Lecture 22 – 19/11/24

Weight initialization

- Can we initialize all weights to an identical value?
  - No, because all nodes within the same hidden layer will behave identically
  - this is known as symmetry problem.

Gradient vanishing

- In a deep neural network:
  - Vanishing → gradients become very small : W's barely update.
  - Exploding → gradients become very large : W's updating causes instability.
  - Releva elements of the problem : activation function, weights initialization
- Vanishing : mostly related to activation
  - → i.e. Sigmoid:

    For its activation there is a large difference between variance (range) of input, output.

    This means that output gets quickly close to 0 or 1 where derivatives are near zero

  Many layers - problem

- Exploding : mostly related to large weights.
  - Leads to instability.

- How to initialize weights to reduce vanishing & exploding?
  - Good technique → keep equal variance of activations across layers.
  - There are some heuristics:

    Glorot : used for sigmoid & tanh

    He : used for ReLU

Pro & Cons of various activation

- Sigmoid

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

  Toward either end of sigmoid, output changes slower. (gradient vanishing)

  Computationally expenses.

- Tanh

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

  Bigger range, derivatives are steeper.

  Suffers from vanishing gradient.

  Computationally expensive due to exponential.

- ReLU

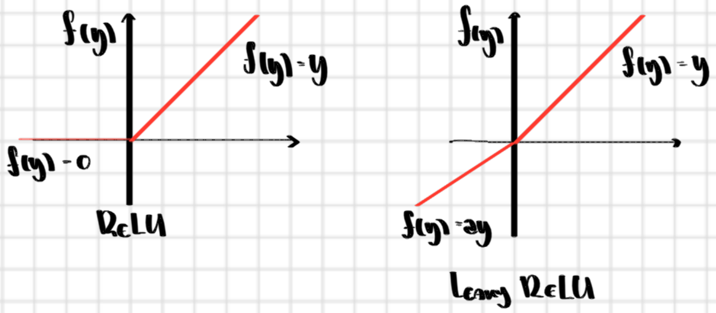
  $$f(x) = \max(0,x)$$

  Computationally efficient

  Dying ReLU problem might happen → negative input implies gradient = 0

Leaky ReLU

- It is an attempt to solve the dying ReLU problem
- Leak increases range output of ReLU $(-\infty, \infty)$
- f and its derivatives are monotonic

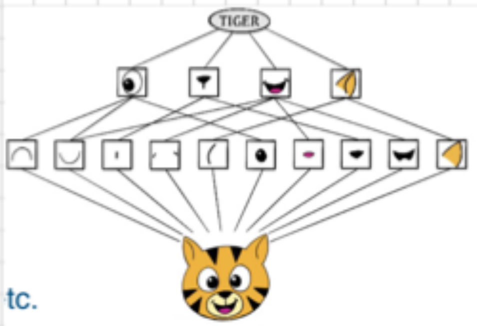


Convolutional Neural Network

- With image flattening we loose spacial dependencies.
- When using fully connected NN for an image:
  - Lose spatial dependencies
  - Large # of parameters:
    - Computer needs a lot of memory!
    - Too many params lead to overfitting!

IDEA

- Check for certain features in different image patches
- Create a hierarchy of features.
- Handle variations of the same object.



Filters & Convolution Operation

- A filter is a matrix that is much smaller than the image.
- Filters are also known as "kernel".
- Convolutional sum is a linear operation.
- Slide the filter over input matrix:
  - Execute convolution
  - multiply overlapping values & sum them up → Gives back final pixel

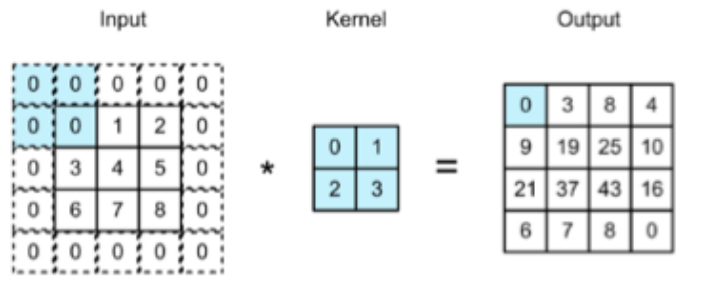
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1\cdot1 \; 1\cdot0 \; 1\cdot1 & " \\ 0\cdot0 \; 1\cdot1 \; 1\cdot0 & " \\ 0\cdot1 \; 0\cdot0 \; 1\cdot1 & \\ " & " \end{bmatrix}$$

results in

$$\begin{bmatrix} 4 & & \\ & & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 1+0+1+0+1+0+0+0+1 = \boxed{4}$$

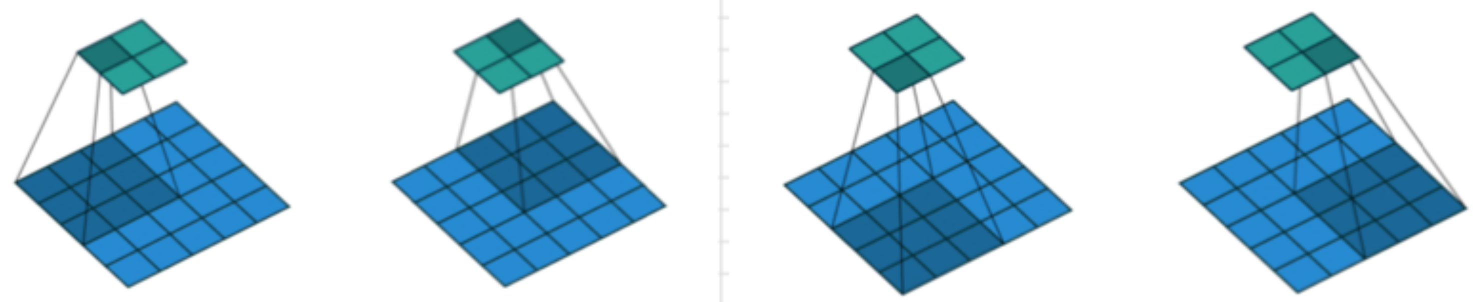
PADDING

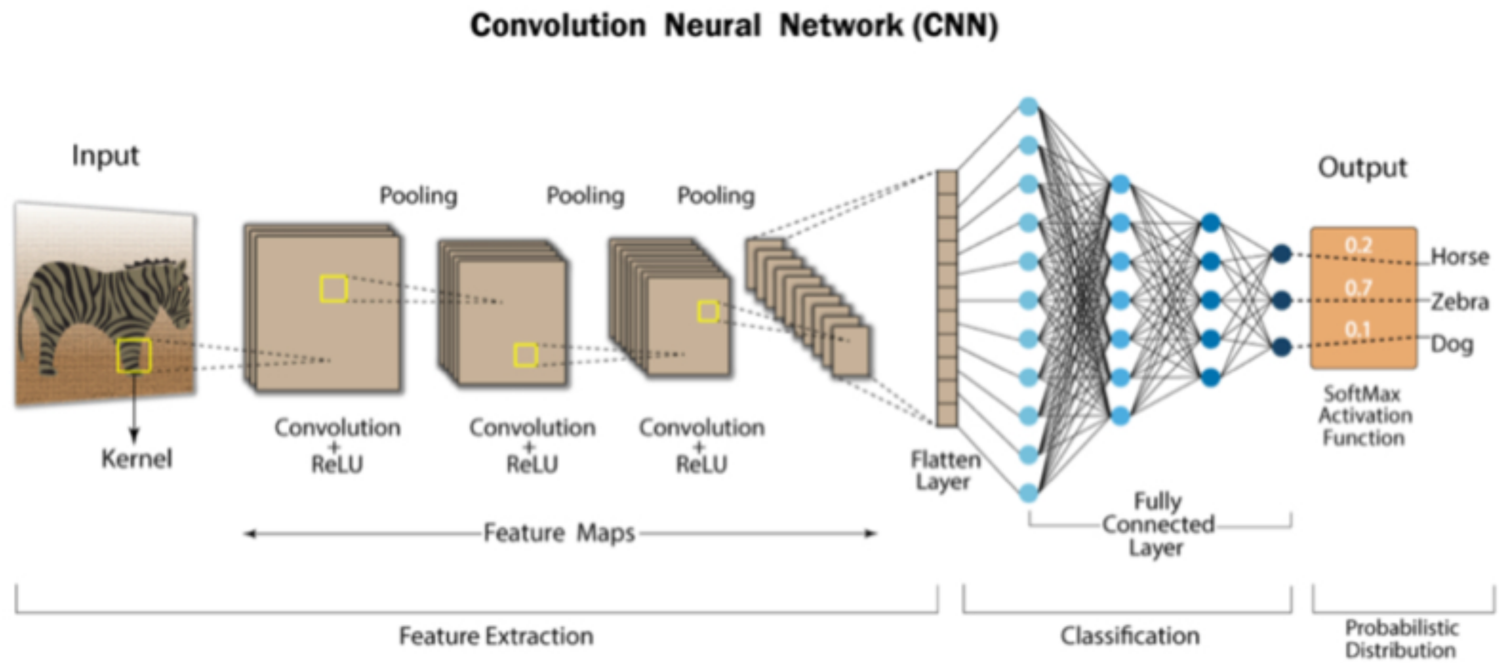


- Output image is smaller than input image
  - ? How to apply filters to borders?
  - ? What if we want the same size?
- Solution:
  - Add extra pixels on rim of original, depending on kernel's size

STRIDE

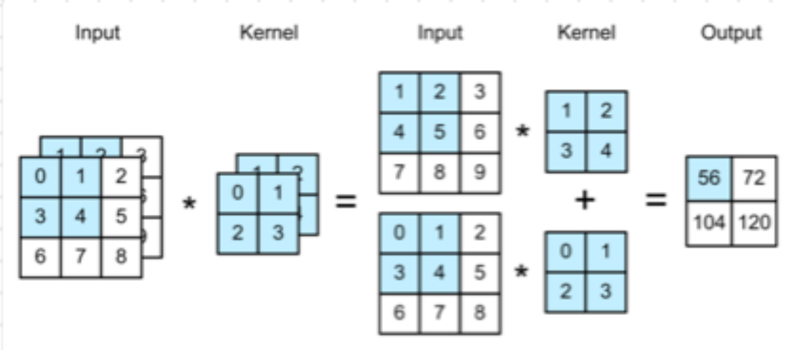
- We start with the convolution window in upper left of the input tensor
- Then slide it over all locations both down & to the right.
- Stride length:
  - # of steps we take when sliding our filter across an image.
- It is a downsampling method that skips pixels when moving the filter across image





MULTIPLE INPUT CHANNELS

- We often have multiple input channels:
  - Images may come with 3 channels for RGB
- We implement multi-channel filters to deal with multi-channel inputs.



- Would like to apply multiple different filters over the image
- Different filters detect different important information

POOLING

- Downsampling a feature map & reducing its size → can emphasis dominant features.
- max-pooling normally better than average-pooling
- normal to insert pooling layer betwees successive convolutional

Extract features

- Extraction from low-level to high-level
  - Flatten last layer's output.
  - Apply fully connected layers
- By adding fully connected layer, networks learns non-linear comb.
- In CNN:
  - convolution can be seen as a neural network layer where weights are shared.