CS 540 Introduction to Artificial Intelligence

Deep Learning IV

University of Wisconsin-Madison
Fall 2025 Sections 1 & 2

Announcements

- **Homework:**
  - HW7 online, due on Monday **November 14th at 11:59 PM**
  - **It takes time to run experiments, please, start early!**

- Class roadmap and schedule:

Deep Learning IV and NN Review
Uninformed Search
Informed Search

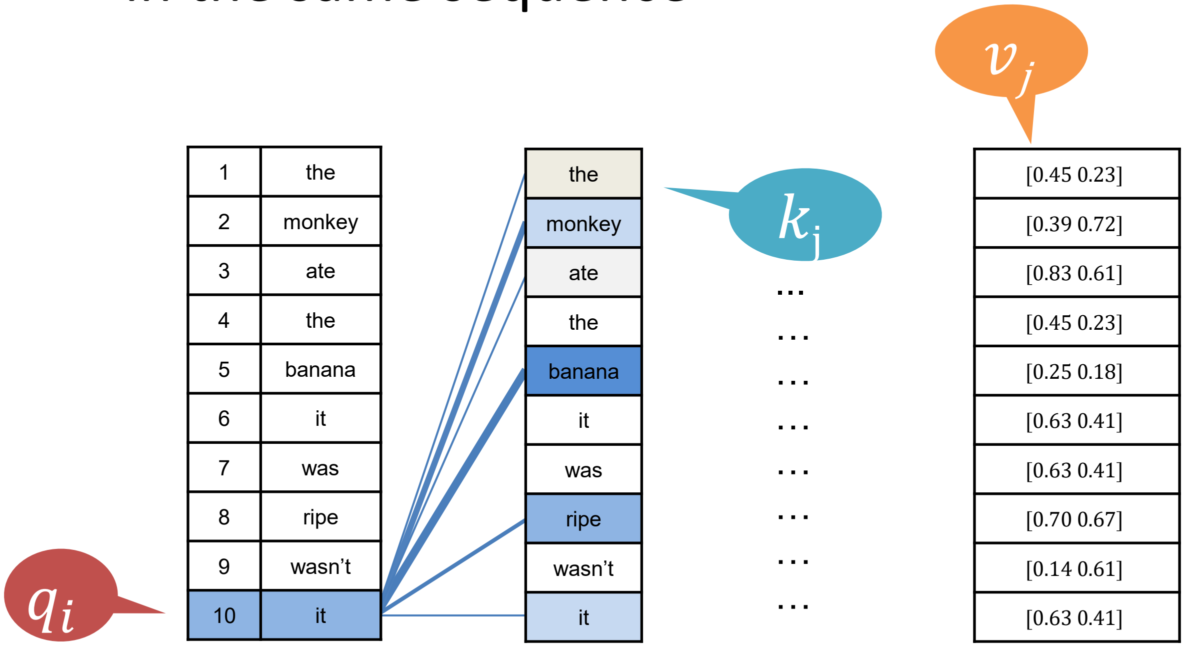
Outline

- Review Attention and Transformers
- Data Augmentation & Regularization
  - Expanding the dataset, avoiding overfitting
- More Signal From our Data
  - Graph-structured data, graph neural networks
- Neural Networks Review

The Attention Mechanism

The Attention Mechanism

Each token attends to all previous 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 $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$

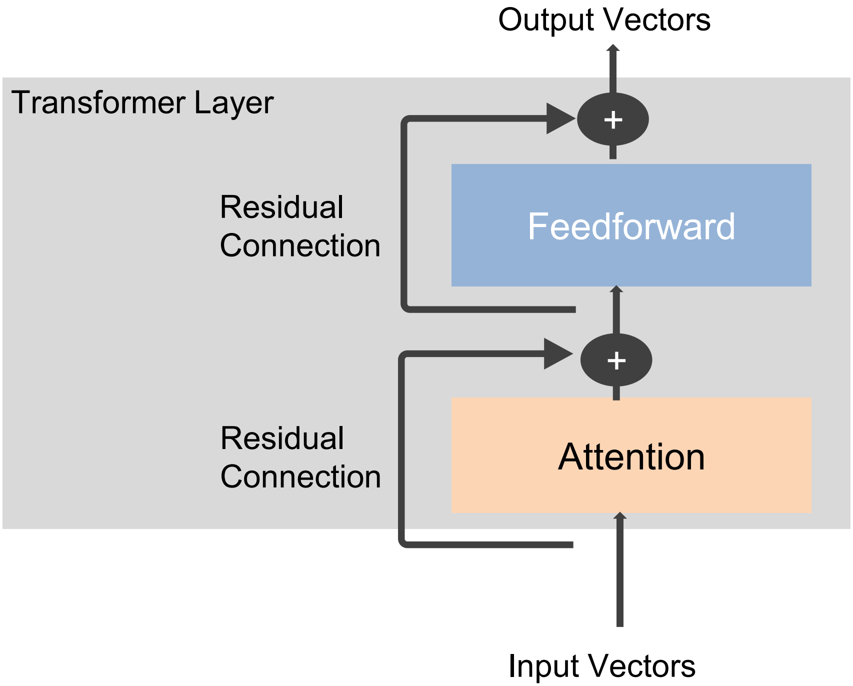
$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)\text{V}$$

The Transformer Architecture

From Attention to Transformer

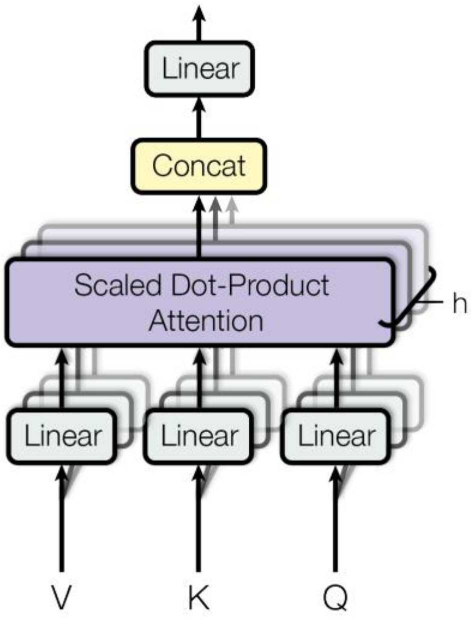
A single layer transformer consists of:

- Attention Mechanism
- Feed-Forward Network
- Residual Connections



Multi-Head Attention

- Outputs combined for richer representations
- Multiple heads learn different relationships (syntax, meaning, position)



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

Positional Encoding

- Transformers have no recurrence — order must be added explicitly
- **Positional Encoding**: Information about the relative or absolute position of the tokens in the sequence
- Added to the input embeddings

position dimension index

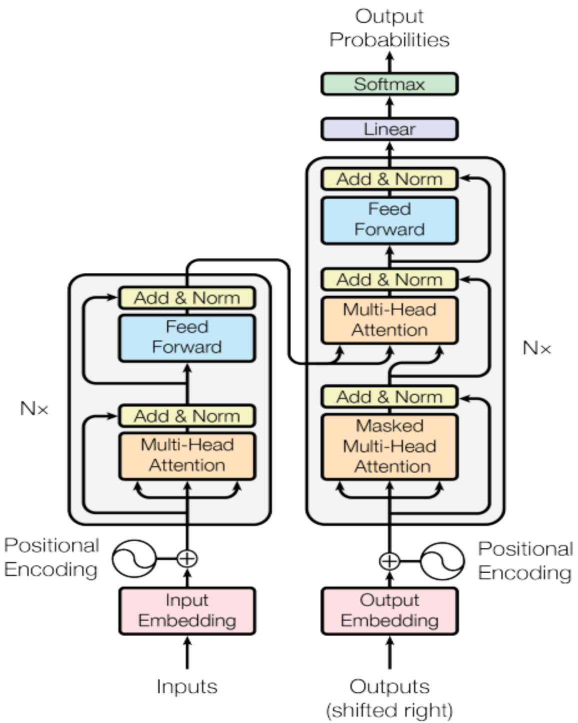
dimension

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Transformer Architecture

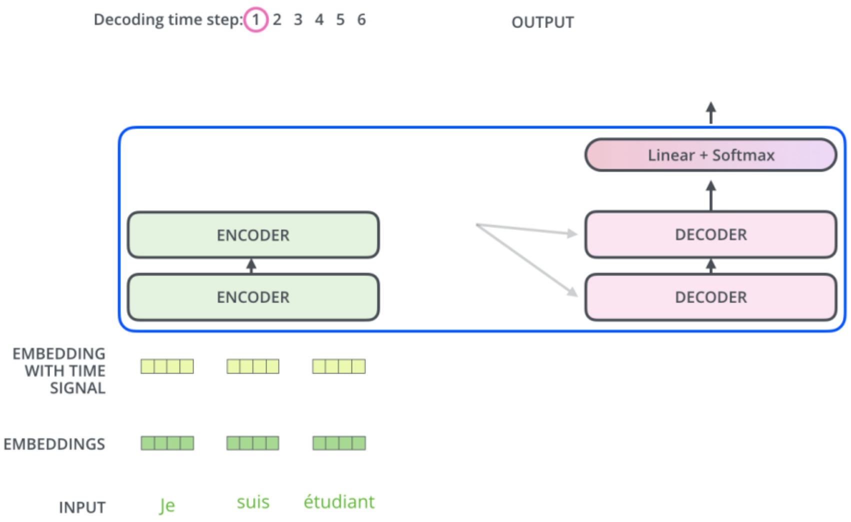
- Encoder–Decoder structure
- **Encoder**: maps an input sequence to a sequence of continuous representations z.
  - Useful for classification
- **Decoder**: Given z, the decoder generates an output sequence of symbols one element at a time.
  - Useful for generation

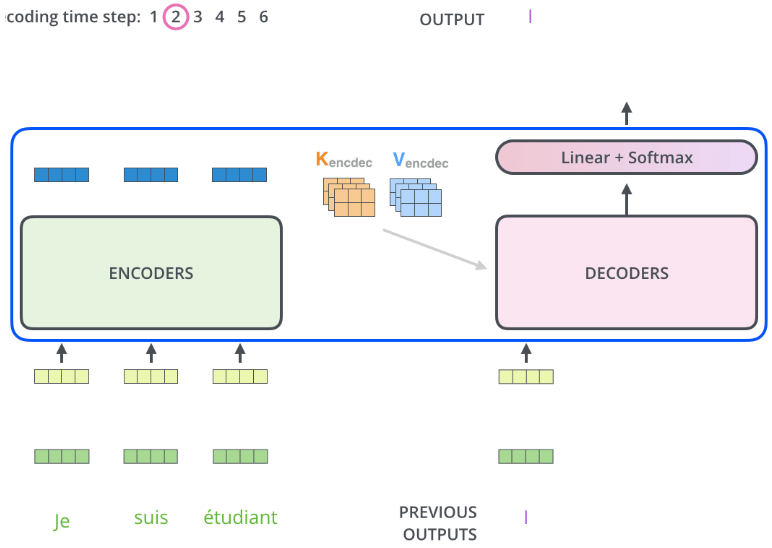


[Vaswani, A., et al. (2017). Attention Is All You Need](#)

Decoder

- **Masked multi-head attention**: each word attends to the words before it
- A **second attention** module that **attends the output of the encoder**





[The Illustrated Transformer]

Q3.1 Quiz Break

What is the primary function of the self-attention mechanism?

A) To track the position of each word in the sequence.
B) To process the input sequence strictly from left to right.
C) To weigh the importance and relationship of all words in a sequence relative to each other.
D) To reduce the size of the model by using fewer parameters.

Q3.1 Quiz Break

What is the primary function of the self-attention mechanism?

A) To track the position of each word in the sequence.
B) To process the input sequence strictly from left to right.
C) To weigh the importance and relationship of all words in a sequence relative to each other.
D) To reduce the size of the model by using fewer parameters.

Q3.2 Quiz Break

In the context of the Transformer model, what role do "Encoders" and "Decoders" play?

A) Encoders are used for text generation, and Decoders are used for text classification.
B) Encoders process the input sequence, and Decoders generate the output sequence.
C) Both Encoders and Decoders are only used for understanding the input sequence.
D) Both Encoders and Decoders map an input sequence to a sequence of continuous representations.

Q3.2 Quiz Break

In the context of the Transformer model, what role do "Encoders" and "Decoders" play?

A) Encoders are used for text generation, and Decoders are used for text classification.

B) Encoders process the input sequence, and Decoders generate the output sequence.

C) Both Encoders and Decoders are only used for understanding the input sequence.

D) Both Encoders and Decoders map an input sequence to a sequence of continuous representations.

Applications

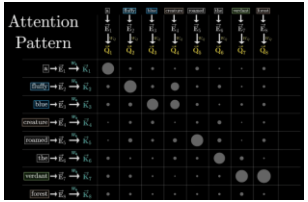
- Language Models: GPT, BERT, T5
- Vision: ViT (Vision Transformer)
- Multimodal: CLIP, DALL·E, GPT-4v
- Scientific: AlphaFold, time-series modeling, robotics

Further Reading/Viewing

- Jurafsky & Martin, Chapter 8
  - https://web.stanford.edu/~jurafsky/slp3/ed3book_aug25.pdf
- Russell & Norvig, Chapter 21.6 and 24

- Andrej Karpathy tutorial
  - https://karpathy.ai/zero-to-hero.html

- 3Blue1Brown:
  - https://www.youtube.com/watch?v=eMlx5fFNoYc



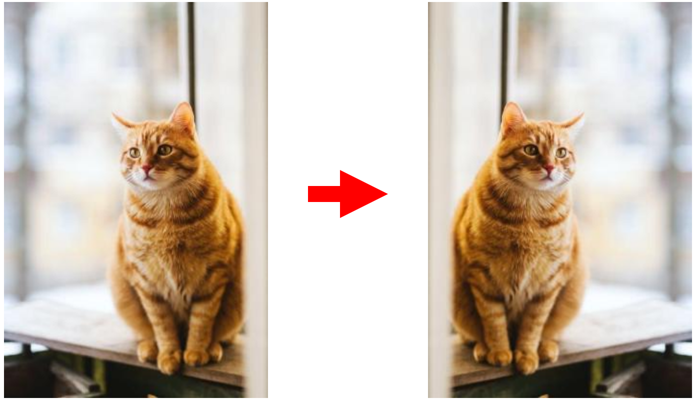
- The Illustrated Transformer
  - https://jalammar.github.io/illustrated-transformer/

Data Concerns

Data Concerns

What if we don't have a lot of data?

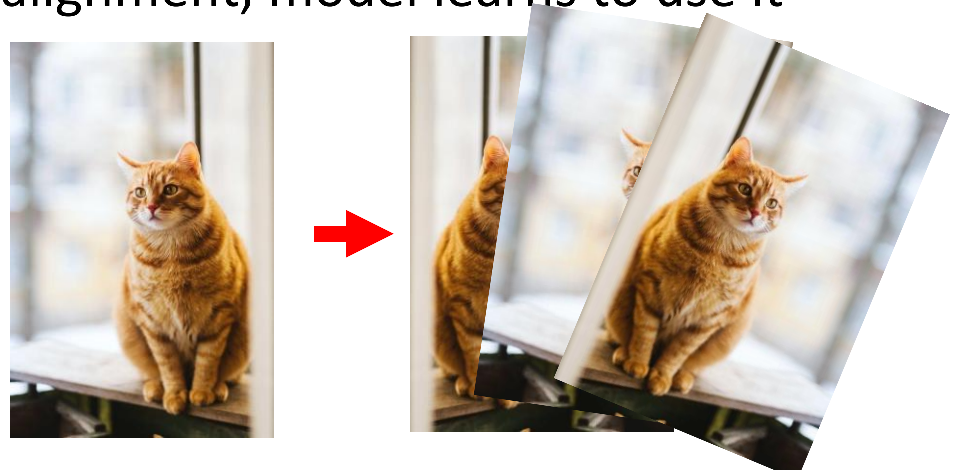
- We risk overfitting
- Avoiding overfitting: **regularization** methods
- Data augmentation: a classic way to regularize



Data Augmentation

Augmentation: transform + add new samples to dataset

- 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!



Transformations

Examples of transformations for images

- Crop (and zoom)
- Color (change contrast/brightness)
- Rotations+ (translate, stretch, shear, etc)

Many more possibilities. Combine as well!

Q: how to deal with this at **test time**?

- A: transform, test, average

Combining & Automating Transformations

One way to automate the process:

- Apply every transformation and combinations
- **Downside:** most don't help…

Want a good policy, ie, → → → → →

- Active area of research: search for good policies

1. **Ratner et al**: "Learning to Compose Domain-Specific Transformations for Data Augmentation"
2. **Cubuk et al**: "AutoAugment: Learning Augmentation Strategies from Data"

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”

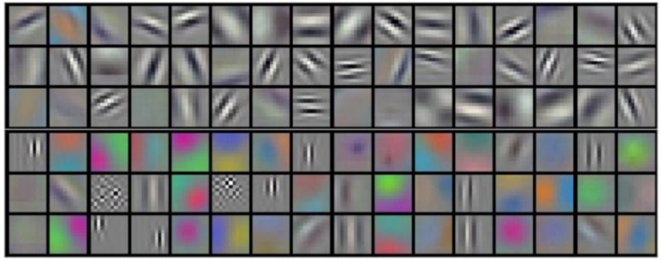
Xie **et al**: “Unsupervised Data Augmentation for Consistency Training”

Importance of Augmentation

Data augmentation is critical for top performance!

- You should use it!
- **AlexNet**: used (many papers re-used as well)
  - Random crops, rotations, flips.

Krizhevsky et al: “ImageNet Classification with Deep Convolutional Neural Networks”



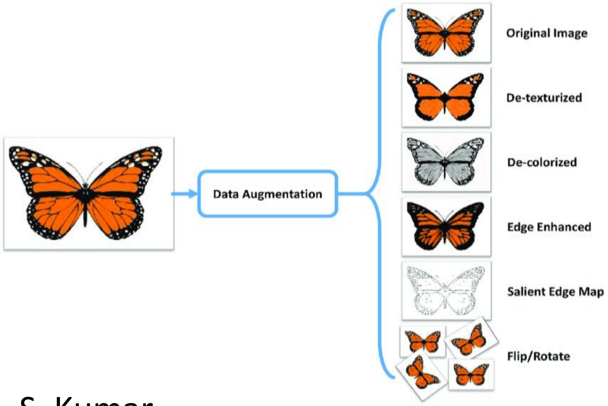
Other Forms of Regularization

Regularization has many interpretations

- **Goodfellow**: "any modification… to a learning algorithm that is intended to reduce its generalization error but not its training error."

- A way of adding knowledge / side information to model

- Enforcing parsimony/simplicity



S. Kumar

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

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

Break & Quiz

Q 1.1: If we apply data augmentation blindly, we might

(i) Change the label of the data point

(ii) Produce a useless training point

- A. (i) but not (ii)
- B. (ii) but not (i)
- C. Neither
- D. Both

Break & Quiz

Q 1.1: If we apply data augmentation blindly, we might

(i) Change the label of the data point

(ii) Produce a useless training point

- A. (i) but not (ii)
- B. (ii) but not (i)
- C. Neither
- **D. Both**

Break & Quiz

Q 1.1: If we apply data augmentation blindly, we might

(i) Change the label of the data point

(ii) Produce a useless training point

- A. (i) but not (ii) (Can do (ii): imagine turning up the contrast till the image is completely black and is unusable).
- B. (ii) but not (i) (Can change label: rotate a 6 into a 9).
- C. Neither (Can do either).
- **D. Both**

Break & Quiz

Q 1.2: What are some consequences of data augmentation?

(i) We have to store a much bigger dataset in memory

(ii) For a fixed batch size, there will be more batches per epoch

- A. (i) but not (ii)
- B. (ii) but not (i)
- C. Neither
- D. Both

Break & Quiz

Q 1.2: What are some consequences of data augmentation?

(i) We have to store a much bigger dataset in memory

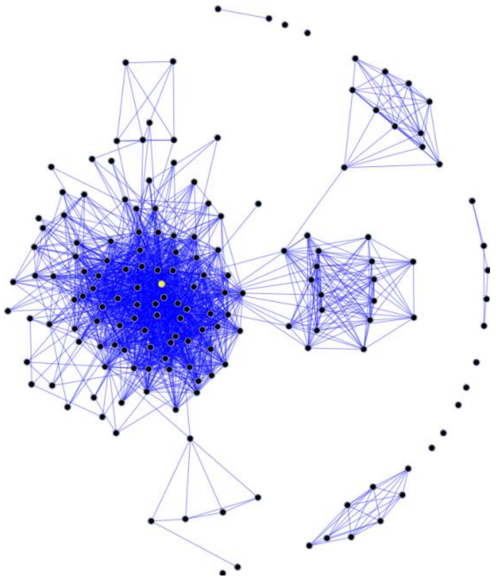
(ii) For a fixed batch size, there will be more batches per epoch

- A. (i) but not (ii)
- B. (ii) but not (i)
- C. Neither
- **D. Both**

Graph Neural Networks

Relationships in Data

- We usually assume all data points are independent, "unrelated" in a sense
- Pretty common to have relationships between points $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
  - **Social networks**: individuals related by friendship
  - **Biology/chemistry**: bonds between compounds, molecules
  - **Citation networks**: Scientific papers cite each other

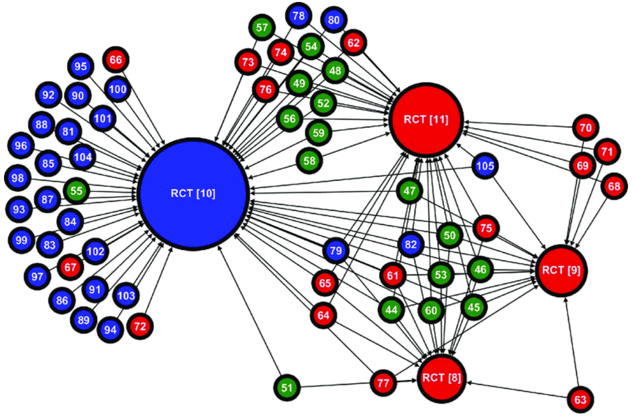


Wiki

Signal from Relationships

Suppose we are classifying scientific papers

- **Features**: title, abstract, authors. **Labels**: math/science/eng.
- Could build a reasonable classifier with the above data
- **More signal** from relationships
  - Cite each other, more likely from the same field
  - Note: citations are not features; they're **links**
  - Need a new type of network to handle



Leng

Graph Neural Networks

Have: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n), G = (V, E)$

How should our new architecture look?

- Still want layers
  - linear transformation + non-linearity
- Now want to integrate neighbors
- Bottom: graph convolutional network

Parameters

Non-Linearity

Hidden Layer Representation

$$H^{(\ell+1)} = \sigma(H^{(\ell)} W^{(\ell)})$$

$$H^{(\ell+1)} = \sigma(A_G H^{(\ell)} W^{(\ell)})$$

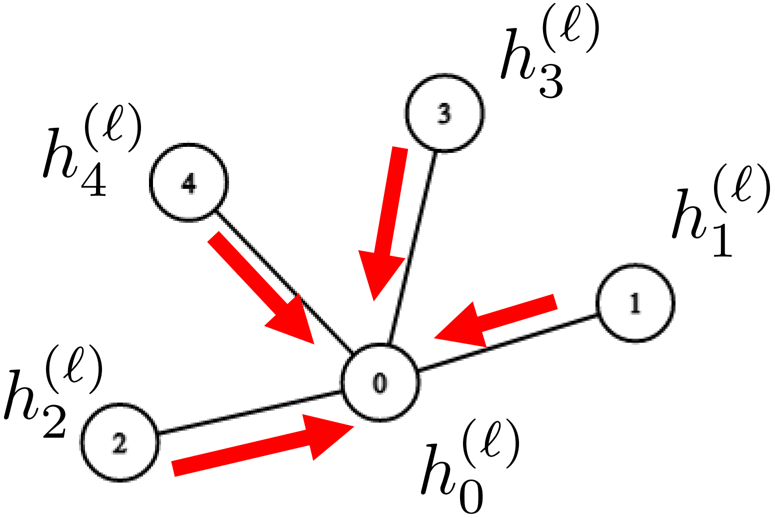
Graph Mixing

Kipf and Welling: "Semi-Supervised Classification with Graph Convolutional Networks"

Graph Convolutional Networks

Let's examine the GCN architecture in more detail

- Difference: "graph mixing" component
- At each layer, get representation at each node
- Combine node's representation with neighboring nodes



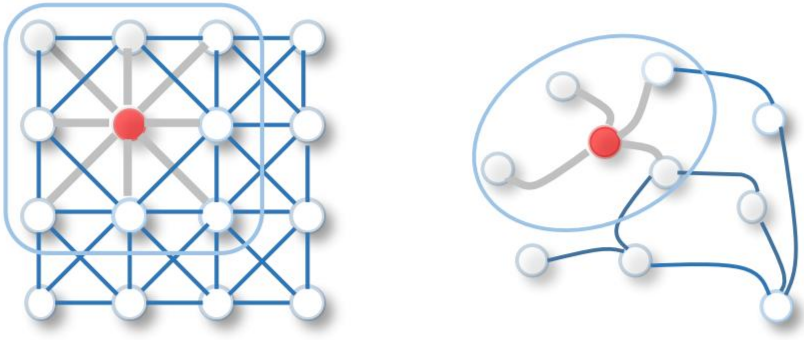
- "**Aggregate**" and "**Update**" rules

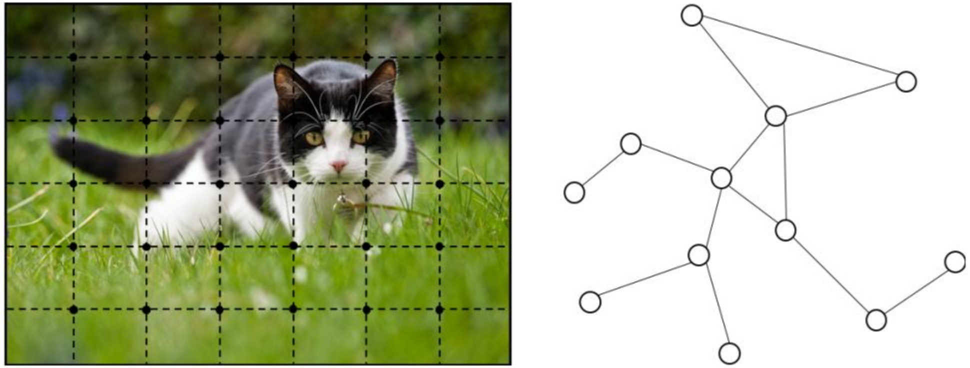
Graph Convolutional Networks

Note the resemblance to CNNs:

- Pixels: arranged as a very regular graph
- Want: more general configurations (less regular)

Wu et al, A Comprehensive Survey on Graph Neural Networks





Zhou et al, Graph Neural Networks: A Review of Methods and Applications

Neural Networks Review

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

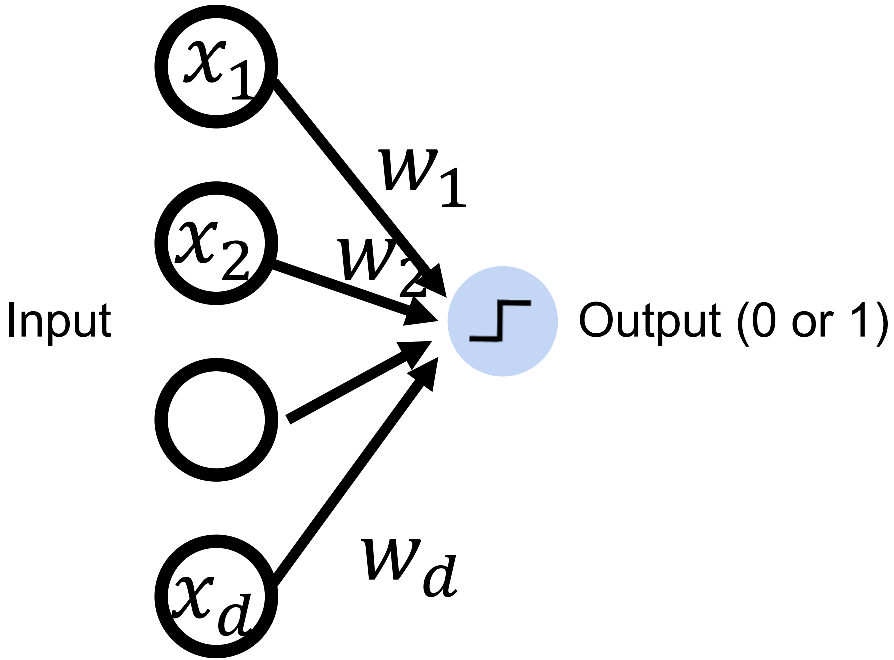
Perceptron

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

$$o = \sigma(\mathbf{w}^\top \mathbf{x} + b) \qquad \sigma(x) = \begin{cases} 1 & \mathit{if}\ x > 0 \\ 0 & \mathit{otherwise} \end{cases}$$

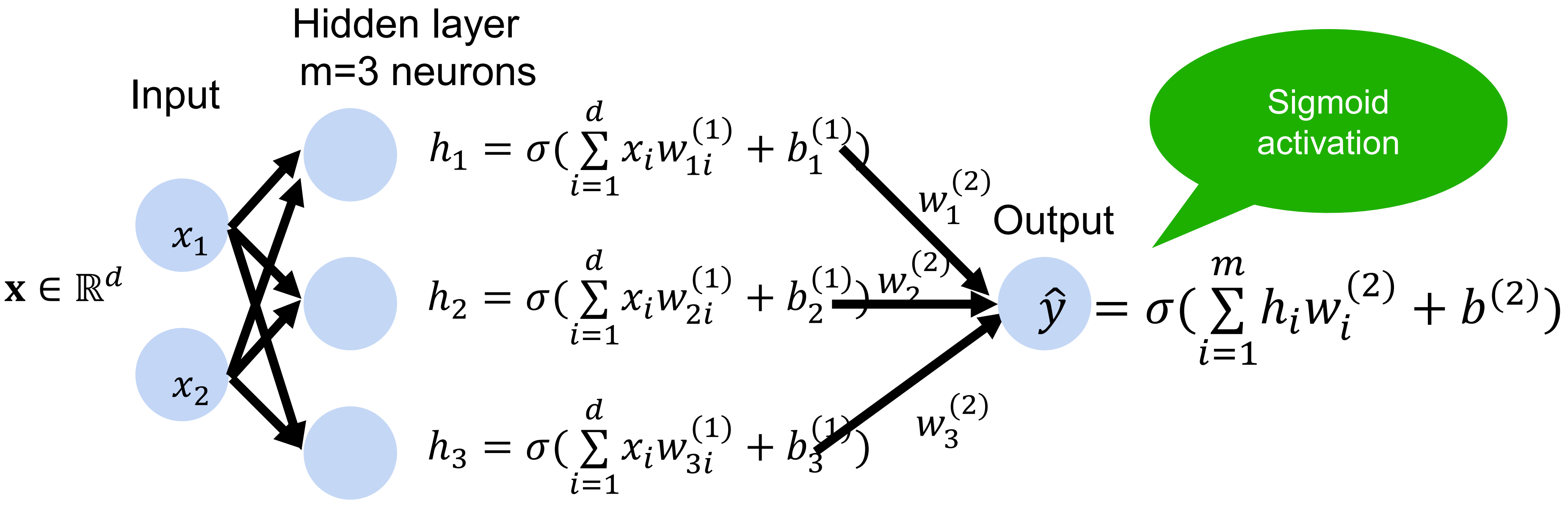
Activation function

Cats vs. dogs?



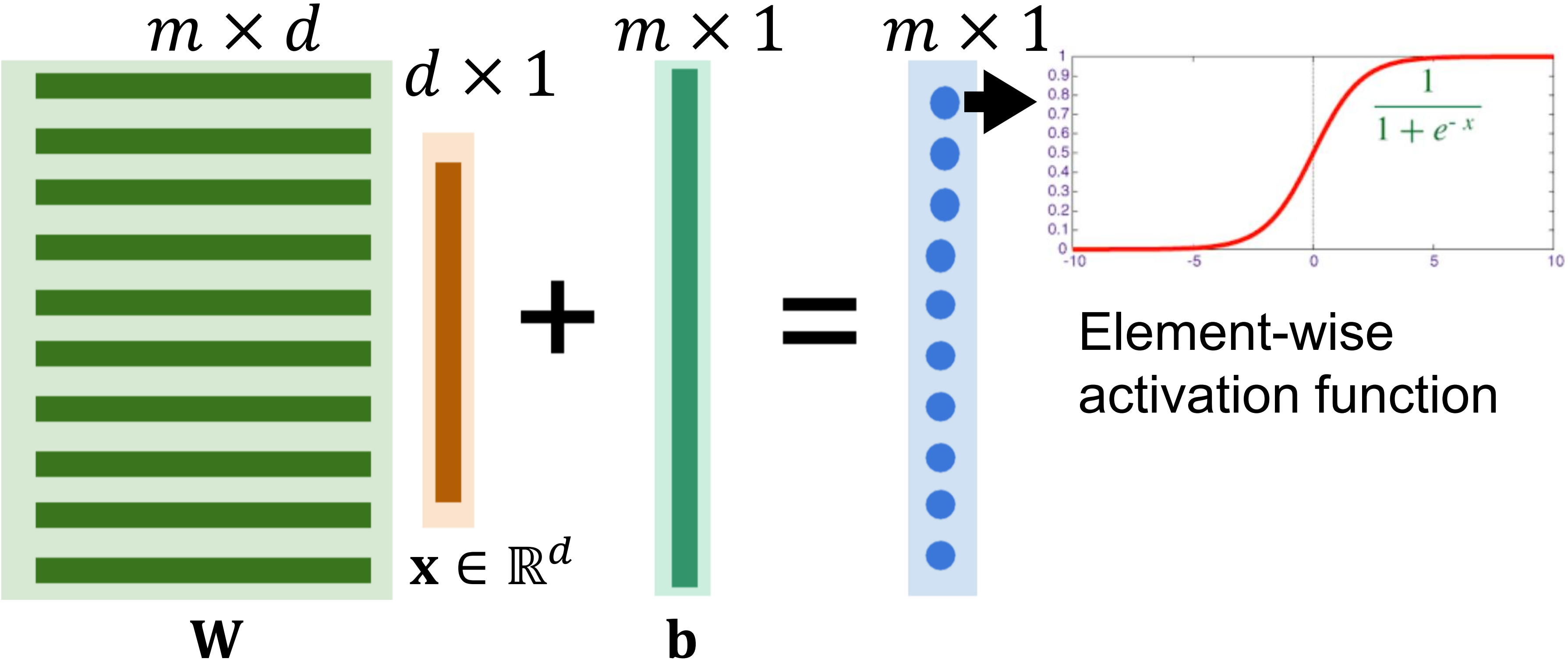
Multi-layer perceptron: Example

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



Neural networks with one hidden layer

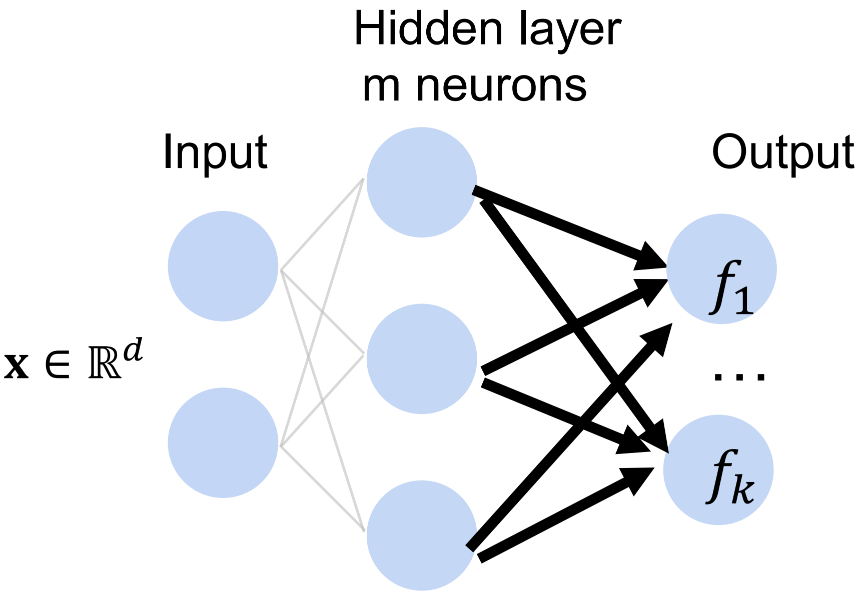
Key elements: linear operations + Nonlinear activations



Element-wise activation function

Multi-class classification

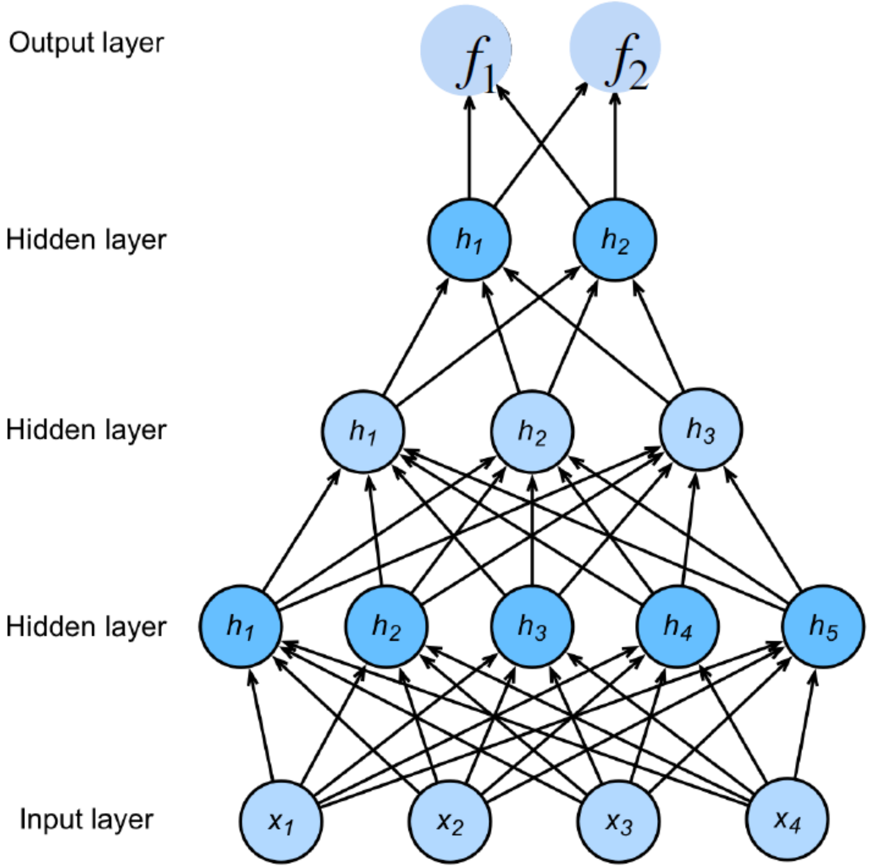
Turns outputs *f* into *k* probabilities (sum up to 1 across *k* classes)



$$p(y|\mathbf{x}) = softmax(\mathbf{f})$$

$$= \frac{\exp f_y(x)}{\sum_i^k \exp f_i(x)}$$

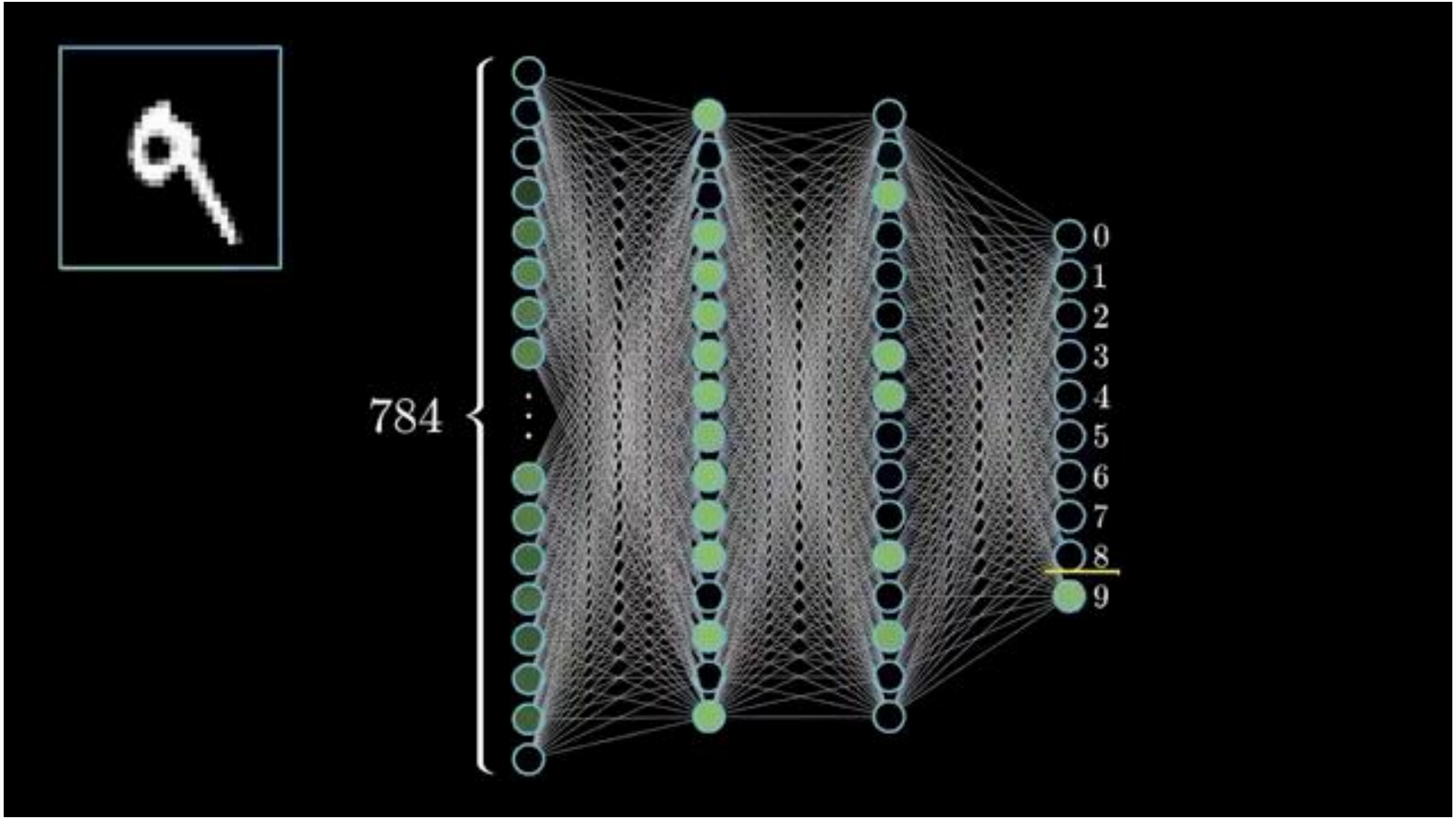
Deep neural networks (DNNs)



$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$
$$\mathbf{f} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$
$$\mathbf{y} = \mathrm{softmax}(\mathbf{f})$$

NNs are composition of nonlinear functions

Classify MNIST handwritten digits



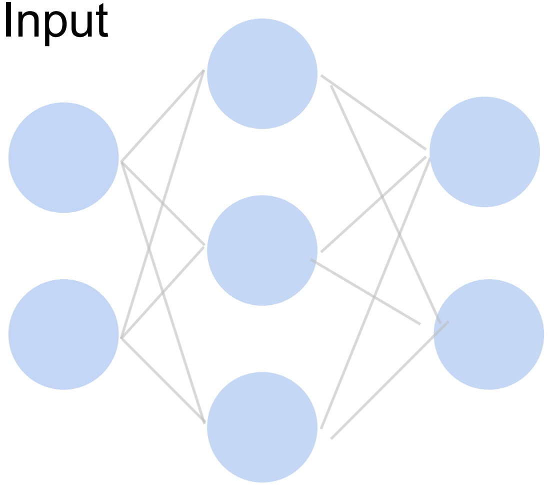
How to train a neural network?

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

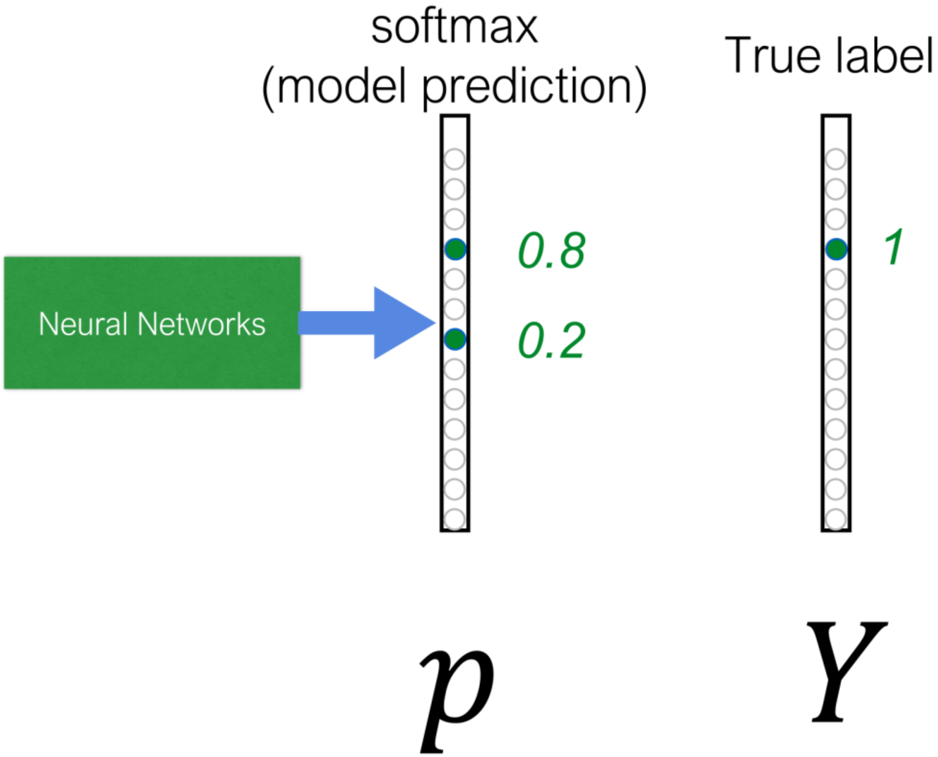
Per-sample loss:

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

Also known as cross-entropy loss or softmax loss



Cross-Entropy Loss



$$L_{CE} = \sum_j - y_j \log(p_j)$$

$$= -\log(0.8)$$

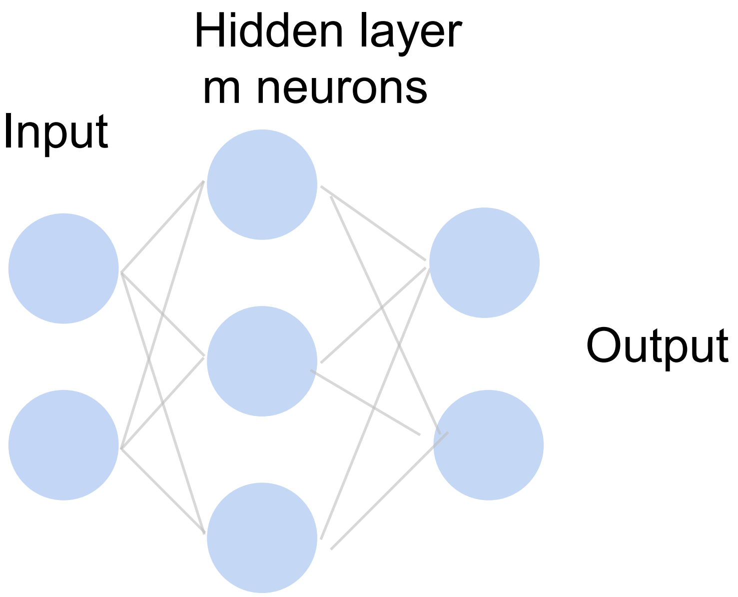
Goal: push ***p*** and ***Y*** to be identical

How to train a neural network?

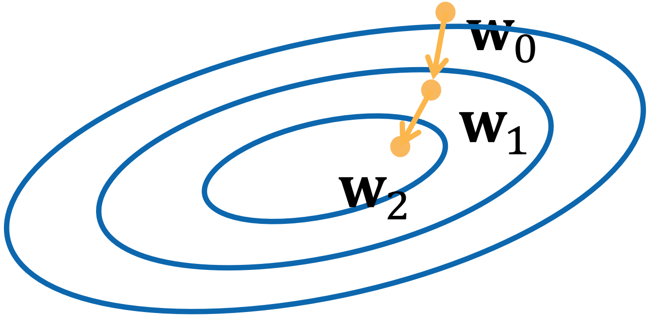
Update the weights W to minimize the loss function

$$L = \frac{1}{|D|} \sum_i \ell(\mathbf{x}_i, y_i)$$

Use gradient descent!



Gradient Descent



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

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

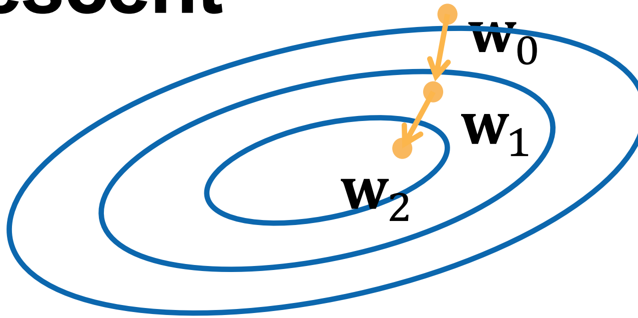
D can be very large. Expensive per iteration

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

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

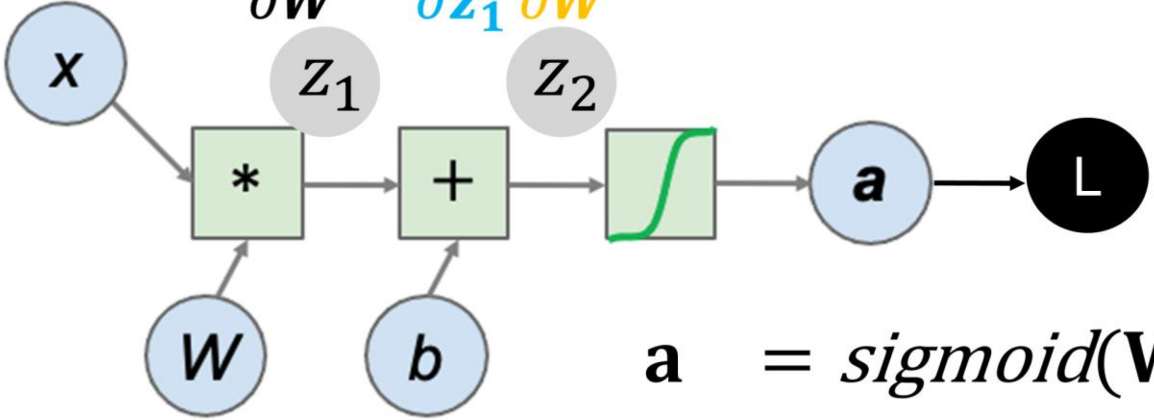
Calculate gradient: backpropagation with chain rule

- 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

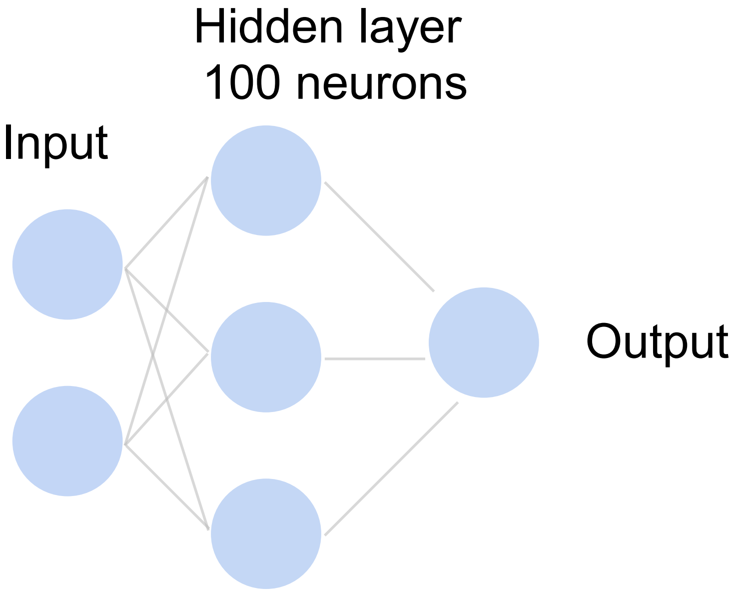
$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{z_1}} \frac{\partial \boldsymbol{z_1}}{\partial \boldsymbol{W}}$$



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

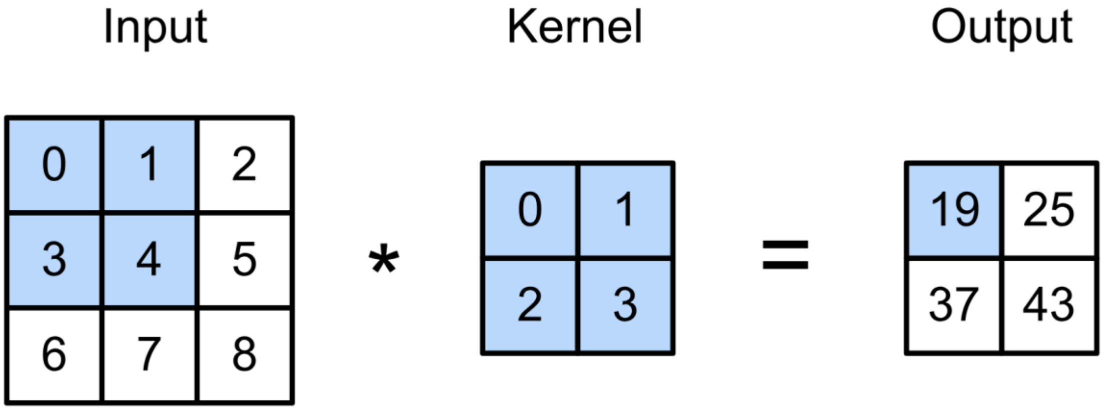
Fully Connected Networks

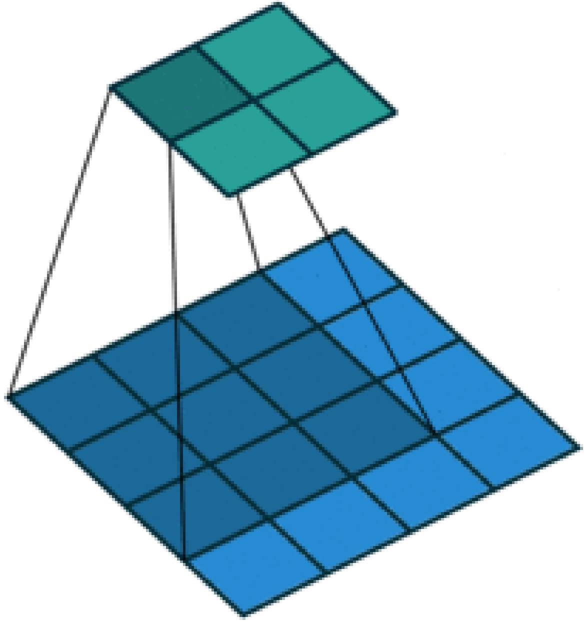
Cats vs. dogs?



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

2-D Convolution

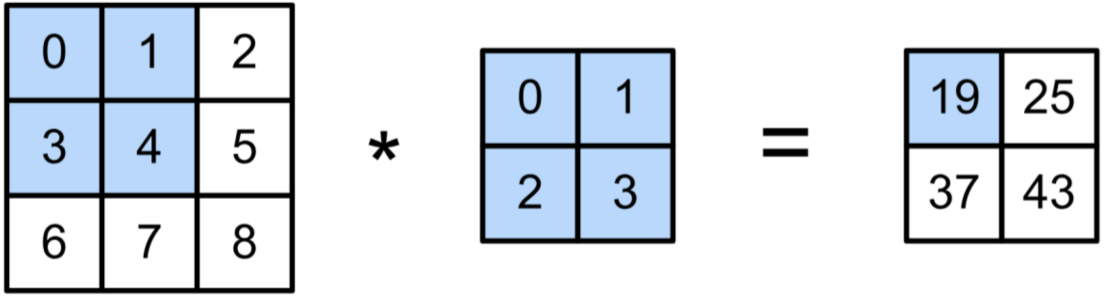




(vdumoulin@ Github)

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$

2-D Convolution Layer



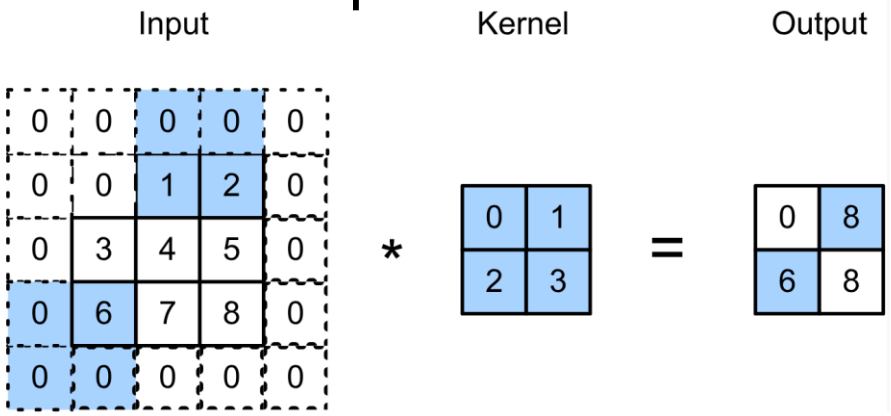
- $\mathbf{X}$: $n_h \times n_w$ input matrix
- $\mathbf{W}$: $k_h \times k_w$ kernel matrix
- b: scalar bias
- $\mathbf{Y}$: $(n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + b$$

- **W** and *b* are learnable parameters

2-D Convolution Layer with Stride and Padding

- Stride is the #rows/#columns per slide
- Padding adds rows/columns around input
- Output shape



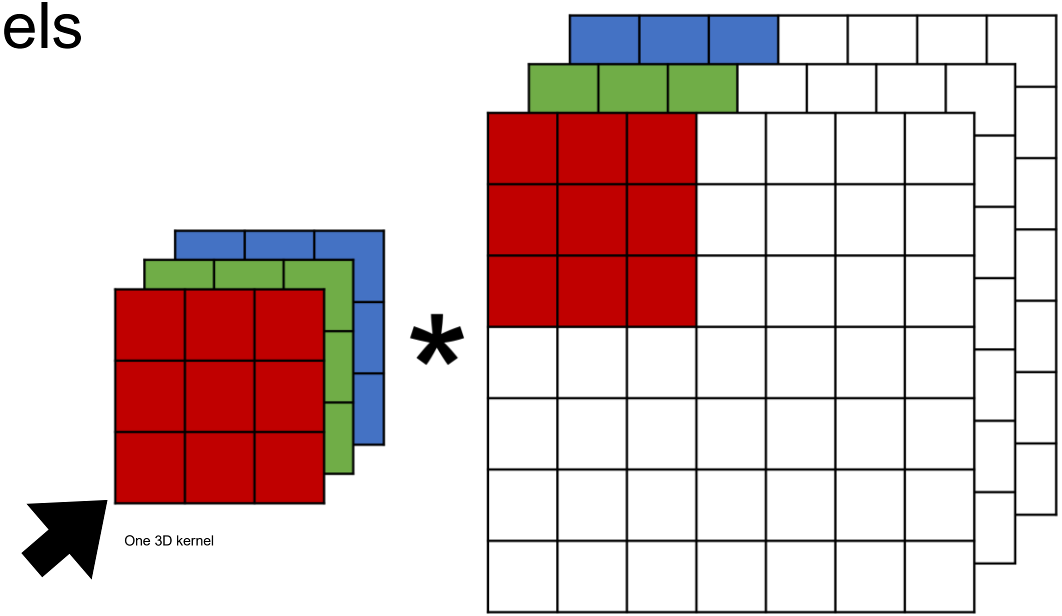
Kernel/filter size

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

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

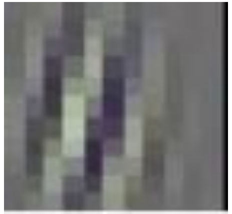
Multiple Input Channels

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a 2D kernel for each channel, and then sum results over 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)

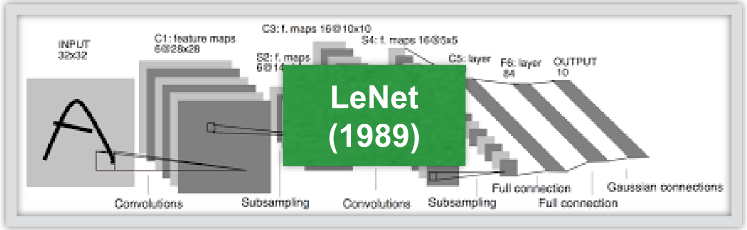
Multiple Output Channels

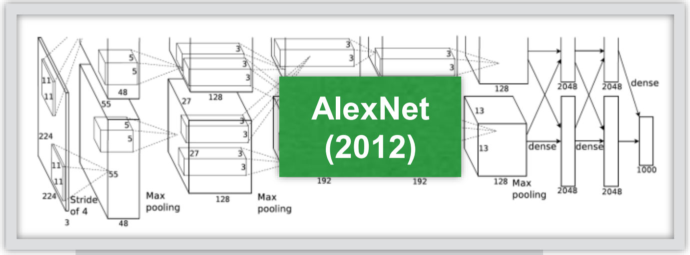
- The # of output channels = # of filters
- Input $\mathbf{X}: c_i \times n_h \times n_w$
- Kernel $\mathbf{W}: c_o \times c_i \times k_h \times k_w$
- Output $\mathbf{Y}: c_o \times m_h \times m_w$

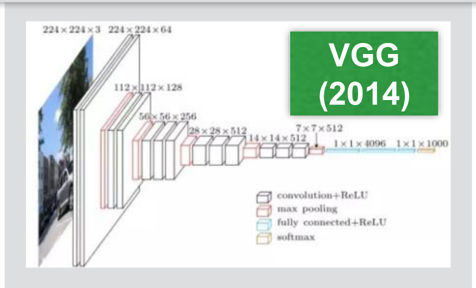
$$\mathbf{Y}_{i,:,:} = \mathbf{X} \star \mathbf{W}_{i,:,:,:}$$

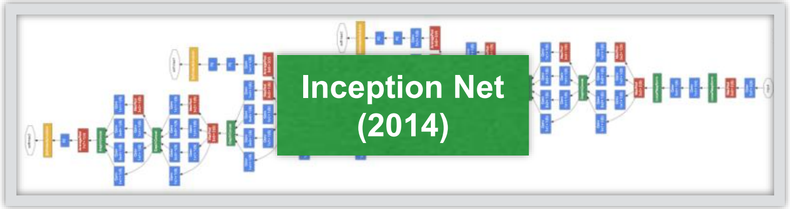
$$for\, i = 1, \ldots, c_o$$

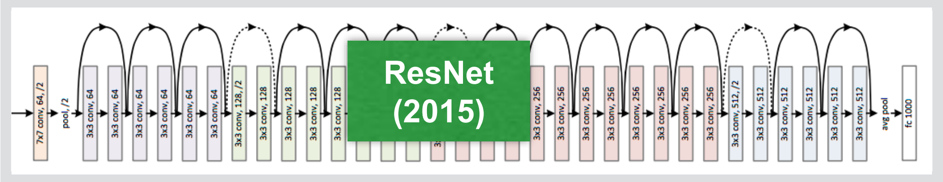
Evolution of neural net architectures

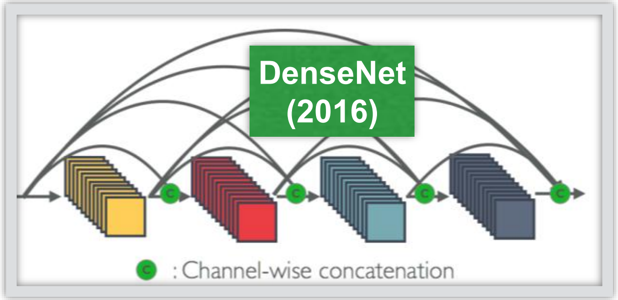












Other Deep Architectures

Other common architectures:

- **Recurrent neural networks:** hidden activations are a function of input and activations from previous inputs. Designed for sequential data such as text.
- **Graph neural networks:** take graph data as input.
- **Transformers:** take sequences as input and learn what parts of input to pay attention to.

Quiz break

Which one of the following is NOT true about perceptron?

A. Perceptron only works if the data is linearly separable.

B. Perceptron can learn AND function

C. Perceptron can learn XOR function

D. Perceptron is a supervised learning algorithm

Quiz break

Which one of the following is NOT true about perceptron?

A. Perceptron only works if the data is linearly separable.

B. Perceptron can learn AND function

C. Perceptron can learn XOR function

D. Perceptron is a supervised learning algorithm

Quiz break

Which one of the following is NOT true?

A. LeNet has two convolutional layers

B. The first convolutional layer in LeNet has 5x5x6x3 parameters, in case of RGB input

C. Pooling is performed right after convolution

D. Pooling layer does not have learnable parameters

Quiz break

Which one of the following is NOT true?

A. LeNet has two convolutional layers

B. The first convolutional layer in LeNet has 5x5x6x3 parameters, in case of RGB input

C. Pooling is performed right after convolution

D. Pooling layer does not have learnable parameters

Pooling is performed after ReLU: conv -> relu -> pooling

Acknowledgements: Inspired by materials by Fei-Fei Li, Ranjay Krishna, Danfei Xu (Stanford CS231n)