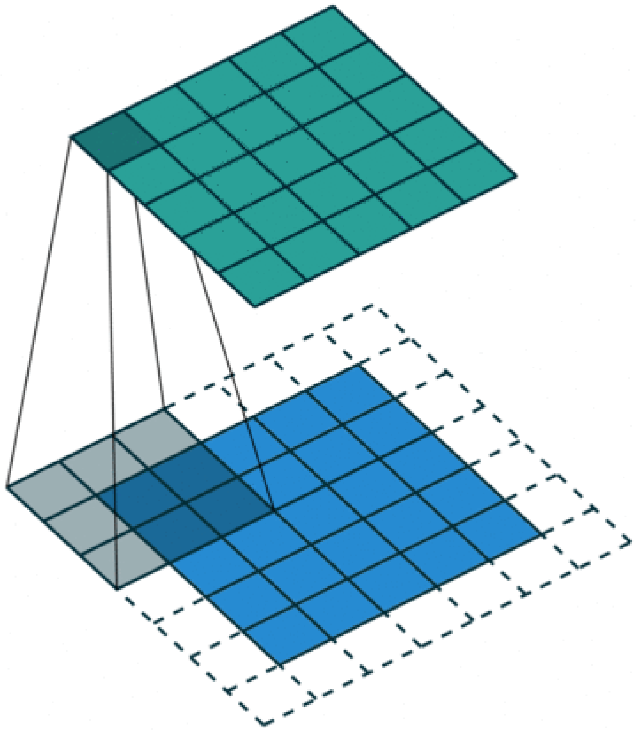
Y = X * W + b

W and *b* are learnable parameters

Convolutional Layers: Padding

Padding adds rows/columns around input

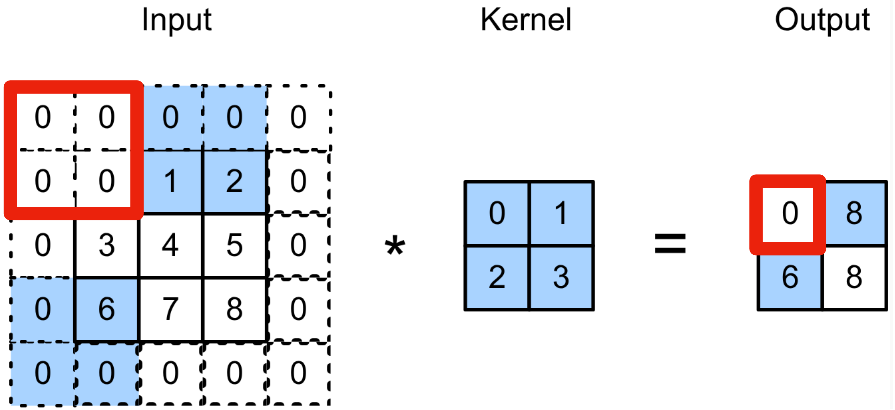




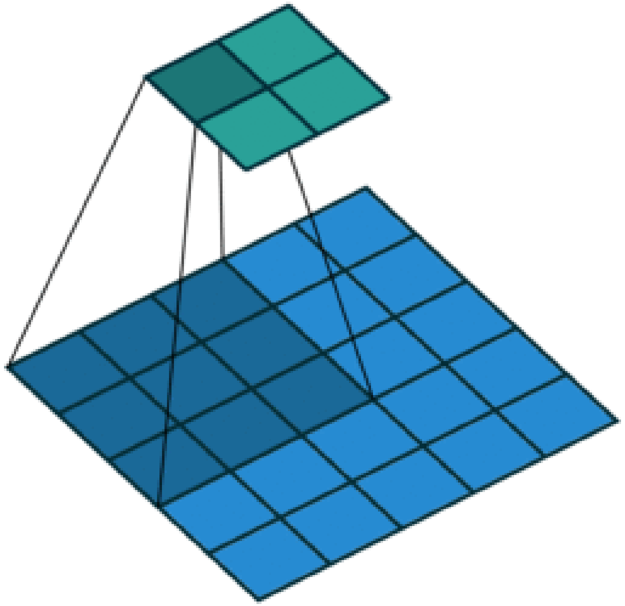
Stride

Stride is the #rows / #columns per slide

Example: strides of 3 and 2 for height and width



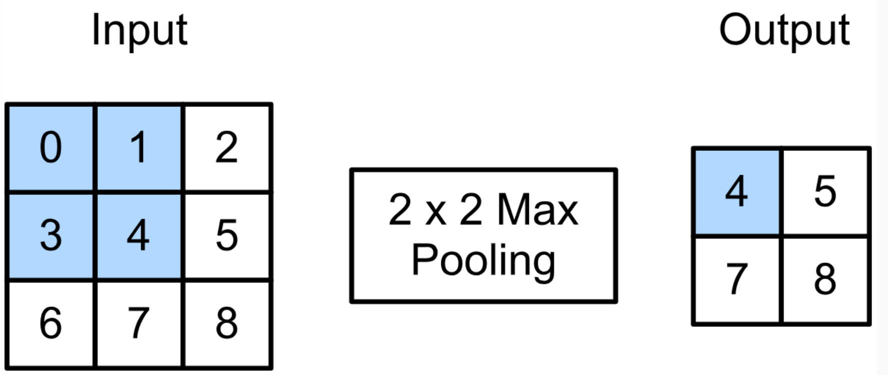
$$0\times0 + 0\times1 + 1\times2 + 2\times3 = 8$$
$$0\times0 + 6\times1 + 0\times2 + 0\times3 = 6$$



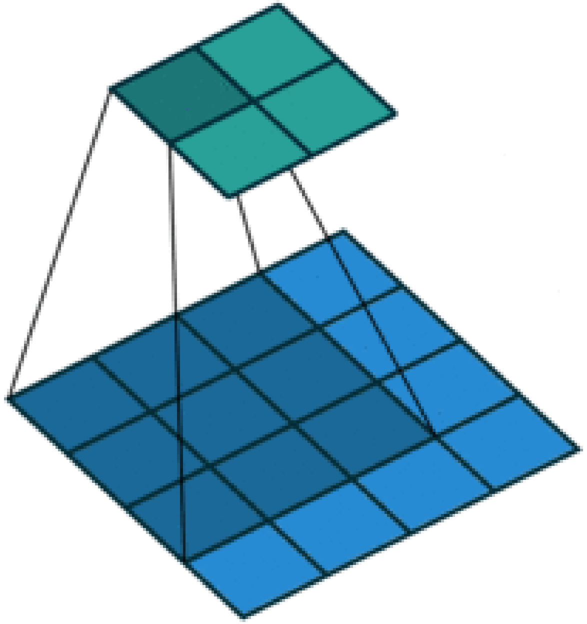
Stride 2,2

2-D Max Pooling

Returns the maximal value in the sliding window



$$max(0,1,3,4) = 4$$



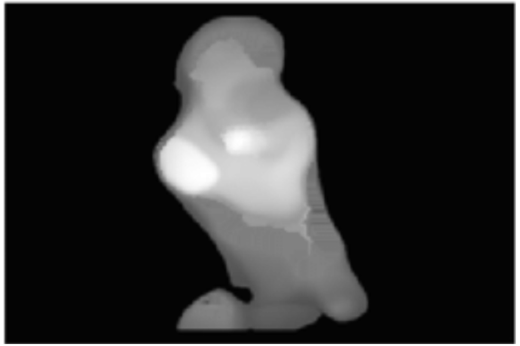
Average Pooling

Max pooling: the strongest pattern signal in a window

Average pooling: replace max with mean in max pooling

- The average signal strength in a window

Max pooling



Average pooling

Output shape

Kernel/filter size

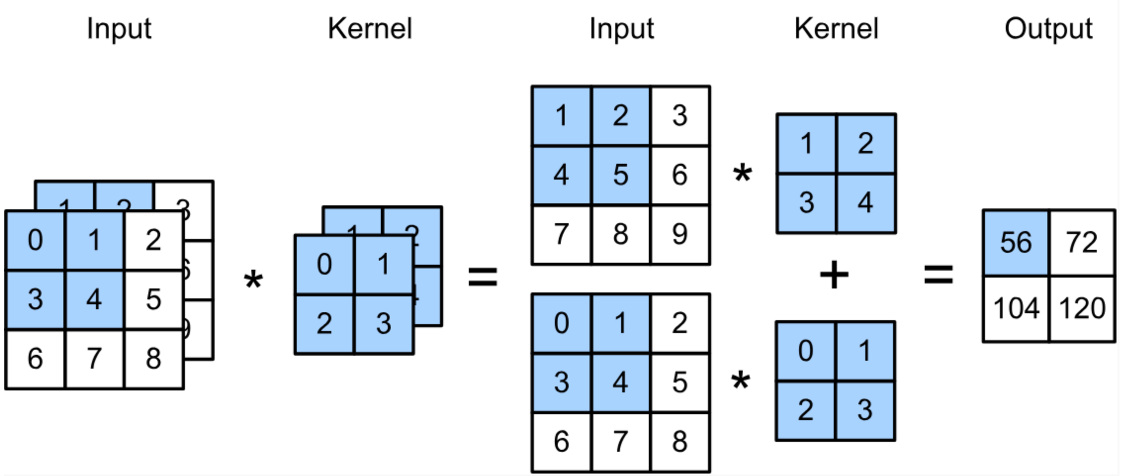
69

$$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$

Input size **Pad** **Stride**

Review: Multiple Input Channels

Have a kernel matrix for each channel, and then sum results over channels

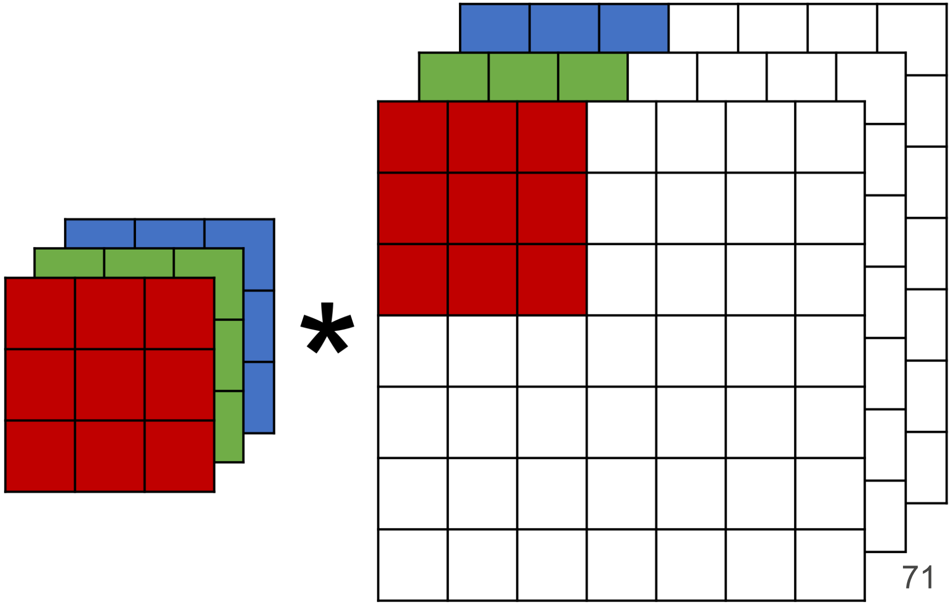


$$(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4)$$
$$+(0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3)$$
$$= 56$$

Multiple Input Channels

Input and kernel can be 3D, e.g., an RGB image have 3 channels

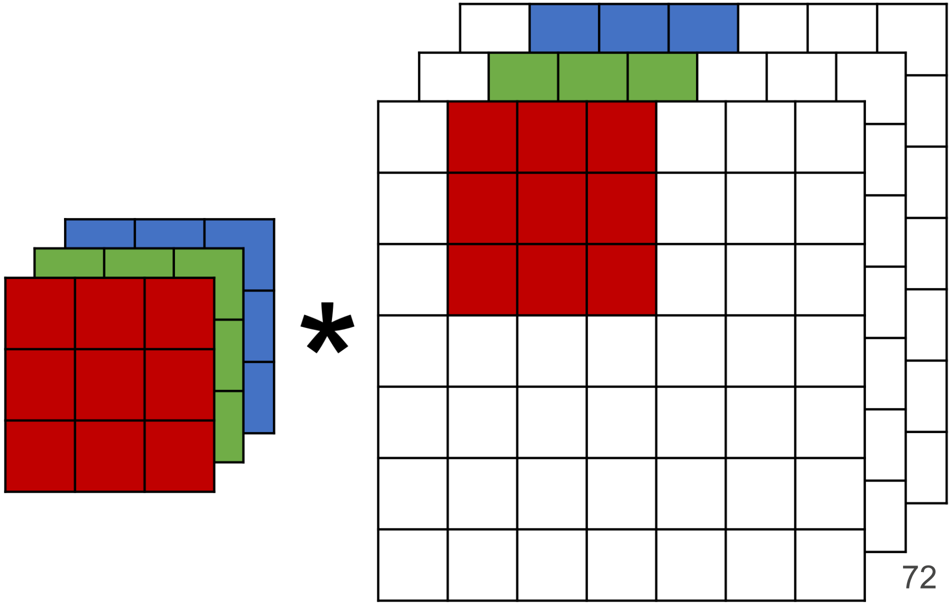
Have a kernel for each channel, and then sum results over channels



Multiple Input Channels

Input and kernel can be 3D, e.g., an RGB image have 3 channels

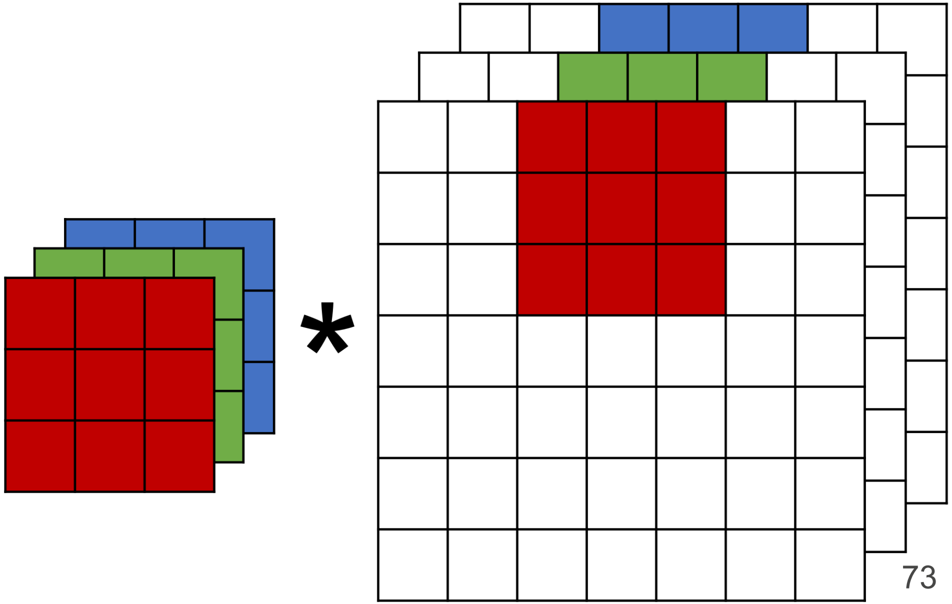
Have a kernel for each channel, and then sum results over channels



Multiple Input Channels

Input and kernel can be 3D, e.g., an RGB image have 3 channels

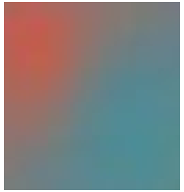
Have a kernel for each channel, and then sum results over channels



Multiple Input Channels

Input and kernel can be 3D, e.g. RGB image has 3 channels

Also call each 3D kernel a "**filter**", which produces only **one** output channel (due to summation over channels)



One filter
(3 channels)

*

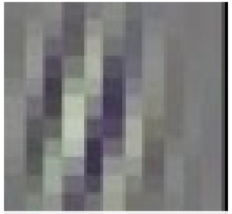
RGB (3 input channels)

Multiple filters (in one layer)

Apply multiple filters on the input

Each filter may learn different features about the input

Each filter (3D kernel) produces one output channel



A different filter

*

RGB (3 input channels)

Break & Quiz

An RGB image of size 227 × 227 × 3 is passed to a convolutional layer with 96 filters, each 11 × 11 in spatial extent, stride 4, no padding. Every filter has its own bias. Approximately how many learnable parameters (weights + biases) are in this layer?:

- A. ~ 10k parameters.
- B. ~ 50k parameters.
- C. ~ 35k parameters.
- D. ~ 40k parameters.
- E. ~ 15k parameters.

Break & Quiz

An RGB image of size 227 × 227 × 3 is passed to a convolutional layer with 96 filters, each 11 × 11 in spatial extent, stride 4, no padding. Every filter has its own bias. Approximately how many learnable parameters (weights + biases) are in this layer?:

- A. ~ 10k parameters.
- B. ~ 50k parameters.
- **C. ~ 35k parameters.**
- D. ~ 40k parameters.
- E. ~ 15k parameters.

One filter has 11 × 11 × 3 = 363 weights plus 1 bias. Hence there are 364 trainable parameters per filter. Hence, the total number of trainable parameters are 364 × 96 = 34944 ~ 35k.

Break & Quiz

2D convolutional kernel of size 3 × 3, stride 2 × 2, and padding 2 × 2 is applied to aninput of size 9 × 13. What is the output size?:

- A. 3 × 5.
- B. 3 × 7.
- C. 5 × 7.
- D. 5 × 9.
- E. 9 × 13.

Break & Quiz

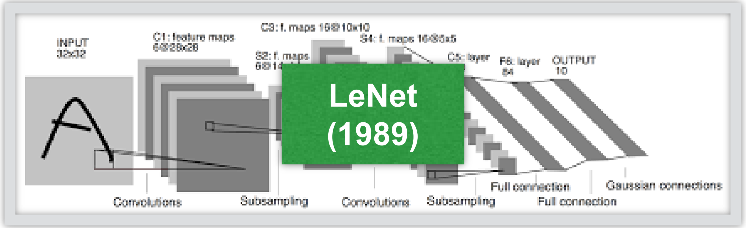
2D convolutional kernel of size 3×3, stride 2×2, and padding 2×2 is applied to an input of size 9×13. What is the output size?:

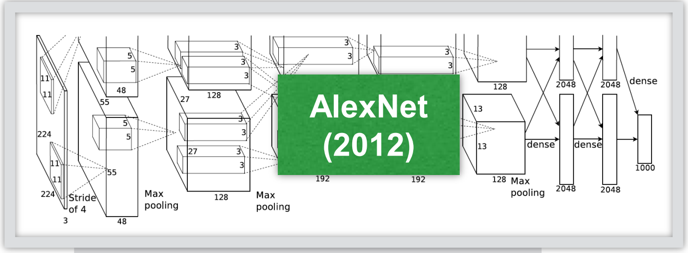
- A. 3×5.
- B. 3×7.
- **C. 5×7**.
- D. 5×9.
- E. 9×13.

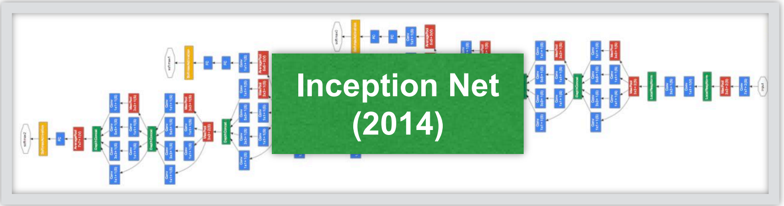
Width = (9 -3+ 2+2)/2 = 5

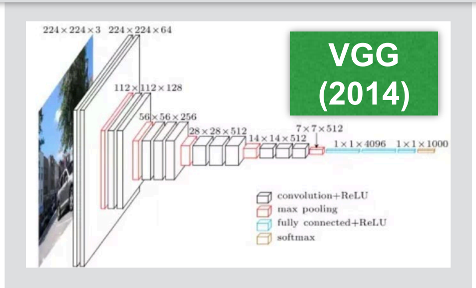
Height = (13-3+2+2)/2 = 7

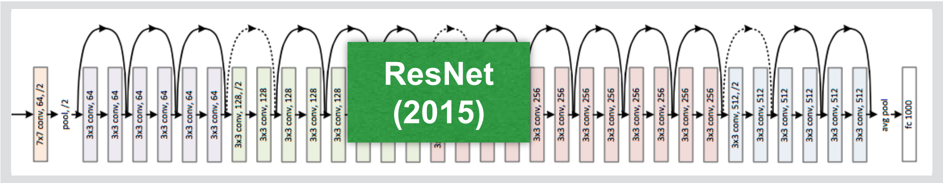
Evolution of neural net architectures

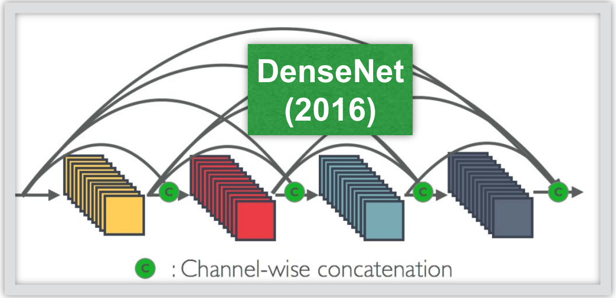








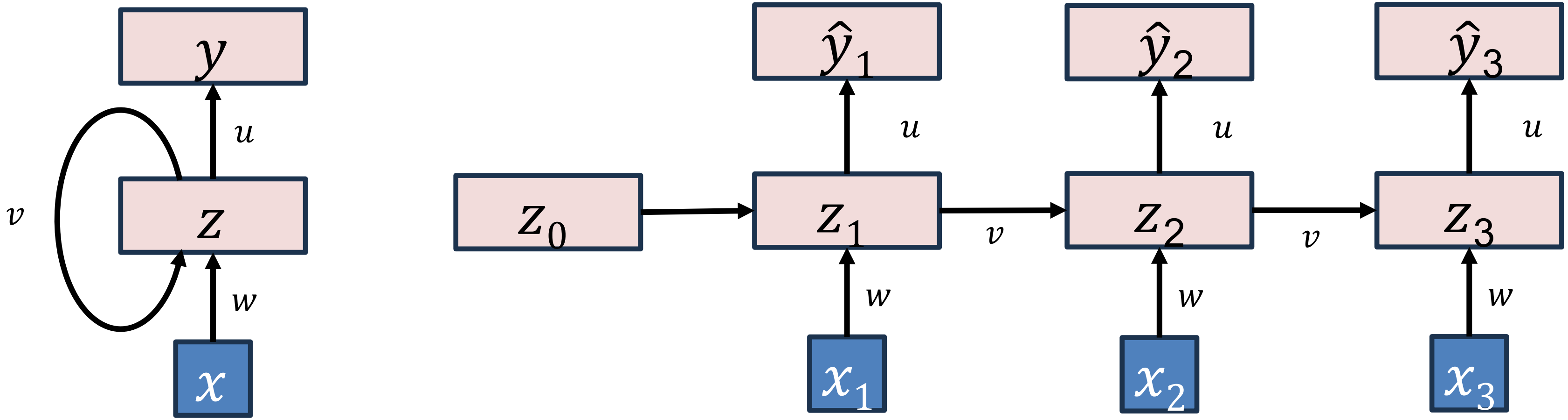




Recurrent Neural Networks (RNNs)

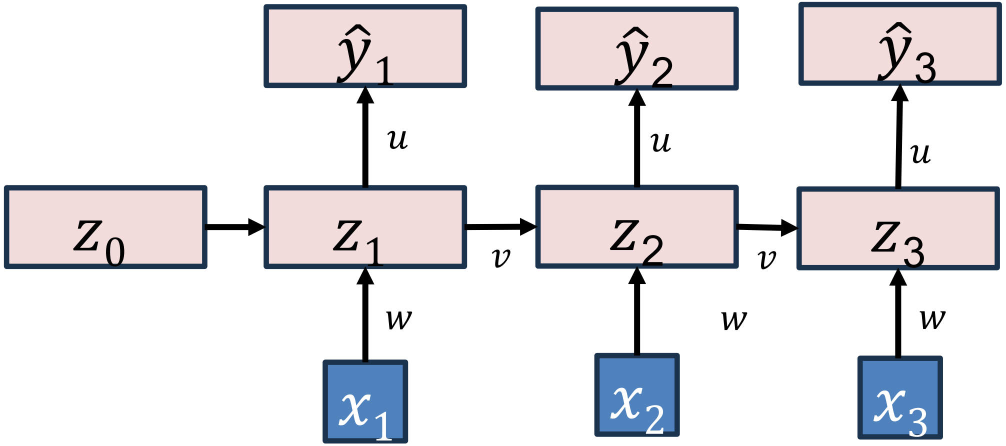
Recurrent Neural Networks (RNNs)

- RNNs introduce **cycles** in the computational graph
- Allowing information to persist; **memory**



Recurrent Neural Networks (RNNs)

- In each time step, the **input value** and the **output of previous hidden state** are used in the computation
- Internal state, **memory** – inputs received at earlier time steps affect the RNN's response to the current input.



Activation Function

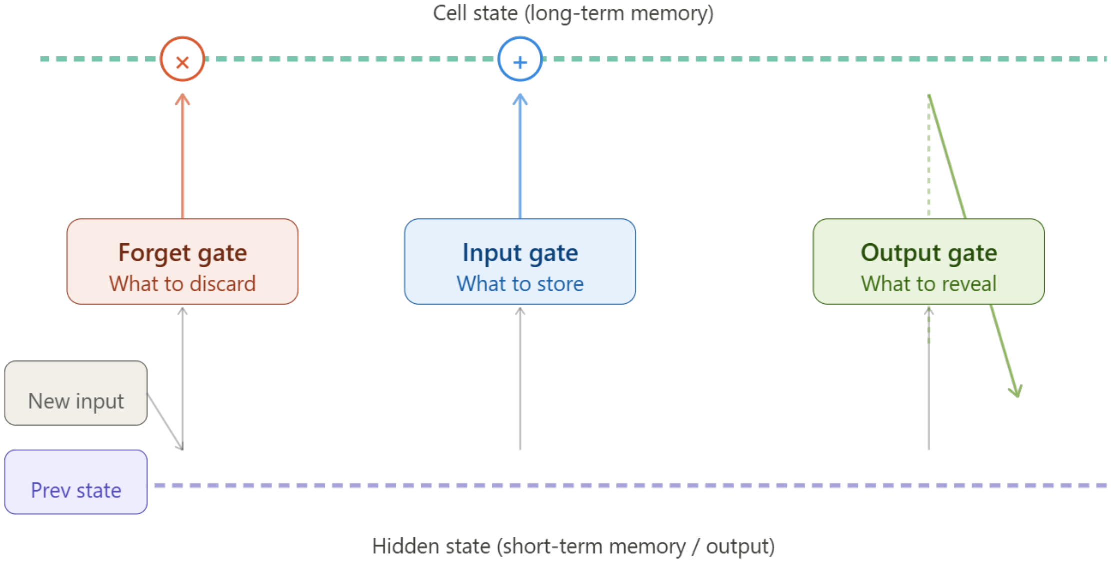
$$\hat{y}_t = g_y(Uz_t)$$

$$z_t = g_z(Vz_{t-1} + Wx + b)$$

Activation Function

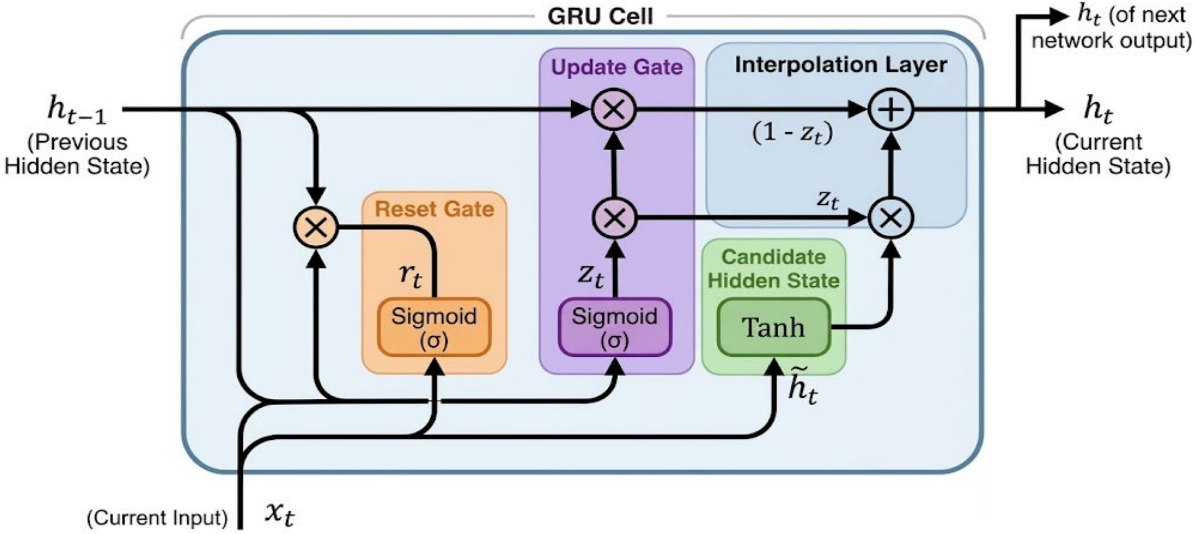
Long Short-term Memory (LSTM)

- Cell state (c): Long-term memory component of LSTM, **copied** from time-step to time-step. (In contrast, the basic RNN multiplies its memory by a weight matrix)
- Hidden State (h): The short-term memory/ working output



Variant: Gated Recurrent Unit (GRU)

- Hidden state acts as both long-term memory and current output
- The Reset Gate $(\boldsymbol{r_t})$ decides how much past context to use for the candidate.
- The Update Gate $(\boldsymbol{z_t})$ decides the blend ratio between old state and candidate.
- Candidate Hidden State $(\tilde{h}_t)$ a proposed new hidden state using (possibly reset) past + current input
- New hidden state $h_t = z_t \times h_{t-1} + (1 - z_t) \times \tilde{h}_t$



Break & Quiz

Which option identifies the central computational property that RNNs possess relative to feedforward networks?:

- A. They compute gradients without backpropagation..
- B. They use residual connections to improve training.
- C. They perform parallel computation across all tokens simultaneously.
- D. They require no activation functions to compute transitions..
- E. They maintain a hidden state that carries information across time steps.

Break & Quiz

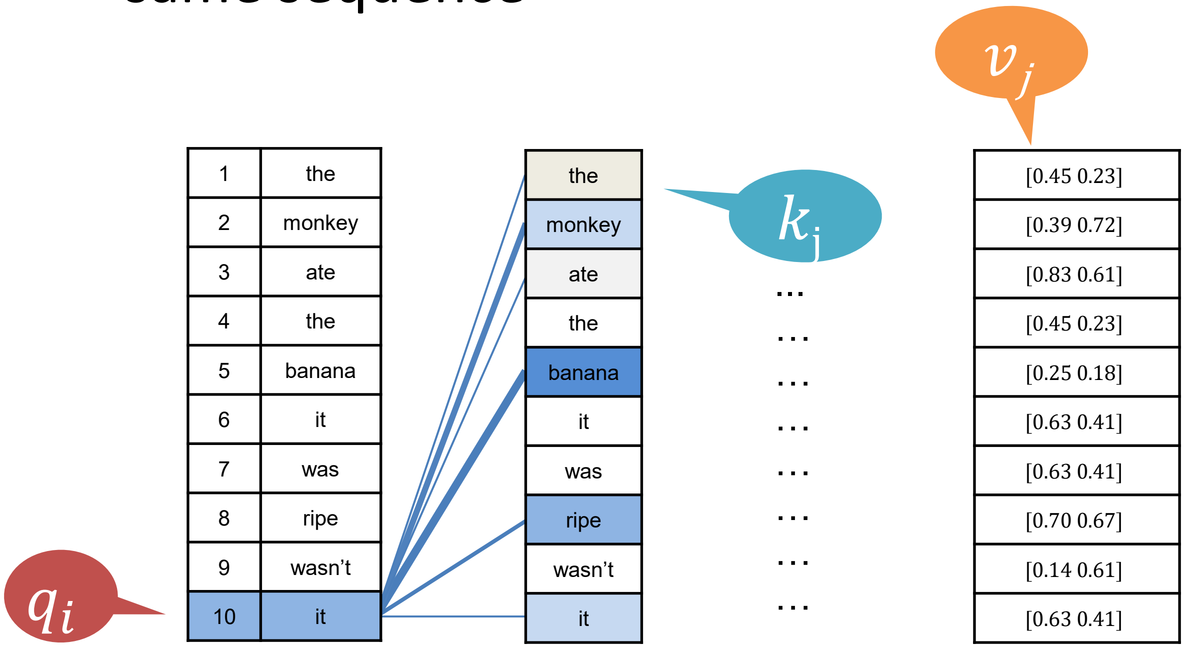
Which option identifies the central computational property that RNNs possess relative to feedforward networks?:

- A. They compute gradients without backpropagation..
- B. They use residual connections to improve training.
- C. They perform parallel computation across all tokens simultaneously.
- D. They require no activation functions to compute transitions..
- **E. They maintain a hidden state that carries information across time steps.**

Attention & Transformers

The Attention Mechanism

Each token attends to all tokens in the same sequence



$$r_{ij} = \frac{\langle q_i, k_j \rangle}{\sqrt{d}}$$

$$p_{i,:} = \mathrm{softmax}(r_{i,:})$$

$$c_i = \sum_j p_{ij} \cdot v_j$$

Notation for Attention

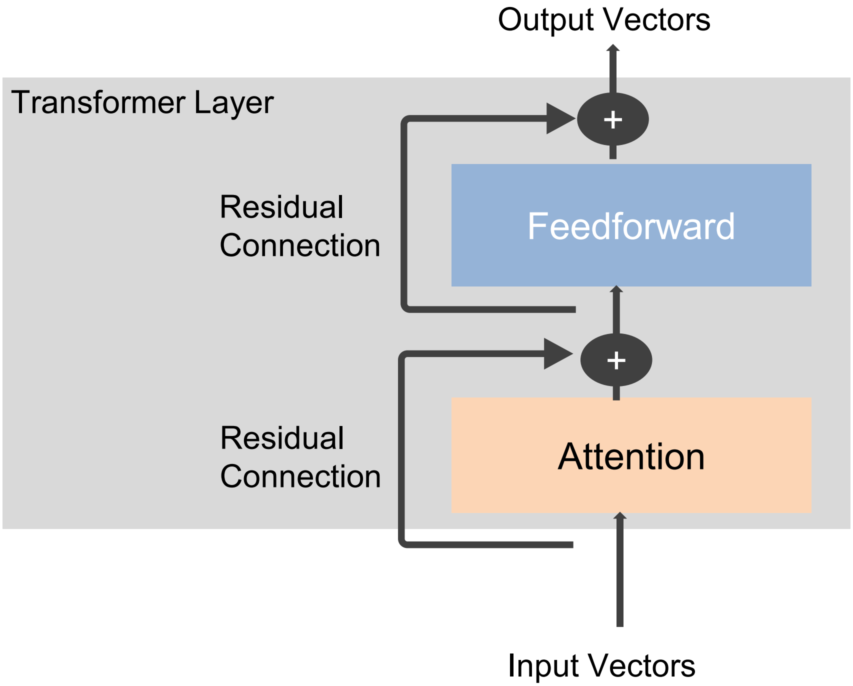
Queries, keys and values are written as matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)\text{V}$$

From Attention to Transformer

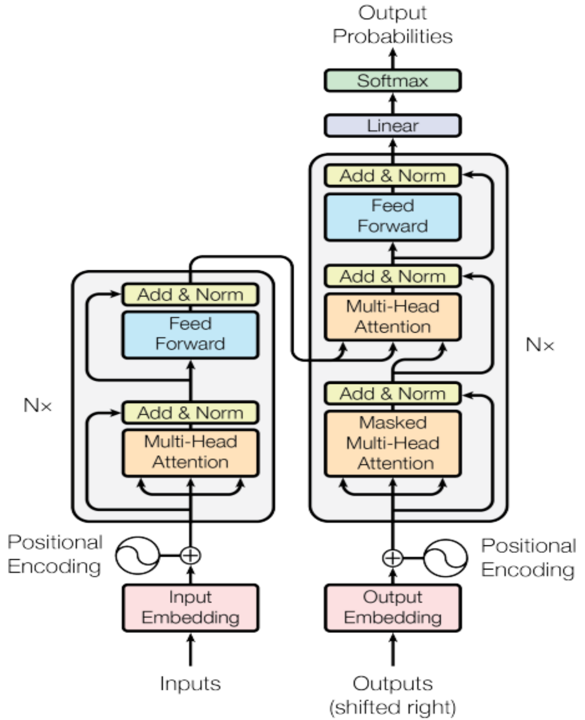
A single layer transformer consists of:

- Attention Mechanism
- Feed-Forward Network
- Residual Connections



Transformer Architecture

- Encoder–Decoder structure
- **Encoder**: maps an input sequence to a sequence of continuous representations z.
  - Useful for classification
- **Decoder**: Given z, the decoder generates an output sequence of symbols one element at a time.
  - Useful for generation



Vaswani, A., et al. (2017). Attention Is All You Need

Break & Quiz

In the context of the Transformer model, what are the primary roles of the "Encoder“ and the "Decoder“?

- A. Encoders are used for text generation, while Decoders are used for text classification.
- B. Encoders process the input sequence into continuous representations, and Decoders generate the output sequence one element at a time.
- C. Both Encoders and Decoders are used solely for understanding the input sequence without generating output.
- D. Both Encoders and Decoders function identically to map inputs to continuous representations.
- E. The Encoder compresses the input into a single fixed-length vector without attention, while the Decoder uses that single vector to reproduce the input sequence exactly for reconstruction tasks.

Break & Quiz

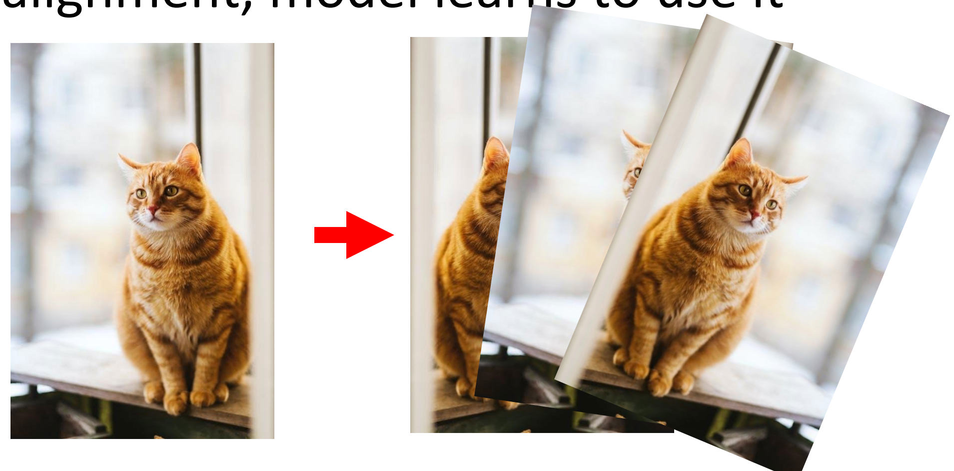
In the context of the Transformer model, what are the primary roles of the "Encoder“ and the "Decoder“?

- A. Encoders are used for text generation, while Decoders are used for text classification.
- **B. Encoders process the input sequence into continuous representations, and Decoders generate the output sequence one element at a time.**
- C. Both Encoders and Decoders are used solely for understanding the input sequence without generating output.
- D. Both Encoders and Decoders function identically to map inputs to continuous representations.
- E. The Encoder compresses the input into a single fixed-length vector without attention, while the Decoder uses that single vector to reproduce the input sequence exactly for reconstruction tasks.

Data Augmentation

Augmentation: transform + add new samples to the training set

- Transformations: based on domain
- Idea: build **invariances** into the model
  - **Ex**: if all images have same alignment, model learns to use it
- Keep the label the same!



Other Domains

Not just for image data. For example, on text:

- Substitution
  - E.g., “It is a **great** day” ➔ “It is a **wonderful** day”
  - Use a thesaurus for particular words
  - Or, use a model. Pre-trained word embeddings, language models
- Back-translation
  - “Given the low budget and production limitations, this movie is very good.” ➔ “There are few budget items and production limitations to make this film a really good one”

Xie **et al**: “Unsupervised Data Augmentation for Consistency Training”

Other Forms of Regularization

Classic regularizations

1. Modify loss functions

   Ex: regularized least squares LR

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + x_i^T \theta - y_i)^2 + \lambda \|\theta\|_2^2$$

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda R(f_\theta)$$

Regularizer

Standard loss

Regularization parameter

2. Modify architecture/training/data
   a) Dropout, batch normalization, augmentation

Break & Quiz

Why might a particular image transformation be unhelpful for data augmentation?

- A. A reflection can change the label of the image.
- B. The contrast of the image might not be changed enough.
- C. Cropping can remove too much of the original image.
- D. The cropped image can be too similar to the original image.
- E. All of the above.

Break & Quiz

Why might a particular image transformation be unhelpful for data augmentation?

- A. A reflection can change the label of the image.
- B. The contrast of the image might not be changed enough.
- C. Cropping can remove too much of the original image.
- D. The cropped image can be too similar to the original image.
- **E. All of the above.**

Thanks!