Fall 2021 | Lecture 18

Neural Networks

Ariel Procaccia | Harvard University

DEEP LEARNING MILESTONES

2011

AlexNet

Convolutional net wins image classification competitions

2012

Cat Experiment

Google NN learns to identify cats from 10M unlabeled images

2014

DeepFace

Facebook NN learns to identify faces with 97% accuracy

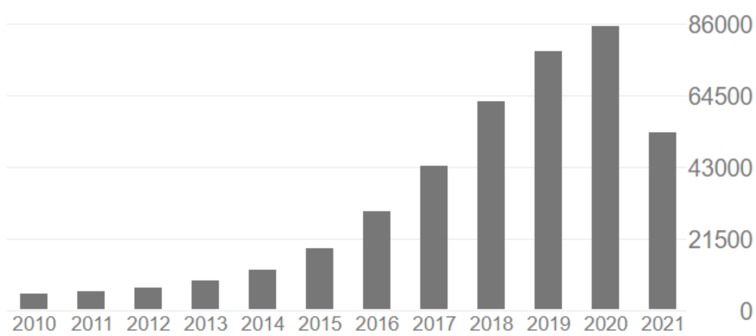
2020

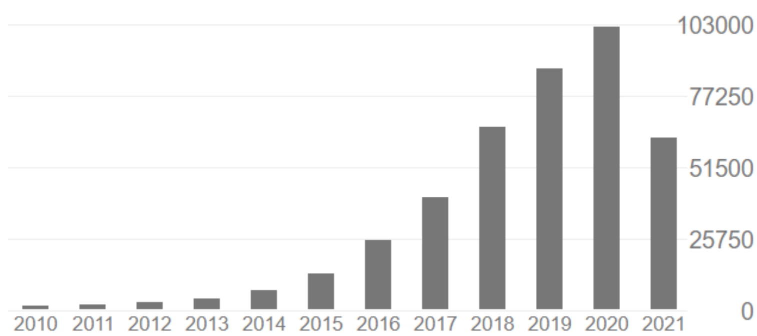
GPT-3

OpenAI’s language model produces human-like text

THE DEEP LEARNING REVOLUTION

… through the lens of Google Scholar

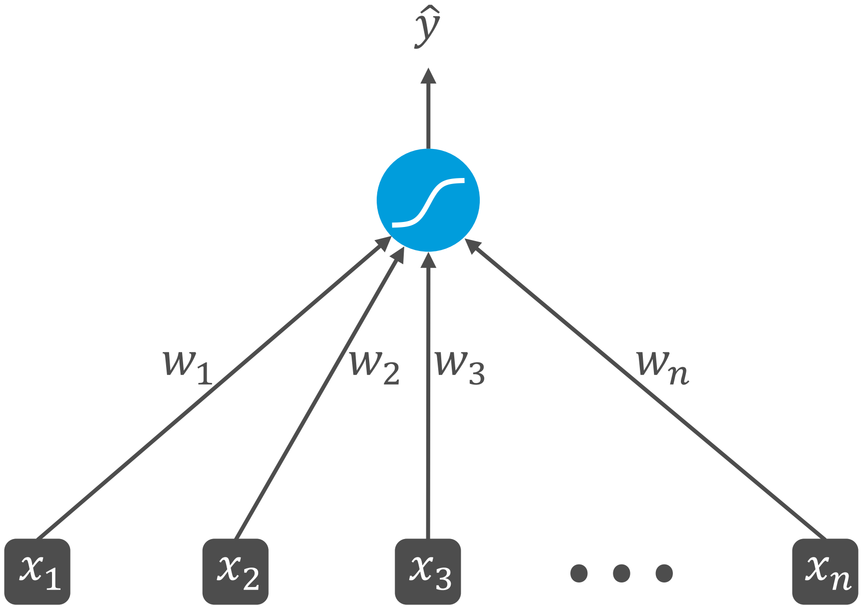




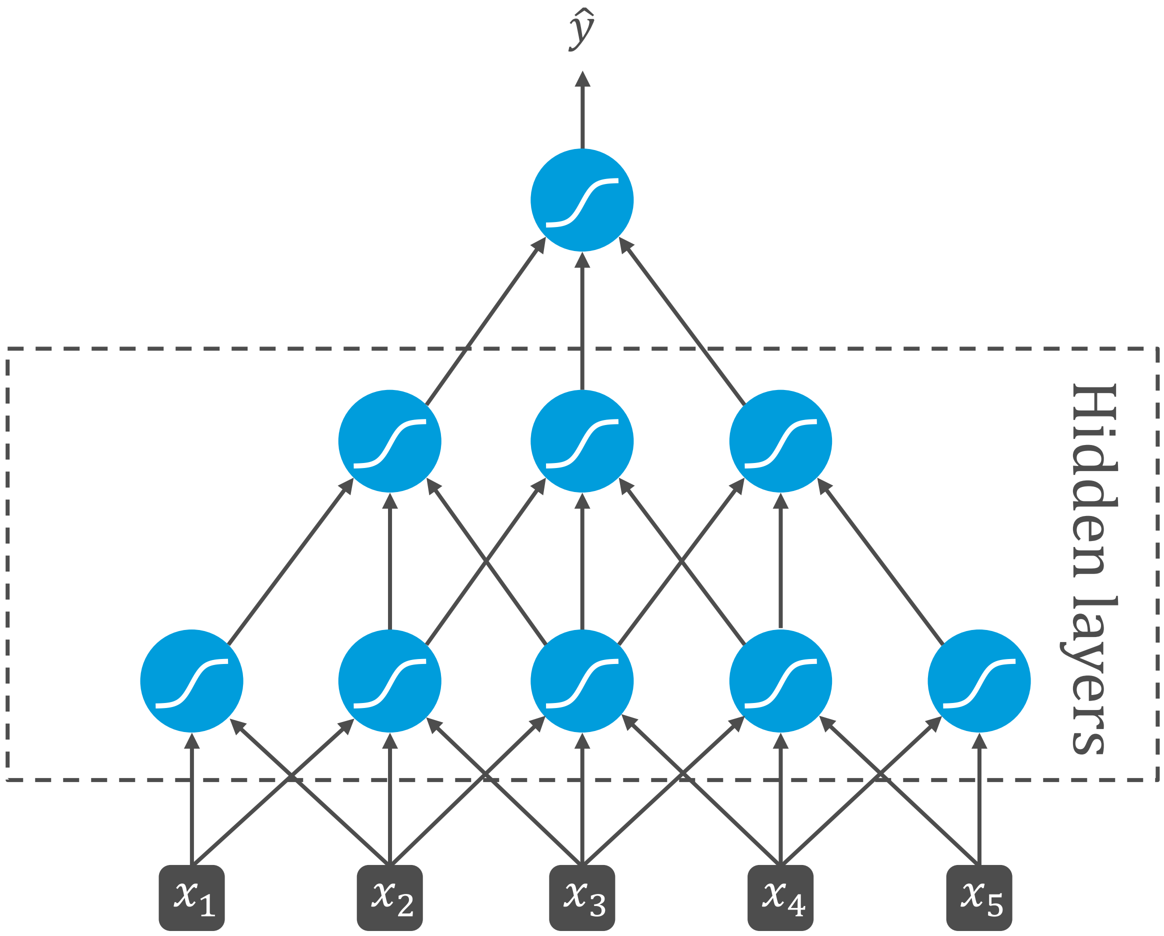
Geoff Hinton
University of Toronto and Google

Yoshua Bengio
University of Montreal

LOGISTIC REGRESSION, REVISITED



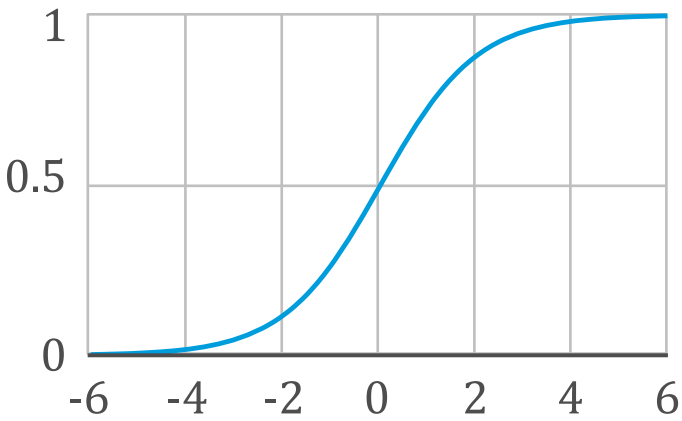
DEEP(ER) NEURAL NETWORK



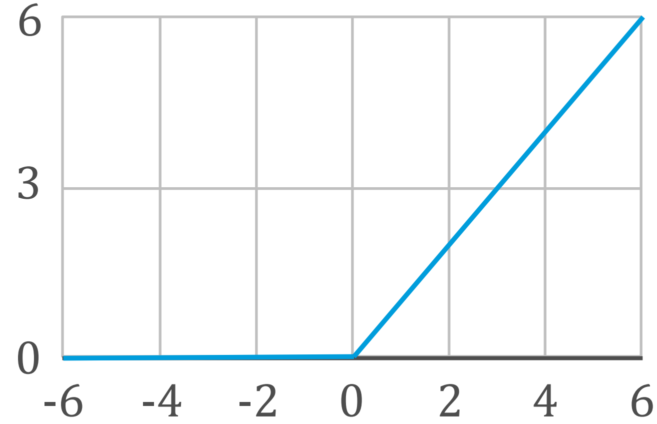
ACTIVATION FUNCTIONS

- For now we will focus on feed-forward networks, which are acyclic
- Each node is called a unit
- A unit calculates the weighted sum of its predecessors and applies an activation function to it
- Poll 1: If each activation function was identity, the whole function would be:
  - Linear ✓
  - Polynomial with degree bounded by the number of units
  - Arbitrary if there are sufficiently many units

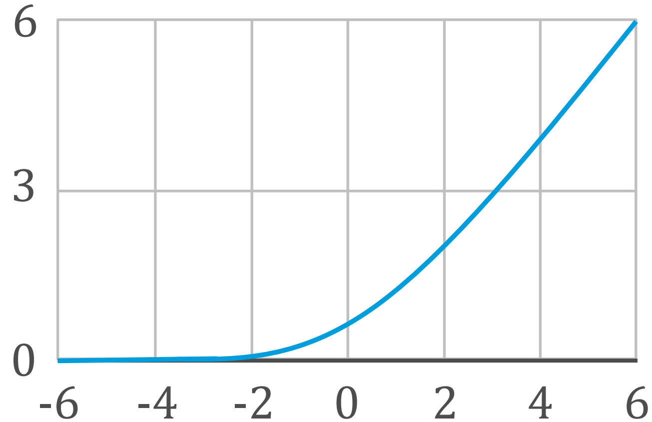
ACTIVATION FUNCTIONS: EXAMPLES



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



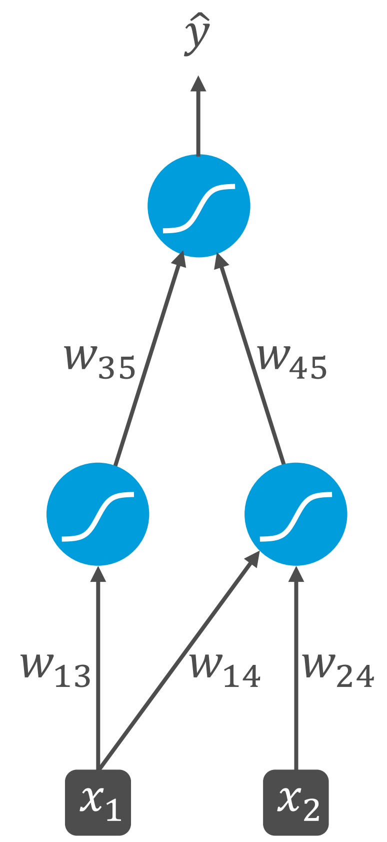
$\text{ReLU}(z) = \max\{0, z\}$



$\text{softplus}(z) = \ln(1 + e^{z})$

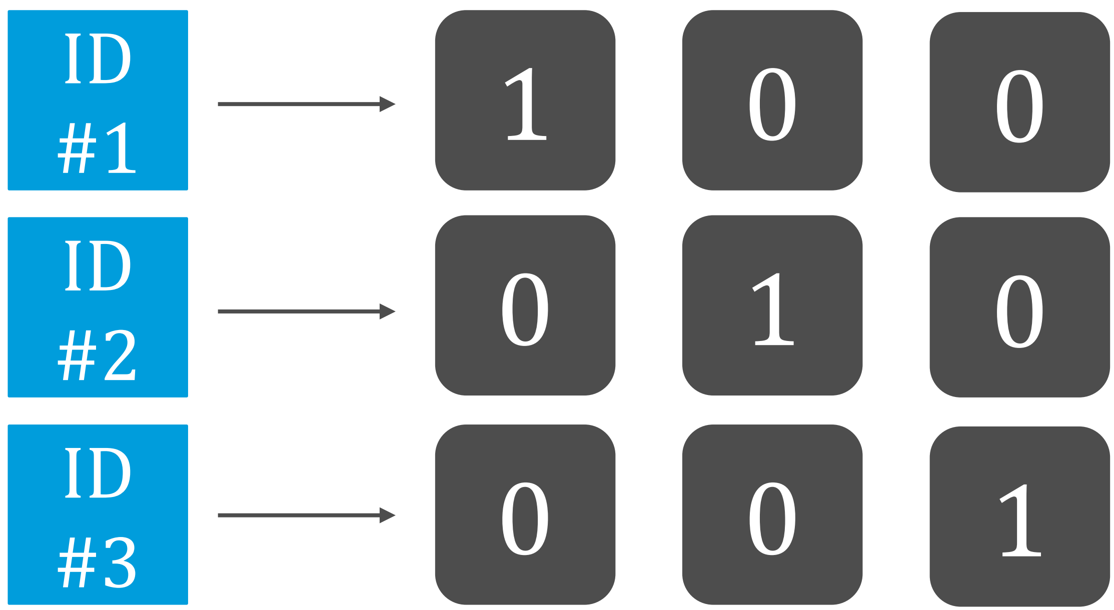
TRAINING NEURAL NETWORKS

- A choice of network architecture (units, activation functions, and edges) defines a hypothesis class whose parameters are the weights on edges
- This hypothesis space is extremely expressive: With just two layers and nonlinear activation functions, neural networks can approximate any continuous function arbitrarily well
- Training can be done "as usual" using gradient descent



INPUT ENCODING

Categorial features are typically encoded using 1-hot encoding



OUTPUT ENCODING

- For binary classification, a sigmoid output unit is often used, and its output is interpreted as the probability of the positive class
- For multiclass classification, we want d output nodes representing probabilities summing up to 1, and this is typically done via a softmax layer, defined by

$$\text{softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{d} e^{z_j}}$$

CONVOLUTIONAL NETWORKS

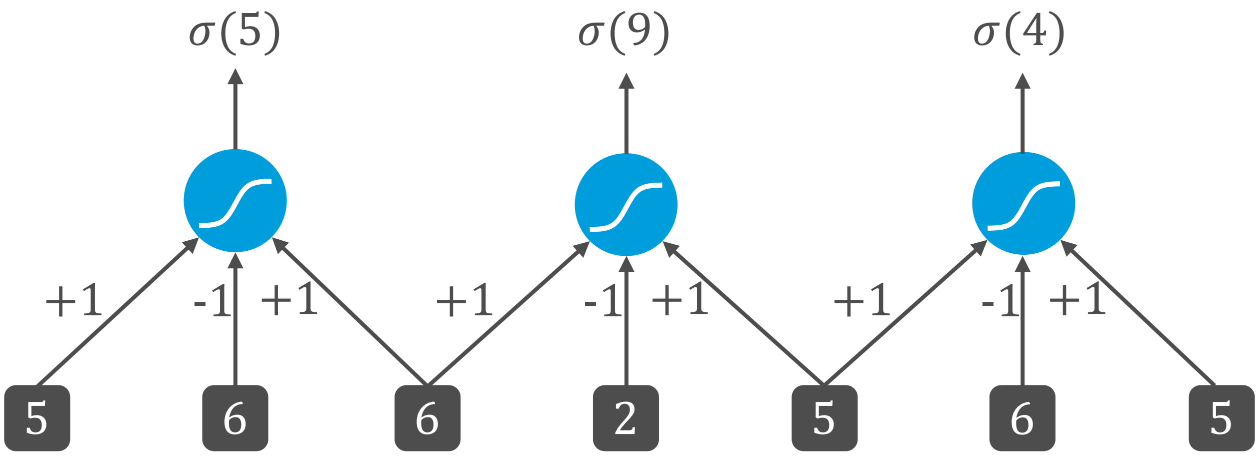
- An image shouldn't be thought of as a vector of pixels, because adjacency matters
- If there are n pixels and n units in the first hidden layer, and they're fully connected, then we already have n^2 weights
- Convolutional neural networks (CNNs) make use of two ideas
  - To respect adjacency, each hidden unit receives input from a local region of the image
  - Anything detectable in one local region would look the same in another local region

KERNELS AND CONVOLUTIONS

- A pattern of weights is called a kernel, and an application of the kernel is a convolution
- Assume for now a 1-D image represented as a vector $\boldsymbol{x}$ of size n, and a vector kernel $\boldsymbol{k}$ of (odd) size ℓ
- The convolution operation is denoted by $\boldsymbol{z} = \boldsymbol{x} * \boldsymbol{k}$, and is defined by

$$z_i = \sum_{j=1}^{\ell} k_j x_{i-(\ell+1)/2+j}$$

KERNEL: EXAMPLE

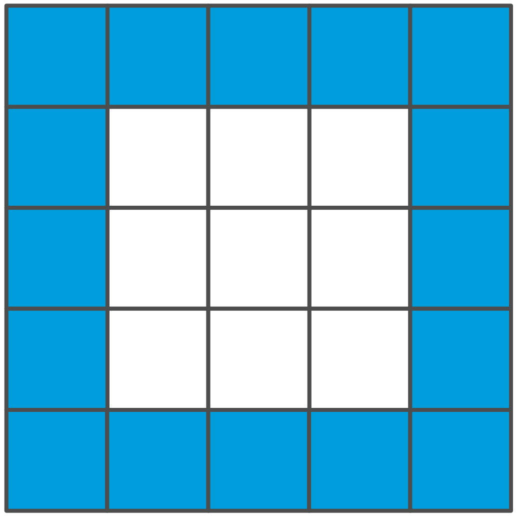


Kernel vector $\boldsymbol{k} = (1, -1, 1)$ that detects a lighter point, applied to $\boldsymbol{x} = (5,6,6,2,5,6,5)$ with a stride of $s = 2$

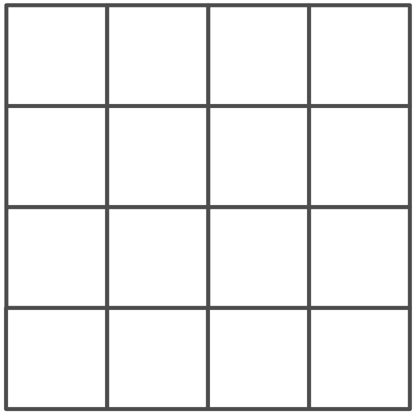
PADDING

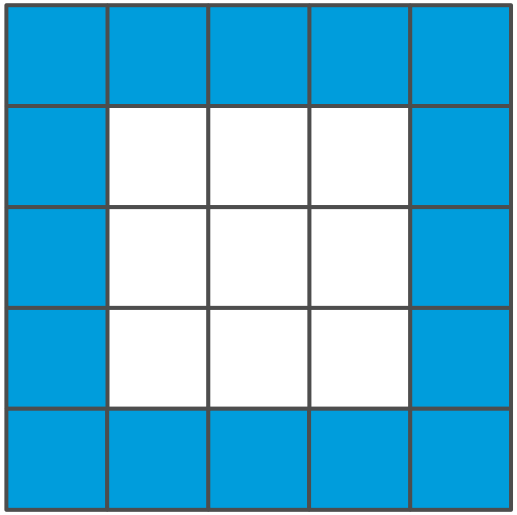
- Poll 2: If we have a 100×100 image and a 5×5 kernel, applied with a stride of 1 (vertically and horizontally), what is the resulting size of the image?
  - 100×100
  - 98×98
  - 96×96 ✓
  - 95×95
- It is often desirable to pad the image to avoid losing information at the boundaries

PADDING ILLUSTRATED

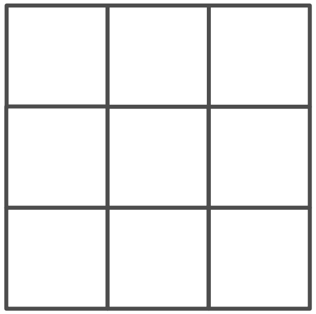


2×2 kernel with a stride of $s = 1$

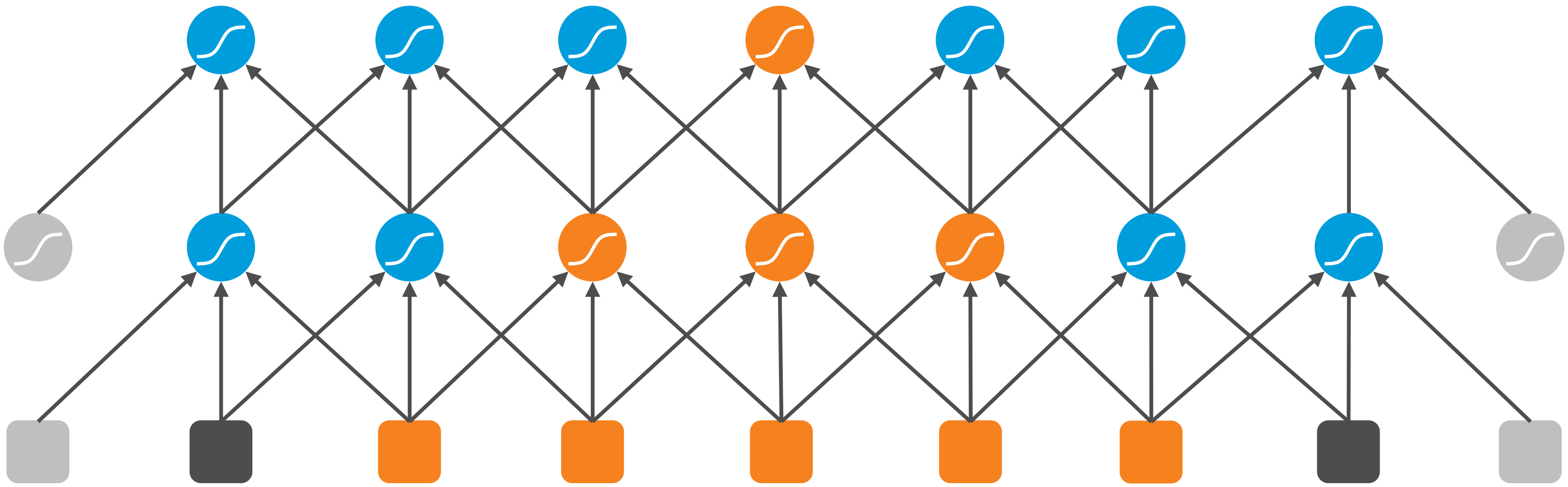




3×3 kernel with a stride of $s = 1$



RECEPTIVE FIELD

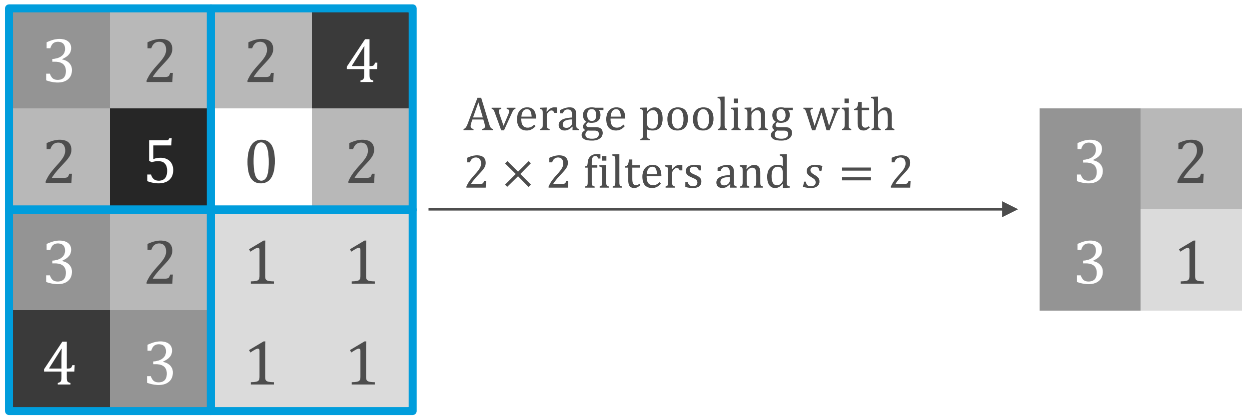


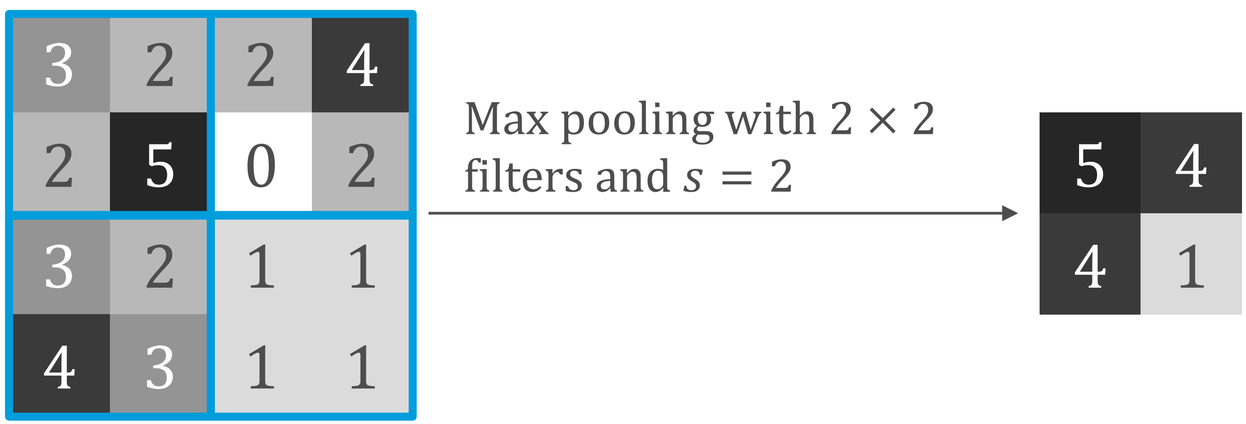
The receptive field of a unit is the portion of the input that can affect the unit. It is ℓ in the first hidden layer but can be larger in deeper layers.

POOLING

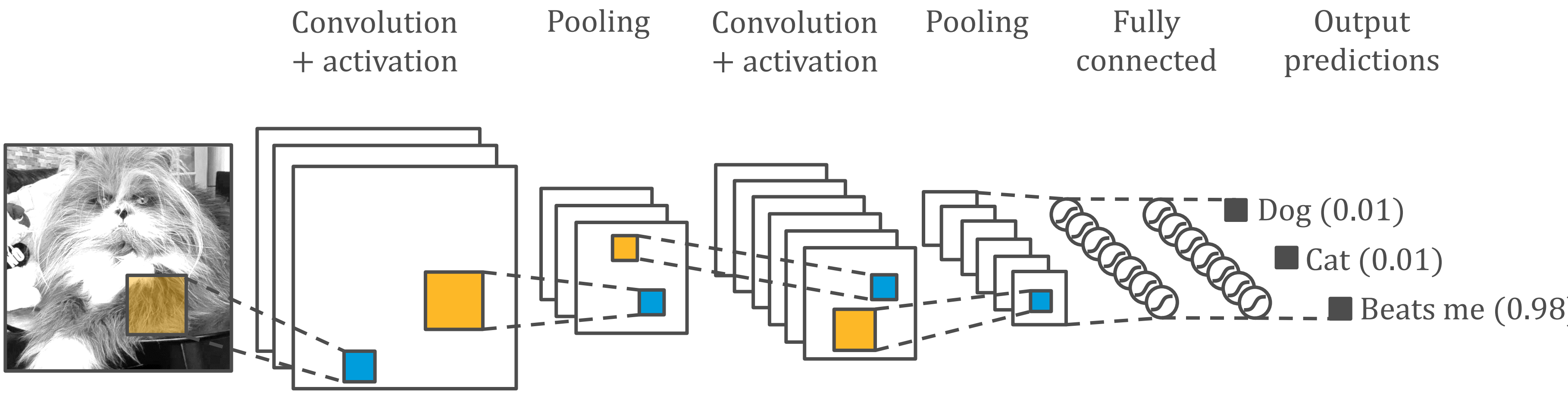
- A pooling layer summarizes adjacent units from a preceding layer
- Like a convolution with kernel size ℓ and stride s but operation is fixed rather than learned and there's no activation function
- Average pooling:
  - Computes the average value of inputs
  - If $\ell = s$, this downsamples the image by a factor of s
- Max pooling:
  - Computes the max value of inputs
  - Acts like a logical disjunction, detecting a feature somewhere in the receptive field

POOLING: EXAMPLE





CNN ARCHITECTURE



Different kernels correspond to different channels, and pooling is applied to each channel separately

SEQUENTIAL MEMORY

- Let us drop the assumption that the neural network is acyclic
- This will allow us to implement the idea of sequential memory

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Given a suffix, it's easy for us to predict the next letter in the sequence

ZYXWVUTSRQPONMLKJIHGFEDCBA

Given a suffix, it's hard for us to predict the next letter in the sequence

RECURRENT NETWORKS

- In a recurrent neural network (RNN), units take their own output as input, which simulates memory
- RNNs are typically used to analyze sequential data, just like HMMs
- As before, we make a Markov assumption: the hidden state $\boldsymbol{z}_t$ captures the relevant information from previous inputs
- We update $\boldsymbol{z}_t = f_{\boldsymbol{w}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_t)$ for a parameter vector $\boldsymbol{w}$
- The trained $f_{\boldsymbol{w}}$ is assumed to capture dynamics that hold for all time steps

RECURRENT NETWORKS

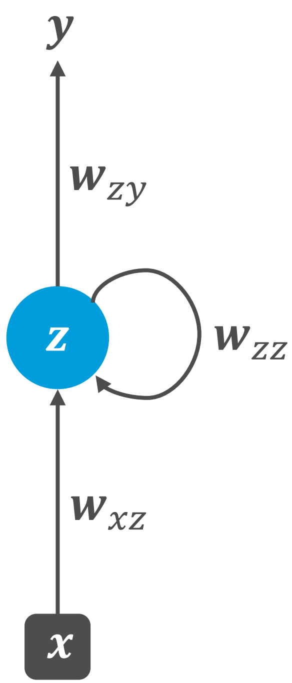
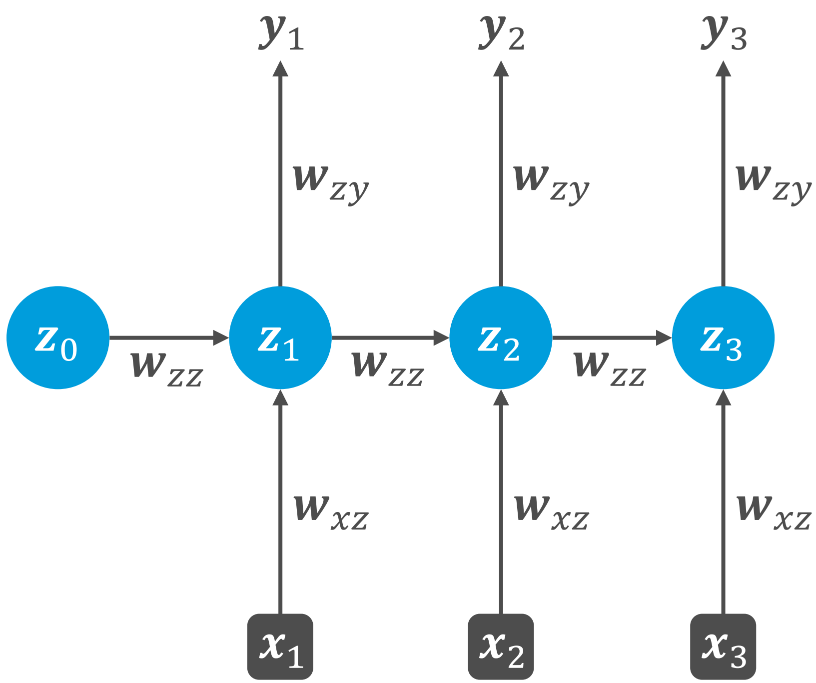


Diagram of a basic RNN where the hidden layer has recurrent connections



Same network unrolled over three time steps to create a feed-forward network

RECURRENT NETWORKS: EXAMPLE

Alphabet is {h,e,l,o}, we want to train an RNN to predict the word “hello” [Example from Andrej Karpathy]