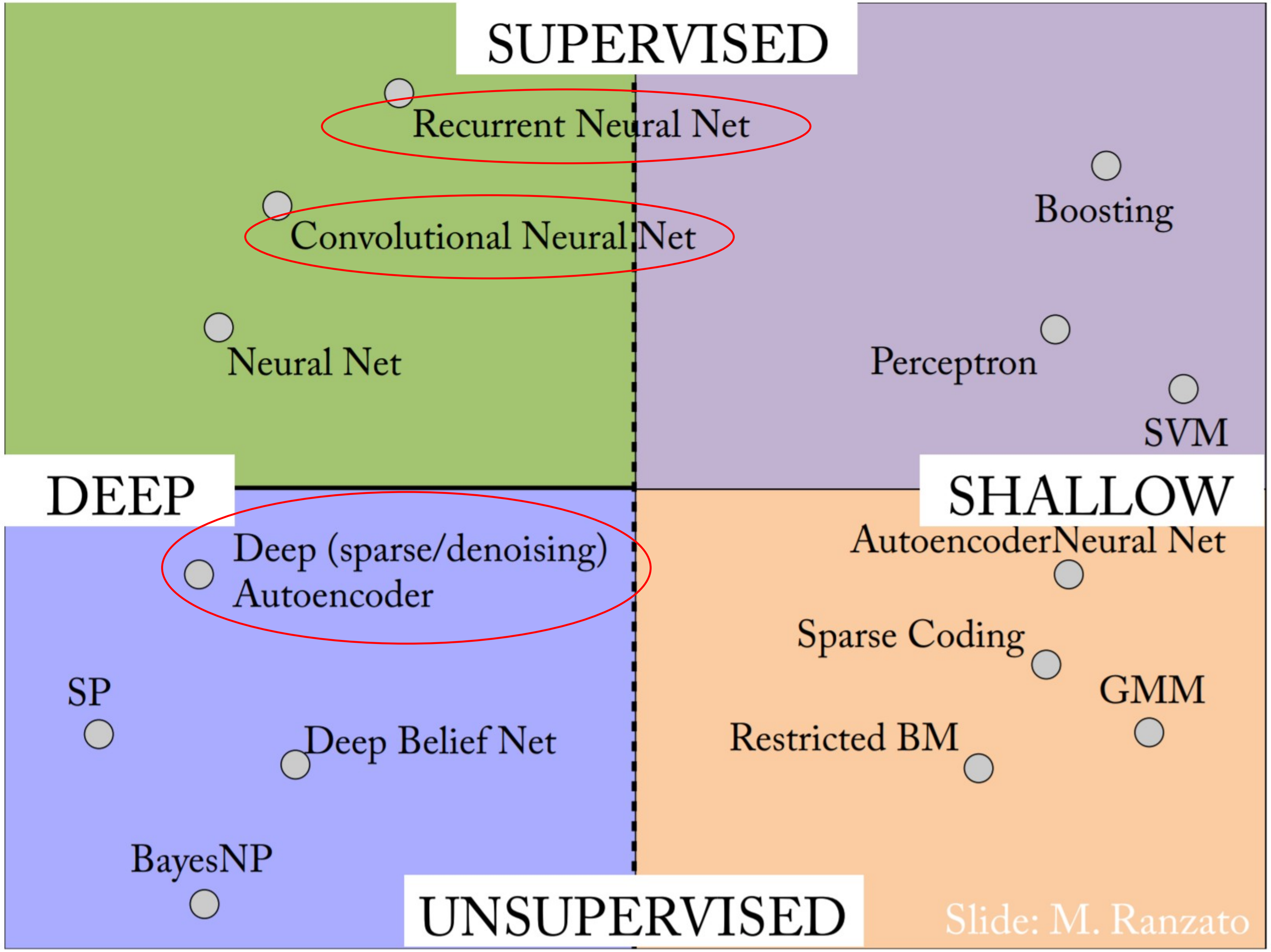
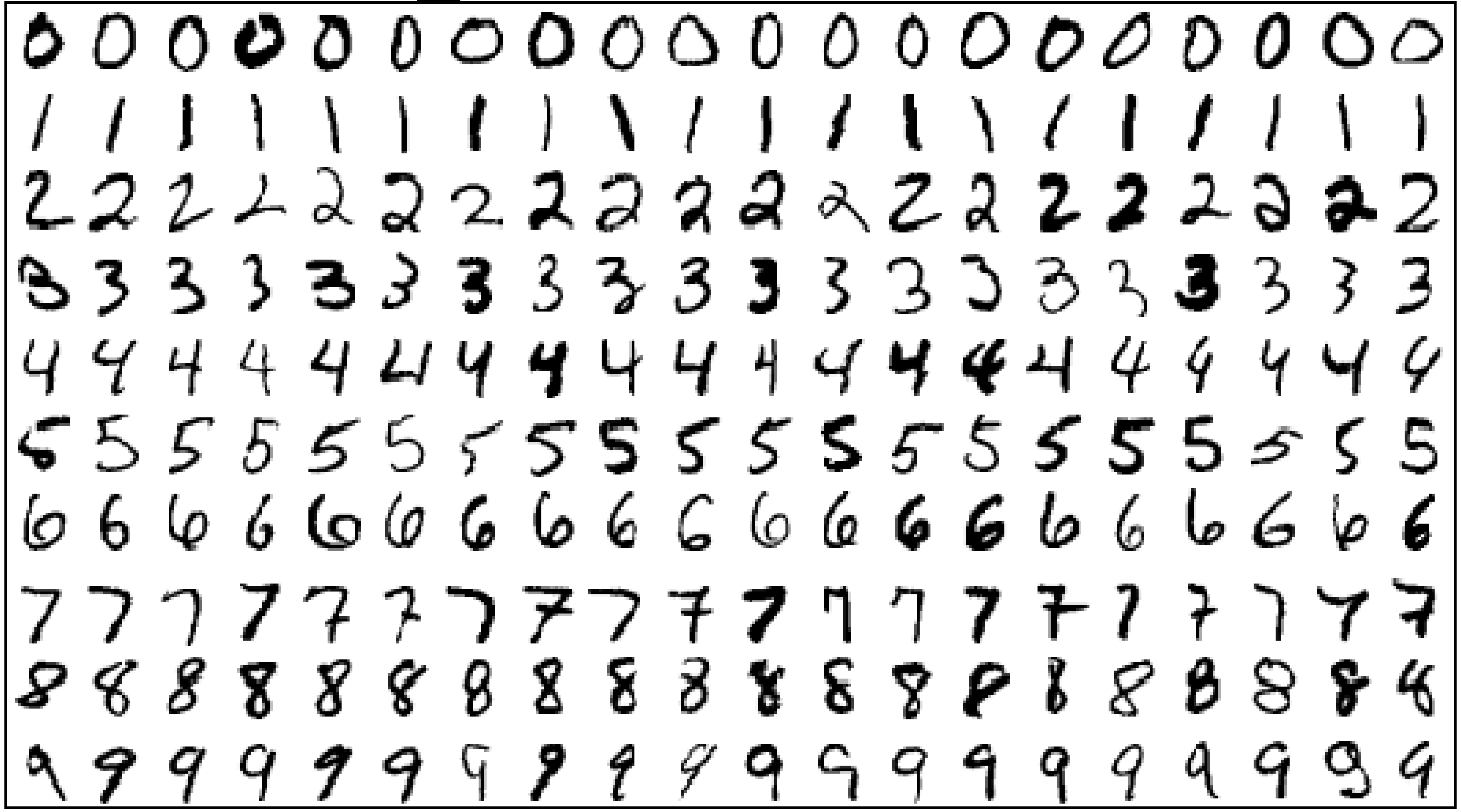
Deep Learning

Deep Learning

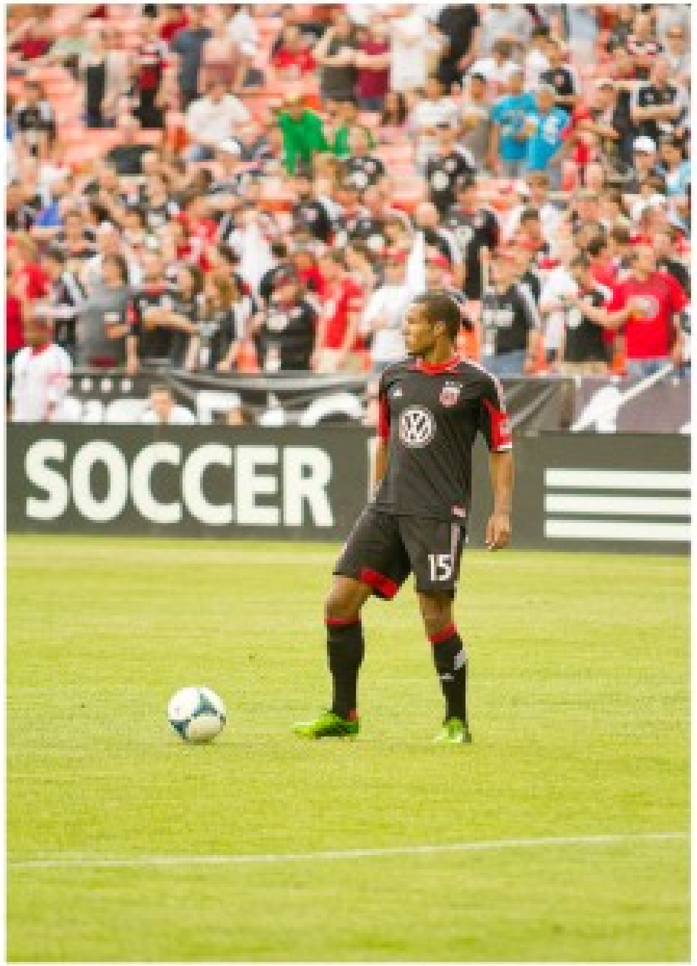
- Most conventional neural networks (MLP) have a few layers
- Deep networks have many layers
- Multiple layers can build an improved feature space
  - First layer learns first order features (edges in an image)
  - Second layer learns higher order features (combinations of edges)
  - And so on…
  - Final layer of transformed features is fed into a set of fully connected layers with backpropagation.


SUPERVISED
Recurrent Neural Net
Convolutional Neural Net
Neural Net
Boosting
Perceptron
SVM
DEEP
SHALLOW
Deep (sparse/denoising) Autoencoder
SP
Deep Belief Net
BayesNP
AutoencoderNeural Net
Sparse Coding
GMM
Restricted BM
UNSUPERVISED
Slide: M. Ranzato

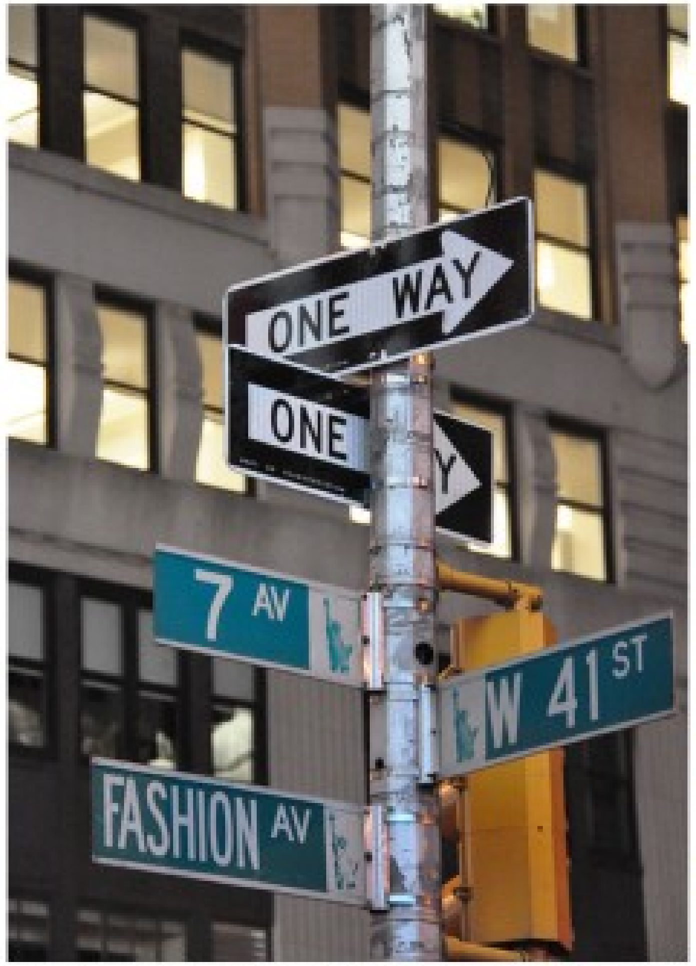
Example: Classifying Hand Written Digits



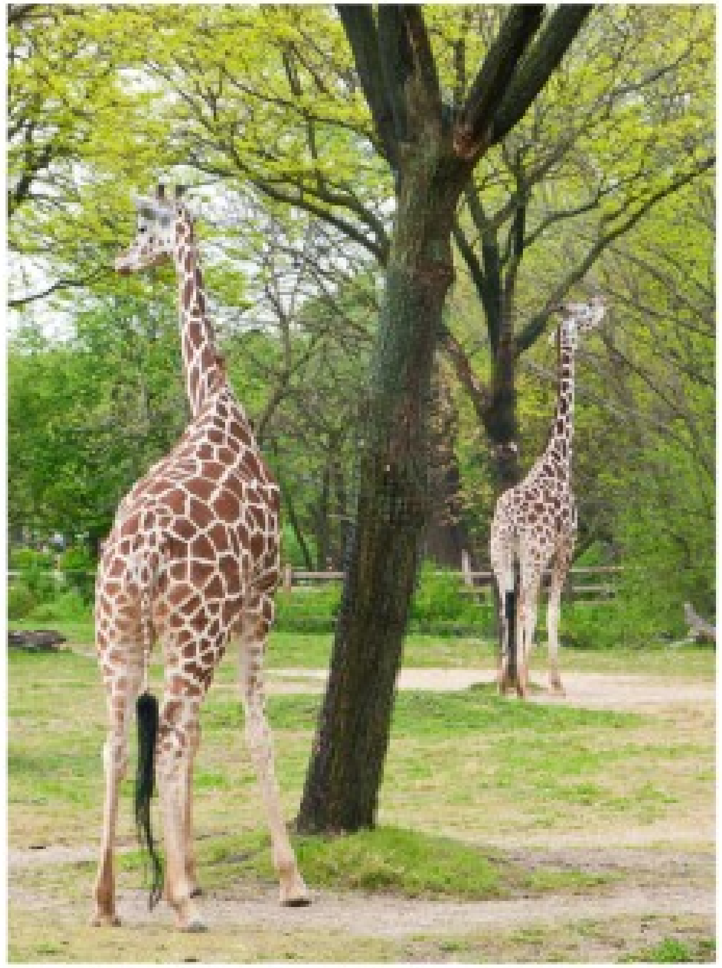
Example: Image Captioning

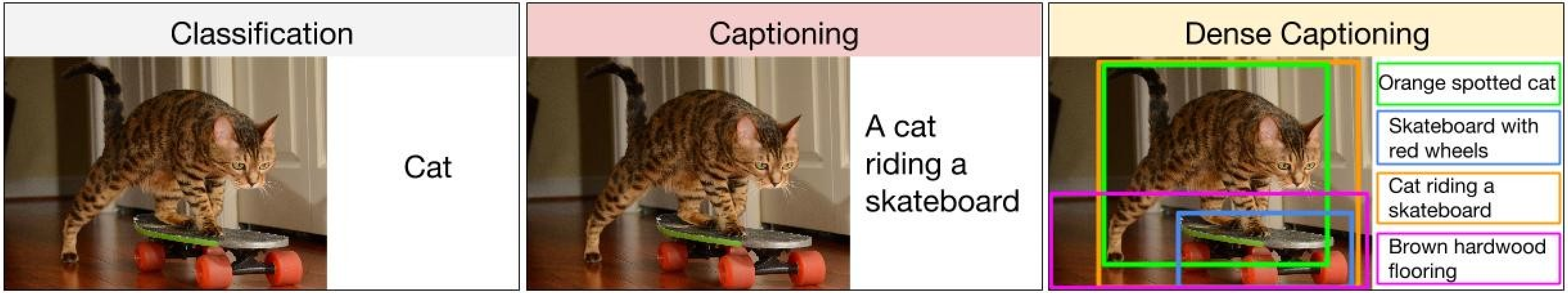


a soccer player is kicking a soccer ball



a street sign on a pole in front of a building


a couple of giraffe standing next to each other



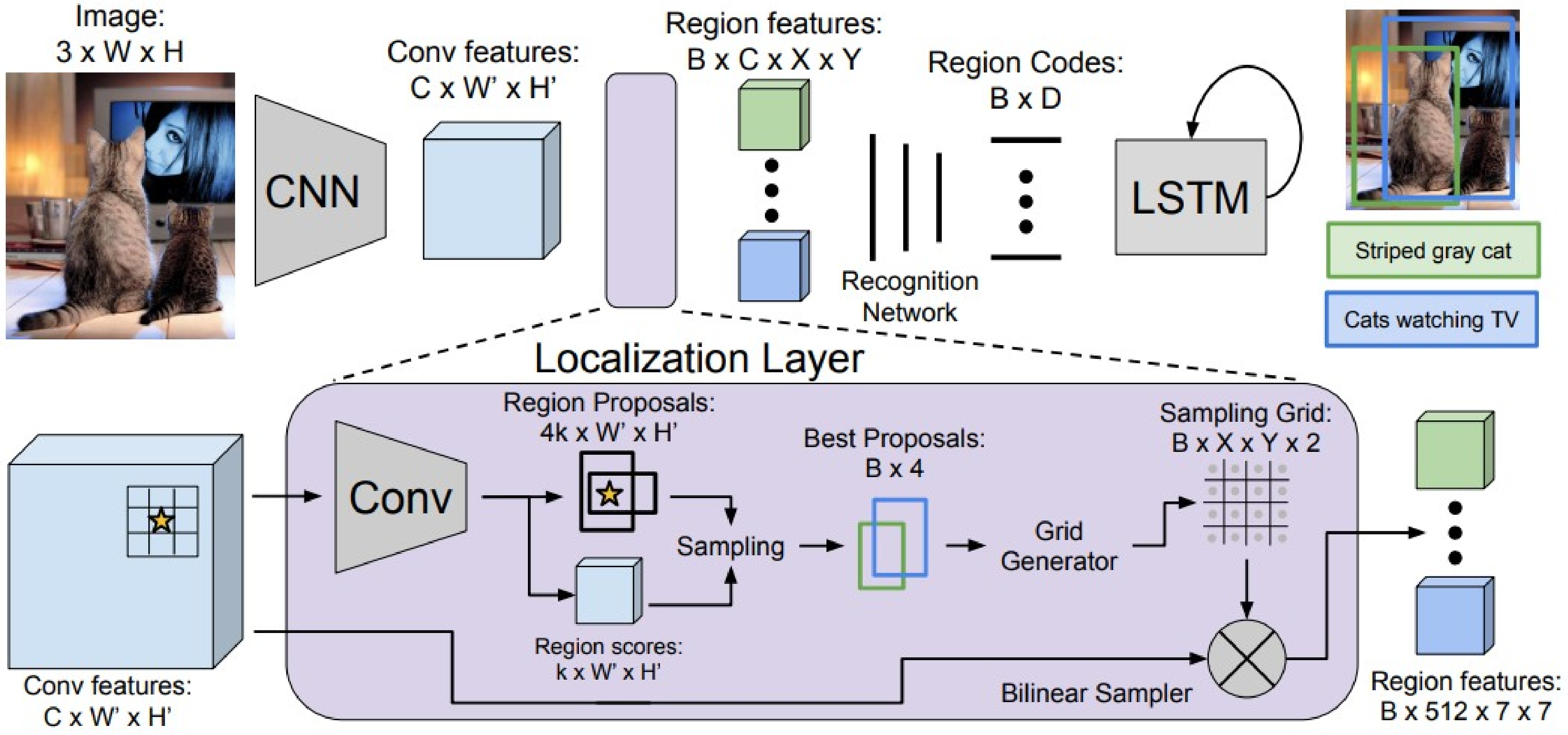
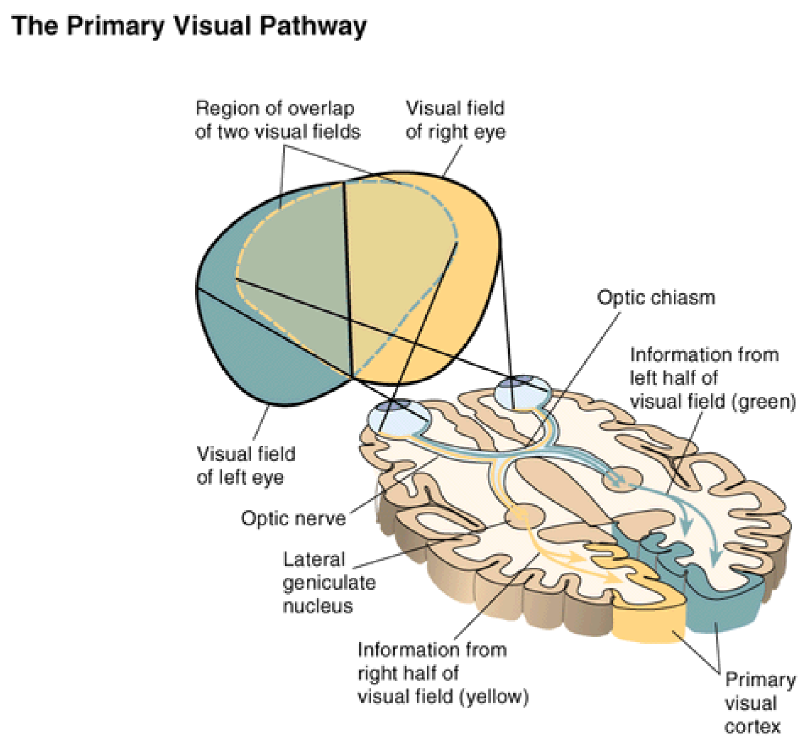


Image: 3 x W x H
CNN
Conv features: C x W' x H'
Region features: B x C x X x Y
Region Codes: B x D
Recognition Network
LSTM
Striped gray cat
Cats watching TV
Localization Layer
Conv features: C x W' x H'
Conv
Region Proposals: 4k x W' x H'
Region scores: k x W' x H'
Sampling
Best Proposals: B x 4
Grid Generator
Sampling Grid: B x X x Y x 2
Bilinear Sampler
Region features: B x 512 x 7 x 7

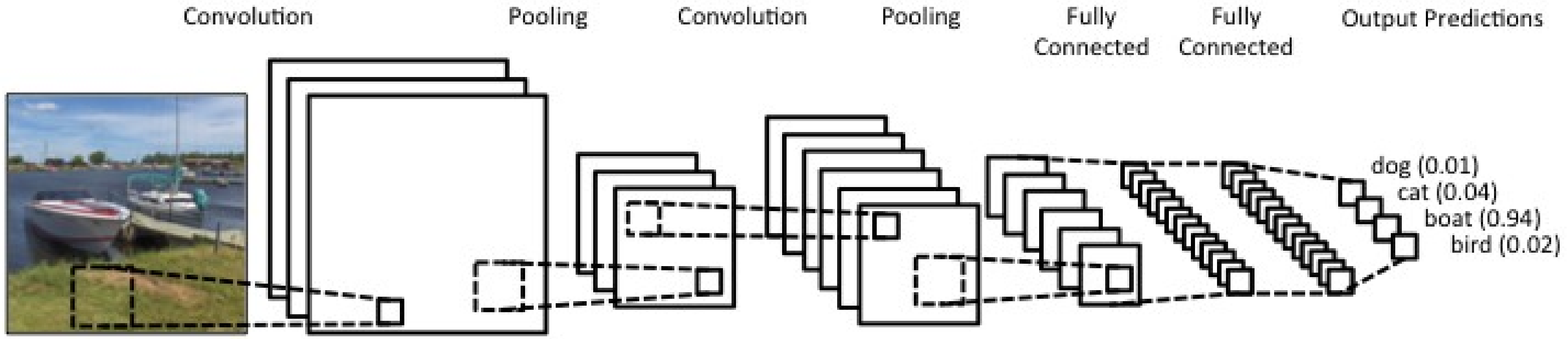
DenseCap (Johnson et al., 2016)

Convolutional Neural Networks (CNNs)

- Emerged from the study of the visual cortex of cats
- Based on the idea of a *local receptive field*
- The local receptive fields of different neurons may overlap
- Some neurons react only to a particular stimuli
- Some neurons react to more complex patterns that are combinations of lower level patterns
- Higher level neurons are based on the outputs of neighboring lower level neurons

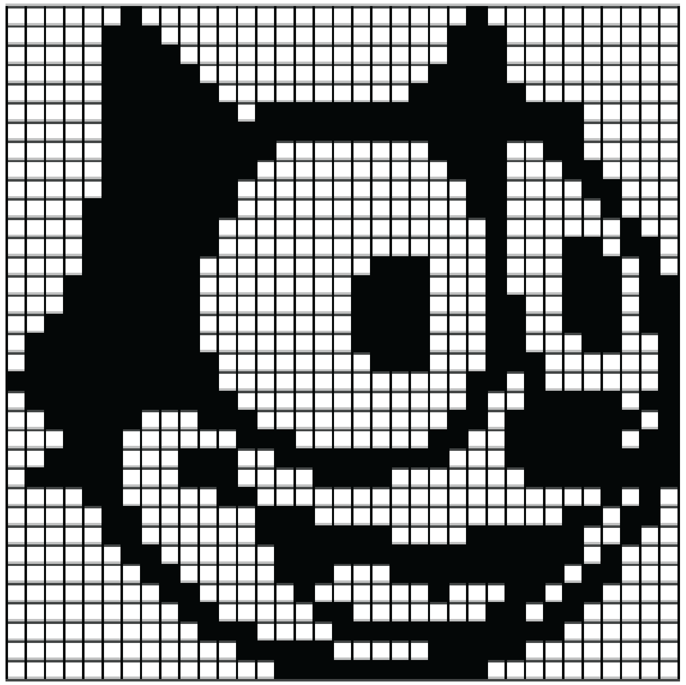

The Primary Visual Pathway
Region of overlap of two visual fields
Visual field of right eye
Optic chiasm
Information from left half of visual field (green)
Visual field of left eye
Optic nerve
Lateral geniculate nucleus
Information from right half of visual field (yellow)
Primary visual cortex

A Simple CNN


Convolution
Pooling
Convolution
Pooling
Fully Connected
Fully Connected
Output Predictions
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)

Back to the Data

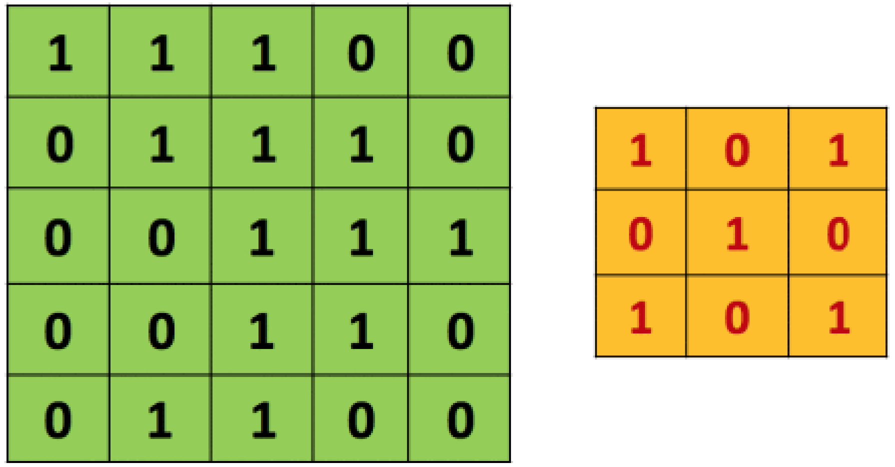
- CNNs are typically used to classify **images**
- Images are a matrix of pixel values
- Grayscale images have <u>one</u> channel
- Color Images have <u>3</u> channels (red, green, and blue)



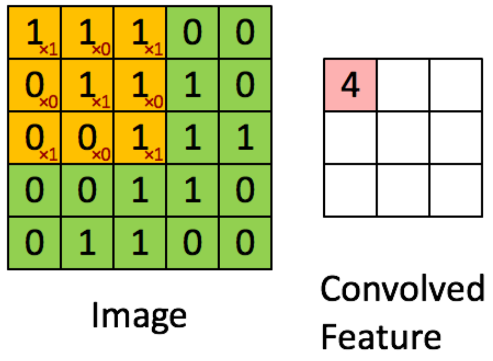
$$\begin{bmatrix}
1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1 \\
1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1 \\
1&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1 \\
1&1&1&1&1&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&1&1 \\
1&1&1&1&1&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\
1&1&1&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1 \\
1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1 \\
1&1&1&1&1&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&0&0&1&1&1&1&1 \\
1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&0&0&1&1&1&1 \\
1&1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&0&0&1&1&1 \\
1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&0&1&1&1 \\
1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1&0&1&1 \\
1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&0&1&0&0&1 \\
1&1&1&1&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&0&1&1&1&0&0&0&1&0&1 \\
1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&0&1&1&1&0&0&0&1&0&0 \\
1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&0 \\
1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&0 \\
1&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&0&0&1&1&1&0&0&1&1&0 \\
1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&1&1&1&0&0&0&1&1&1&1&1&1&0 \\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&0&1&1&1&1&1&1&0 \\
1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&0&0&0&0&0&1&0&0 \\
1&1&0&0&0&0&0&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1&1&0&0&1&0&0&0&0&0&0&0&1&0 \\
1&1&1&0&0&0&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&0&0&1&1&0&0&0&0&0&0&1&0&0 \\
1&1&0&0&0&0&1&1&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0 \\
1&0&0&0&0&0&1&1&1&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0 \\
1&1&1&1&0&0&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&0&1 \\
1&1&1&1&1&0&0&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&0&0&1 \\
1&1&1&1&1&0&0&1&1&1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&1&1&0&1&1 \\
1&1&1&1&1&1&0&0&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&1 \\
1&1&1&1&1&1&1&0&0&1&1&1&1&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&1 \\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&0&1&1&1&1&1&1&0&1&1&1&0&0&1&0&0&1&1&1&1 \\
1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&0&0&1&1&1&1&1&1&1&0&0&1&0&0&1&1&1&1&1 \\
1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1 \\
1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&1&1&1&1&1&0&0&0&0&0&1&1&1&1&1&1&1&1 \\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1
\end{bmatrix}$$

Convolution Step

- The purpose of convolution is to ***extract features*** from the image
- The convolution step in a CNN applies a *sliding window-like filter* over the image
- For every position, the element wise multiplication between the two matrices is computed to create the convolved feature



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

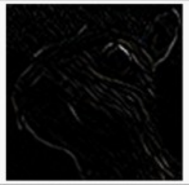
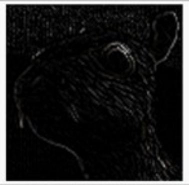
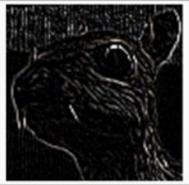
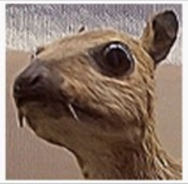
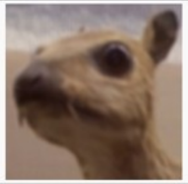
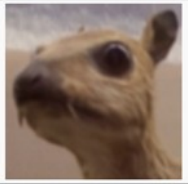


$$\begin{bmatrix} 1_{\times 1} & 1_{\times 0} & 1_{\times 1} & 0 & 0 \\ 0_{\times 0} & 1_{\times 1} & 1_{\times 0} & 1 & 0 \\ 0_{\times 1} & 0_{\times 0} & 1_{\times 1} & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Types of Filters

- There are different types of filters that can be applied to an image for feature detection.

- Different filters can detect different features from an image

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Another Example

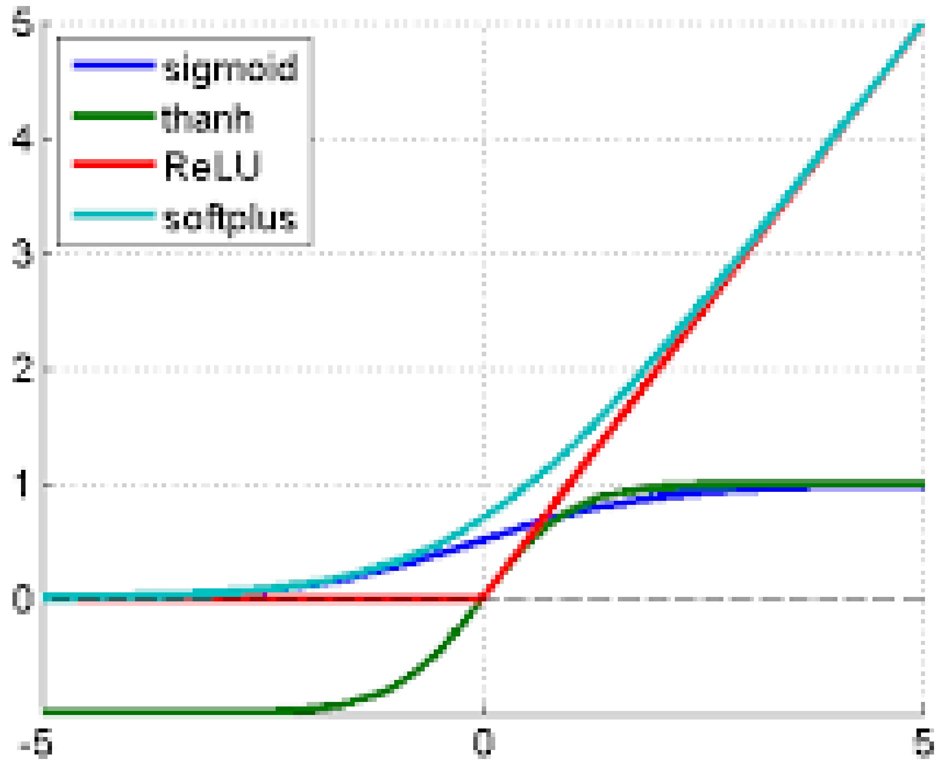
Considerations for Convolution

- **Depth:** corresponds to *the number of filters* we use for the convolution operation.
  - Different filters find different features
  - Three filters result in a feature map depth of three
- **Stride:** the number of pixels by which we slide our filter matrix over the input matrix.
  - When the stride is 1 then we move the filters one pixel at a time.
  - When the stride is 2, then the filters jump 2 pixels at a time as we slide them around.
  - Having a larger stride will produce smaller feature maps.
- **Zero-padding: p**adding the input matrix with zeros around the border
  - Allows the filter to be applied to bordering elements of our input image matrix.
  - Allows for the control of the size of the feature maps.

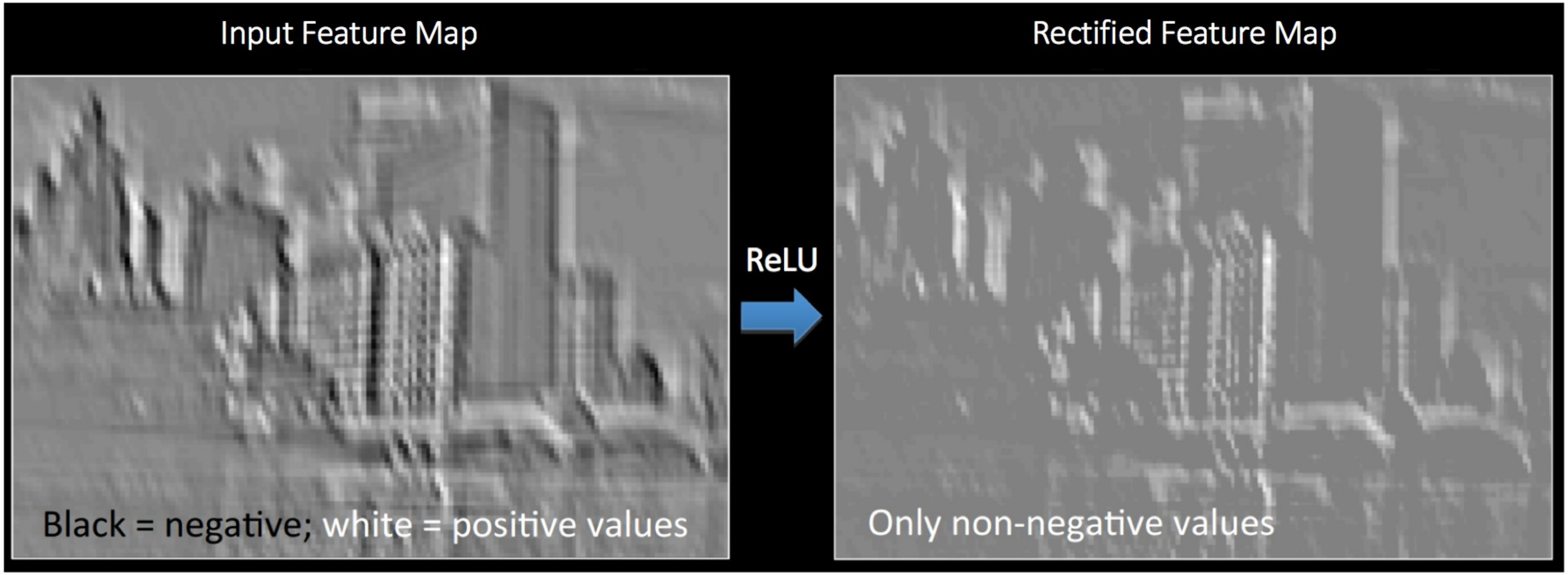
Introducing Non-Linearity

- Convolution is a linear operation
- Rectified linear unit function (ReLU)
- Applied per pixel of the feature map
- Computes a linear function of the inputs and outputs the result if it is positive and 0 otherwise

  Output = Max(0, Input)
- Effectively replaces all negative values in the feature map with 0
- Other functions
  - tanh, sigmoid, softmax

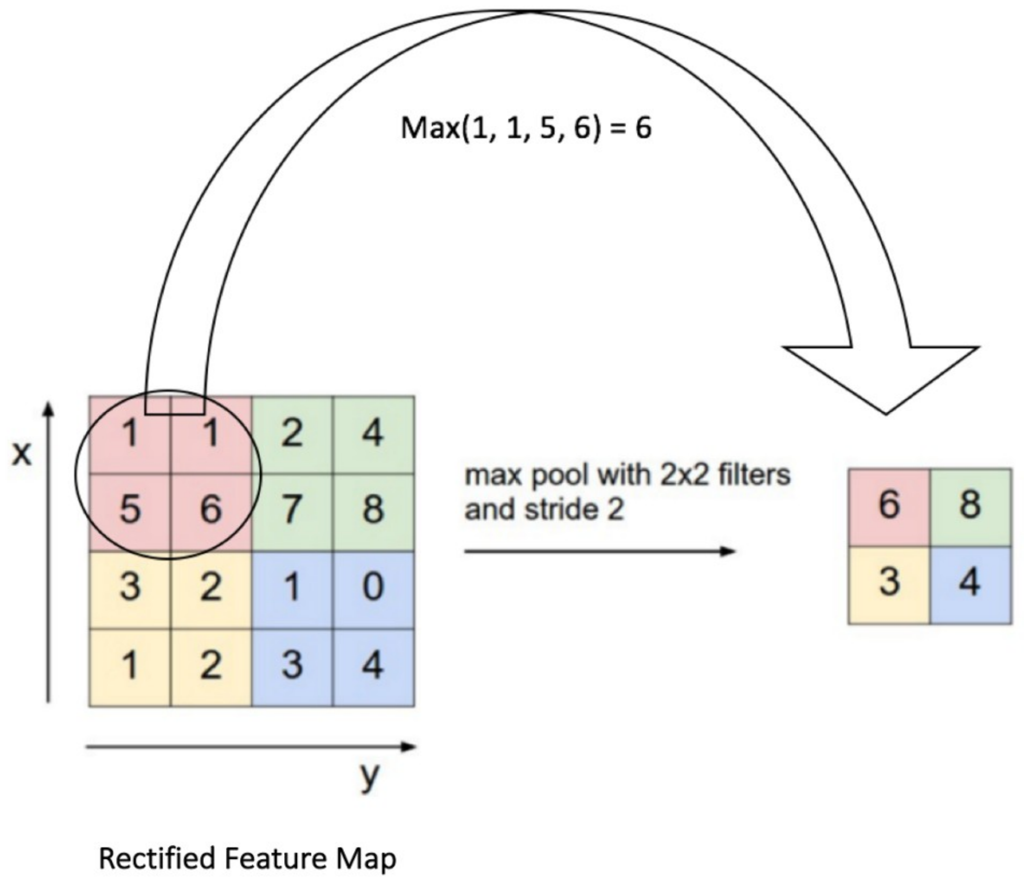


ReLU Example

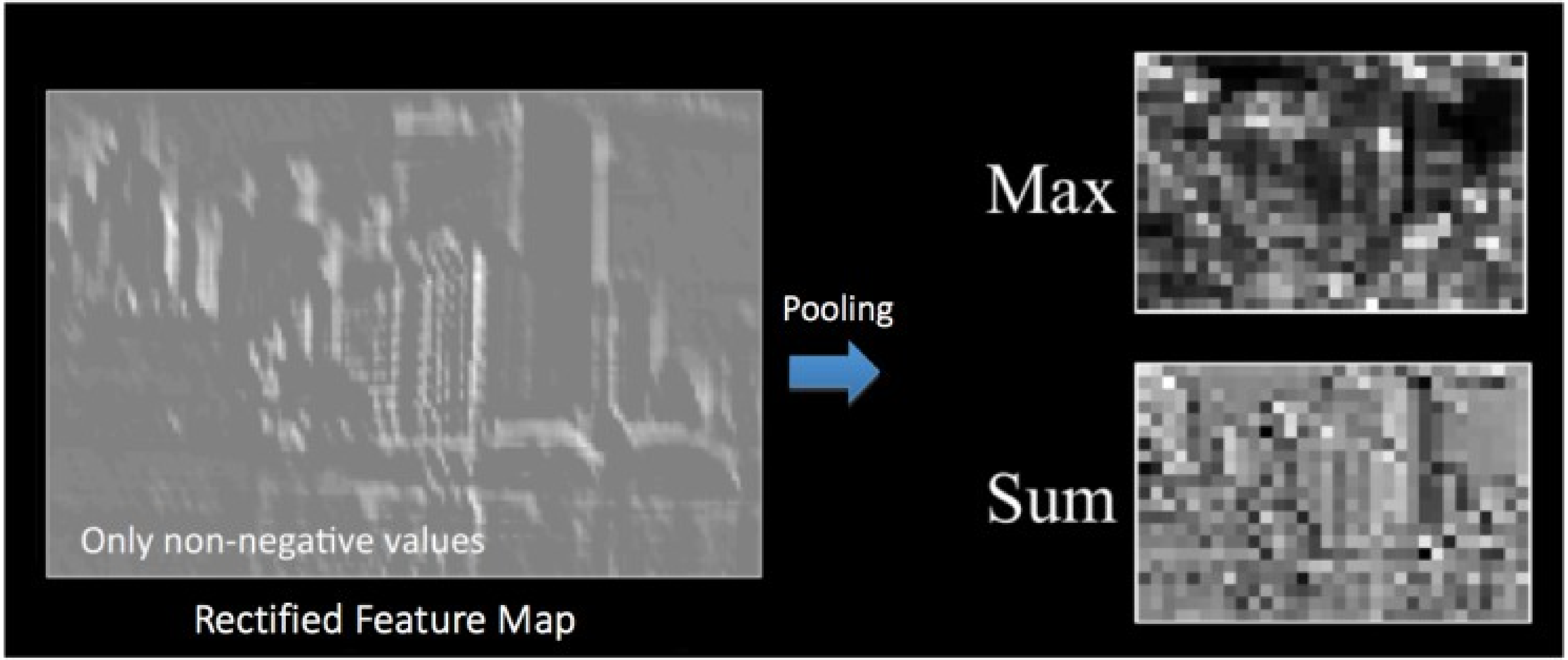


Pooling Step

- **Subsamples** the feature map
- Reduces the size of each feature map retaining important information
- Can be of different types (max, average, sum)
- Parameters
  - size of kernel
  - Stride
  - padding

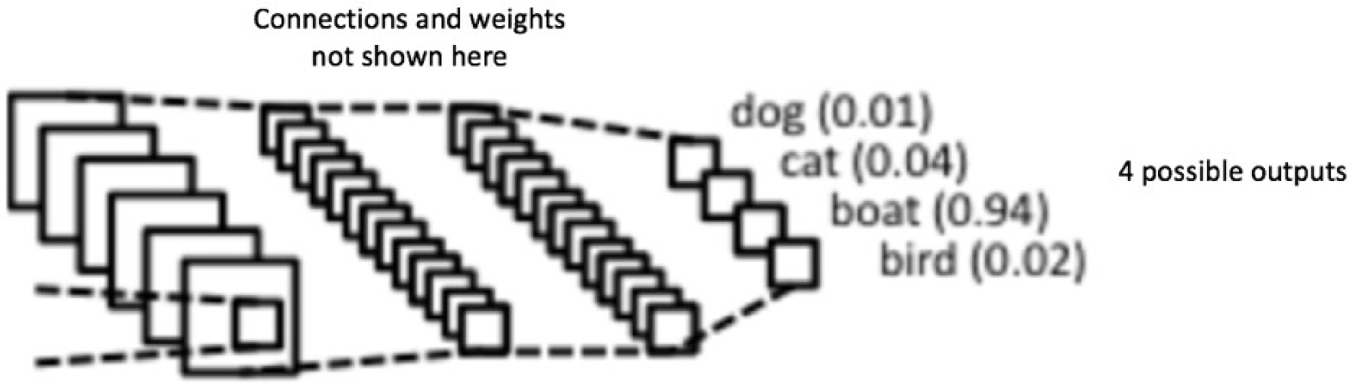


Pooling Example

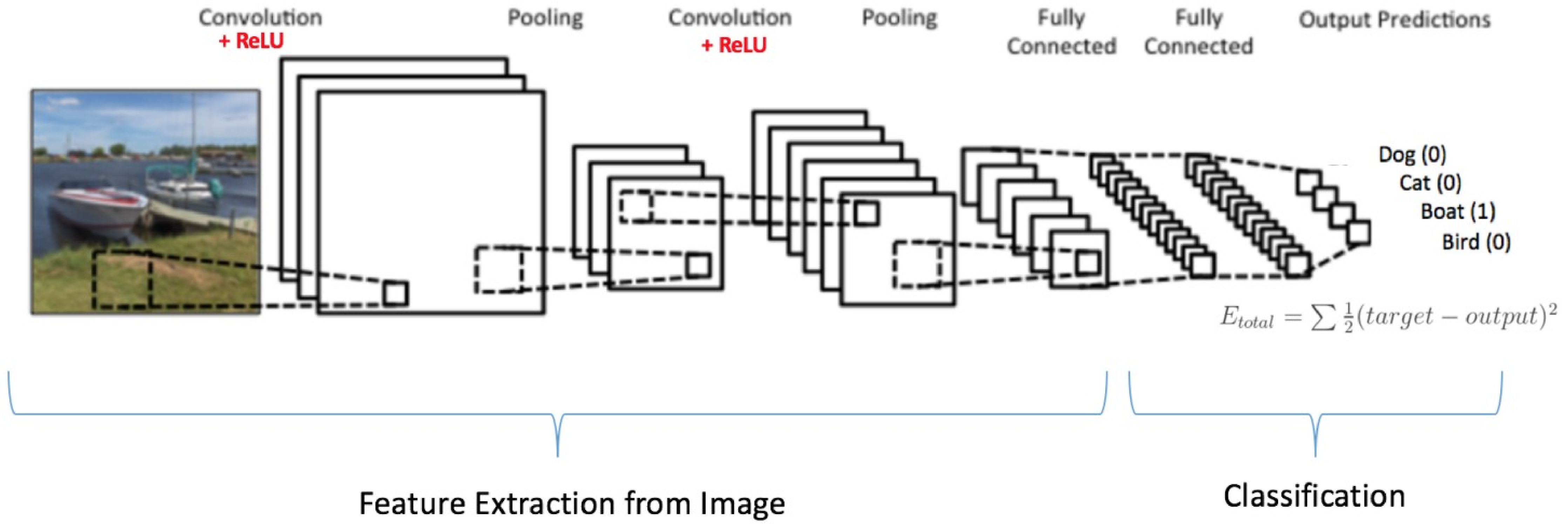


Fully Connected Layers

- Features of pooling layer is used as input to fully connected layer
- Output of convolution and pooling layers represent high-level features of the image
- Fully connected layer is a traditional multilayer perceptron.
- High level features are fed to the fully connected layer to classify the image



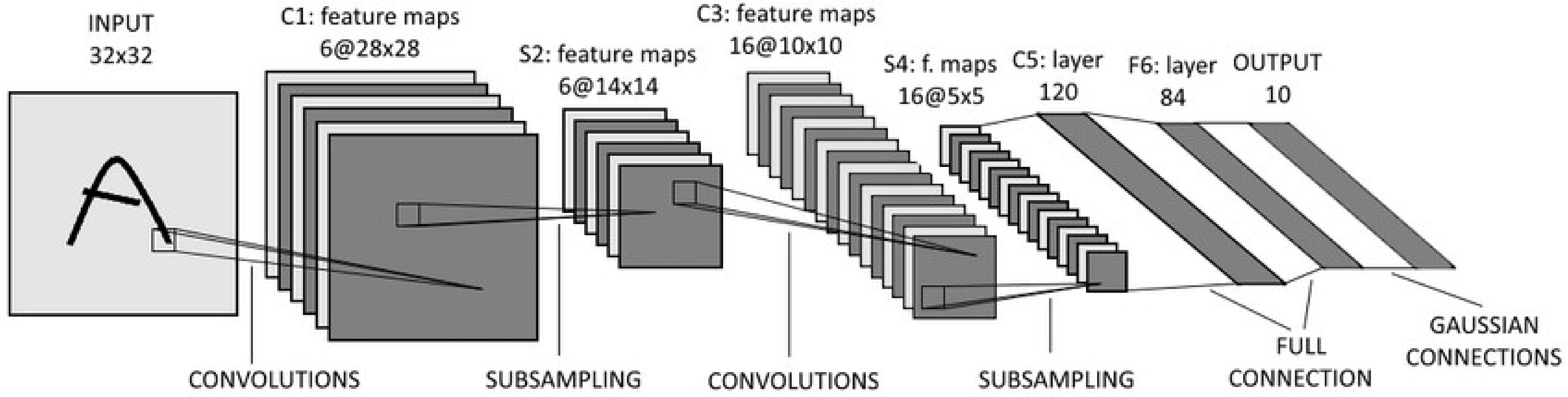
Training the Network



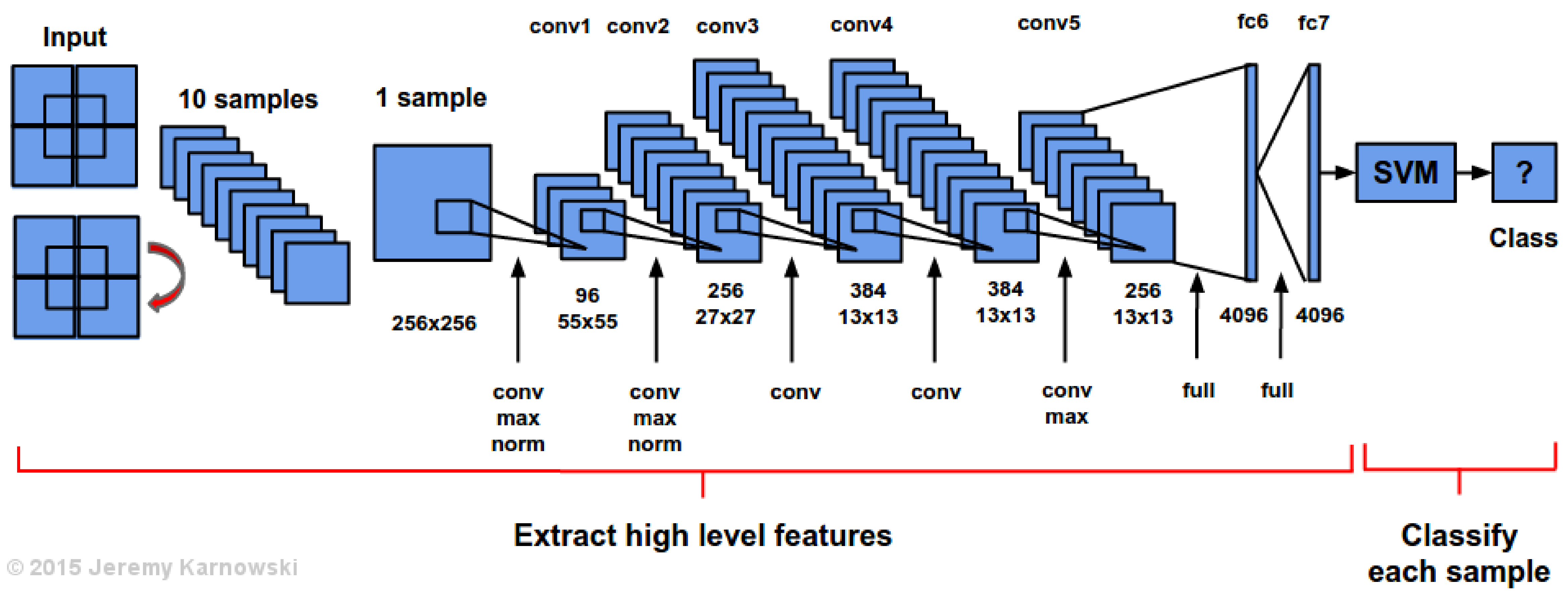
The Problem with HyperParameters

- CNNs have a large number of hyperparameters
  - Number of layers
  - Convolution parameters (depth, stride, padding)
  - Non-linear function (ReLU, softmax, sigmoid, tanh)
  - Pooling layer parameters (kernel size, stride, padding)
  - Fully connected layer (number of hidden layers, optimization algorithm, learning rate, momentum term, etc…)
  - Weight initialization
  - Dropout (reduces overfitting)
- Still a lot of open issues

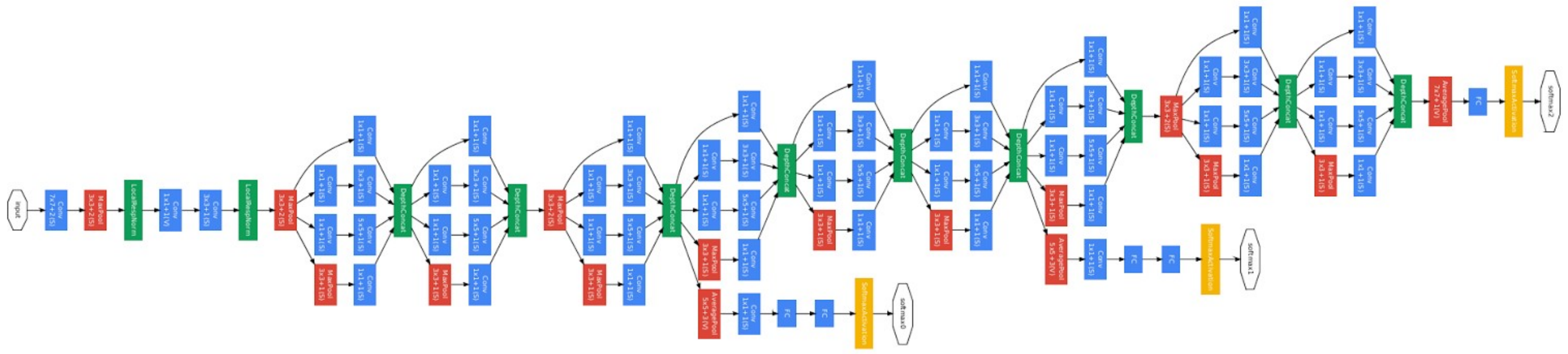
Popular CNN Architectures: LeNet-5 (LeCun, 1998)



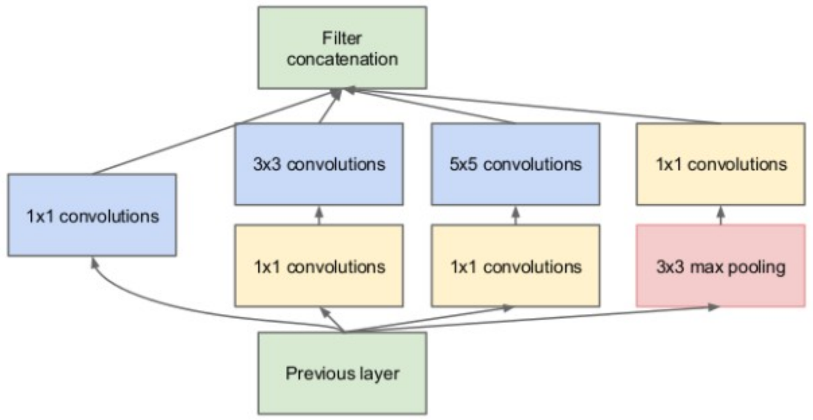
Popular CNN Architectures: AlexNet (Krizhevsky, 2012)



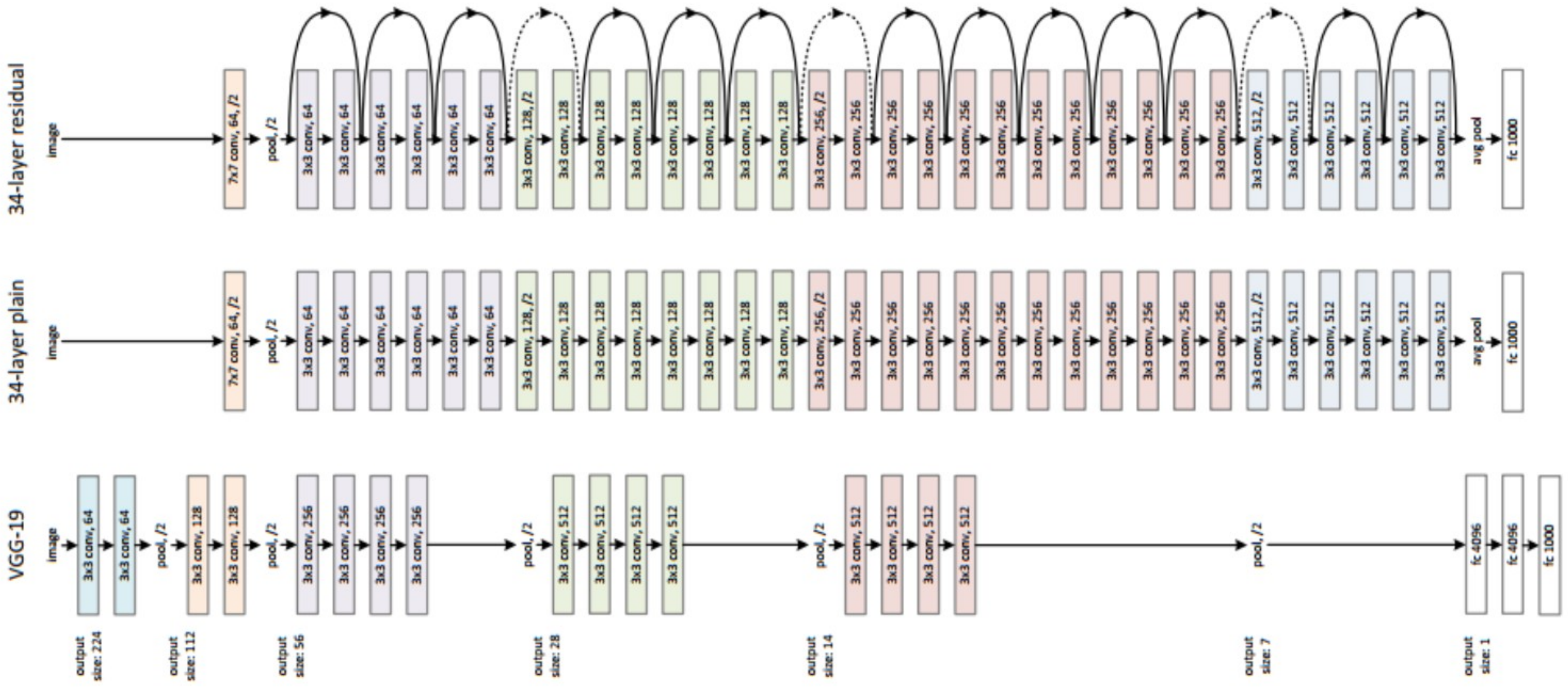
Popular CNN Architectures: GoogLeNet (Szegedy et al., 2015)

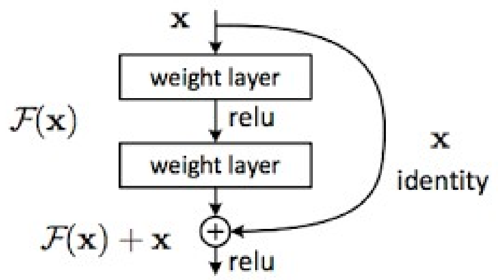


Inception Module

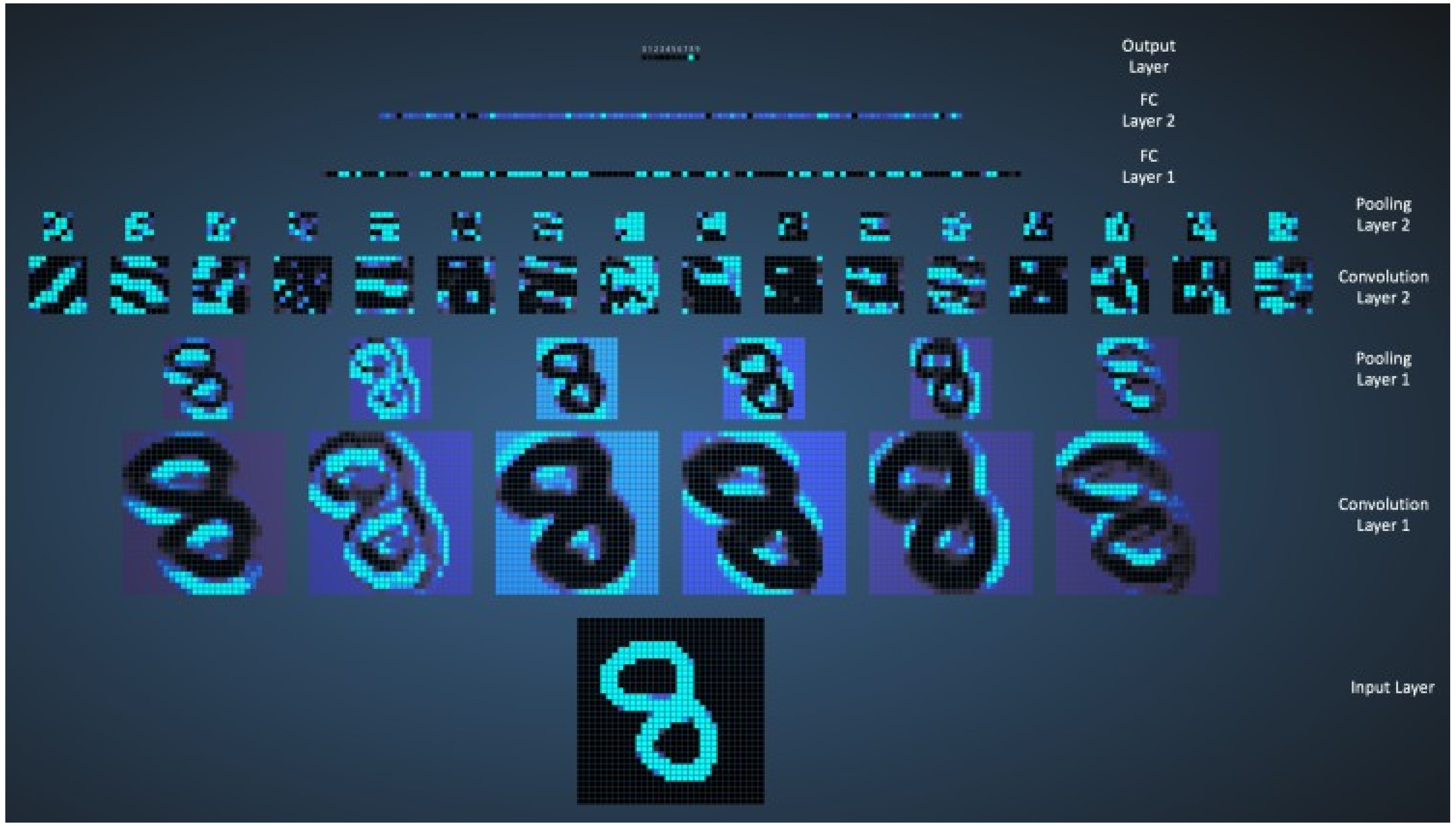


Popular CNN Architectures: ResNet (He, 2015)




x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

CNN Demo


0 1 2 3 4 5 6 7 8 9
Output Layer
FC Layer 2
FC Layer 1
Pooling Layer 2
Convolution Layer 2
Pooling Layer 1
Convolution Layer 1
Input Layer

Recurrent Neural Networks

- *We are the sum total of our experiences. None of us are the same as we were yesterday, nor will be tomorrow. (B.J. Neblett)*

Recurrent Neural Networks

- Idea: our brain learns from experience and that experience is used to solve new problems
- Human learning can be distilled into the following:
  - <u>Memorization:</u> every time we gain new information, we store it for future reference.
  - <u>Combination:</u> not all tasks are the same, so we couple our analytical skills with a combination of our memorized, previous experiences to reason about the world.
- Our brains work with <u>sequences</u> to learn

Recurrent Neural Networks

- Traditional neural networks operate on the fundamental assumption that inputs are independent.
  - The output at any given time is completely determined by the input at the same time.

- Recurrent neural networks map an entire history of inputs to each output.

- Can be thought of as a traditional neural network with a loop

Memory in Recurrent Neural Networks

Weights in a Recurrent Neural Network

$$y_0 = f(W_x X_0)$$
$$y_1 = f(W_x X_1)$$
$$y_2 = f(W_x X_2)$$

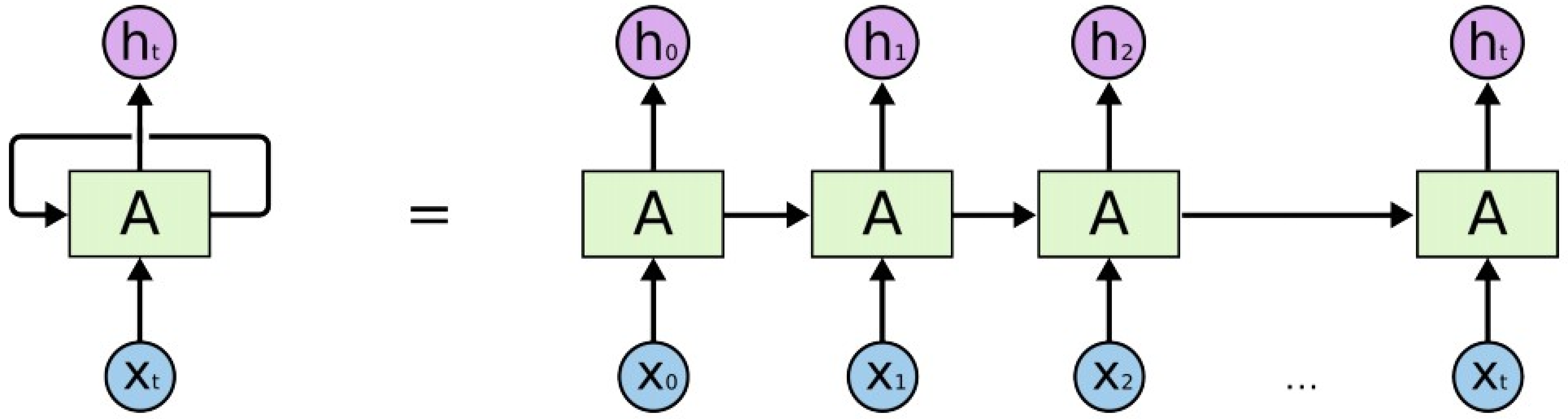
Weights with Current Input and Previous Input

$$y_0 = f(W_x X_0)$$
$$y_1 = f(W_x X_1 + W_h X_0)$$
$$y_2 = f(W_x X_2 + W_h X_1)$$

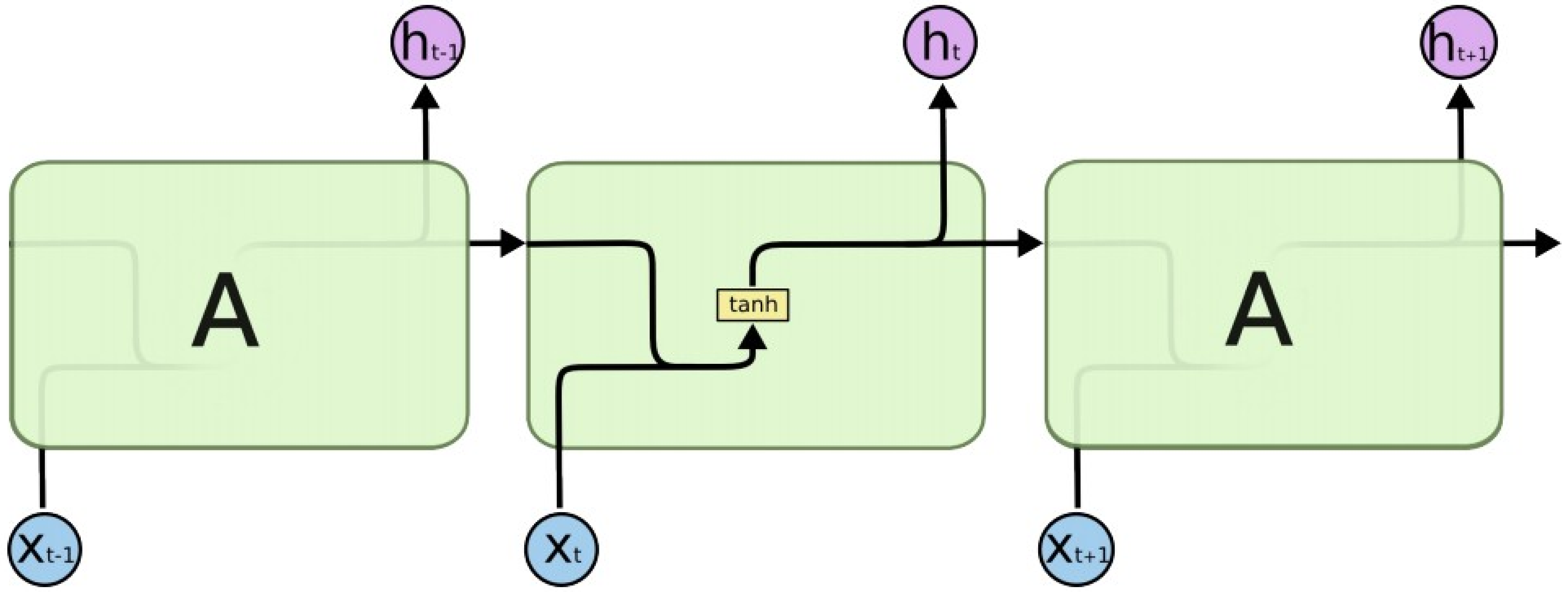
Weights with a **Combination** of Current Input and All Previous Inputs

$$y_0 = f(W_x X_0)$$
$$y_1 = f(W_x X_1 + W_h\ f(W_x X_0))$$
$$y_2 = f\left(W_x X_2 + W_h\ f(W_x X_1 + W_h\ f(W_x X_0))\right)$$

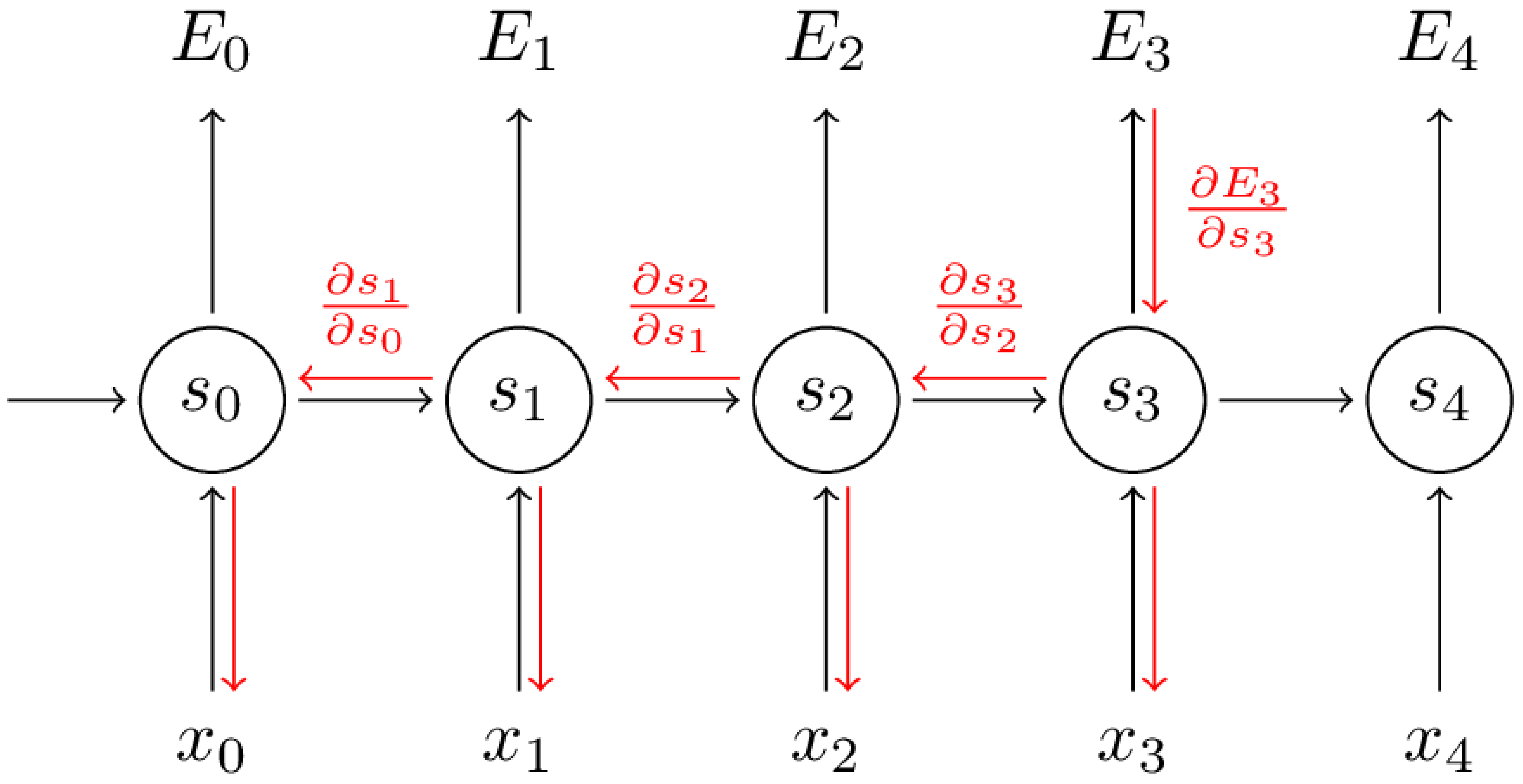
An Unrolled Recurrent Neural Network



Standard RNN Architecture



Back Propagation in an RNN

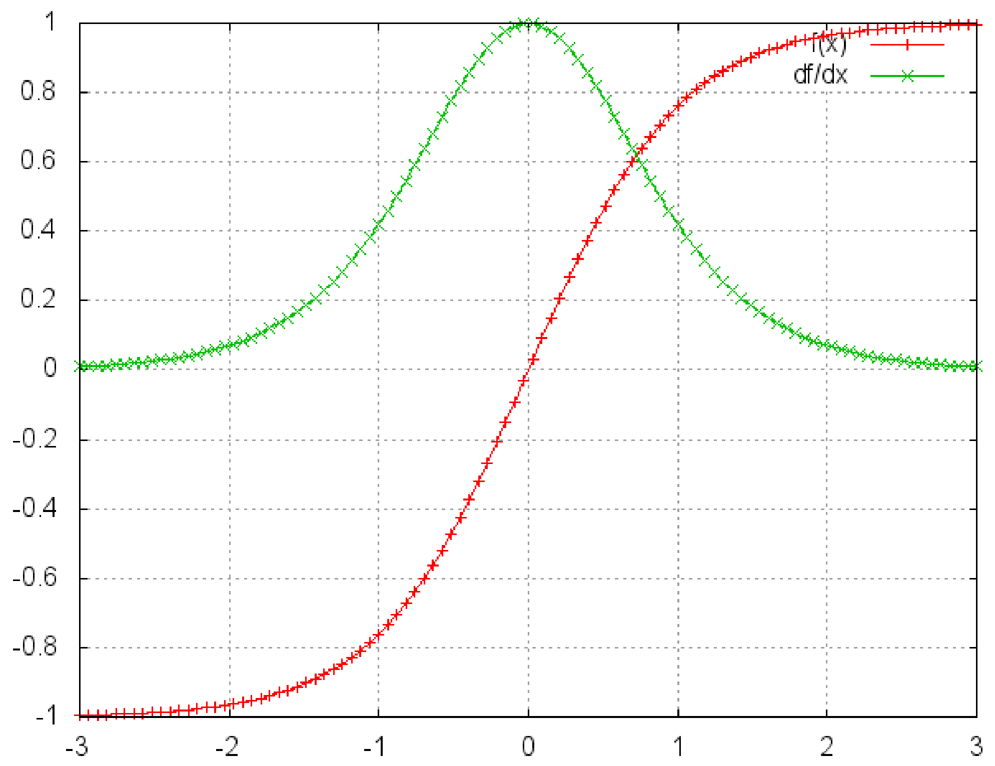

E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4

The Vanishing Gradient Problem

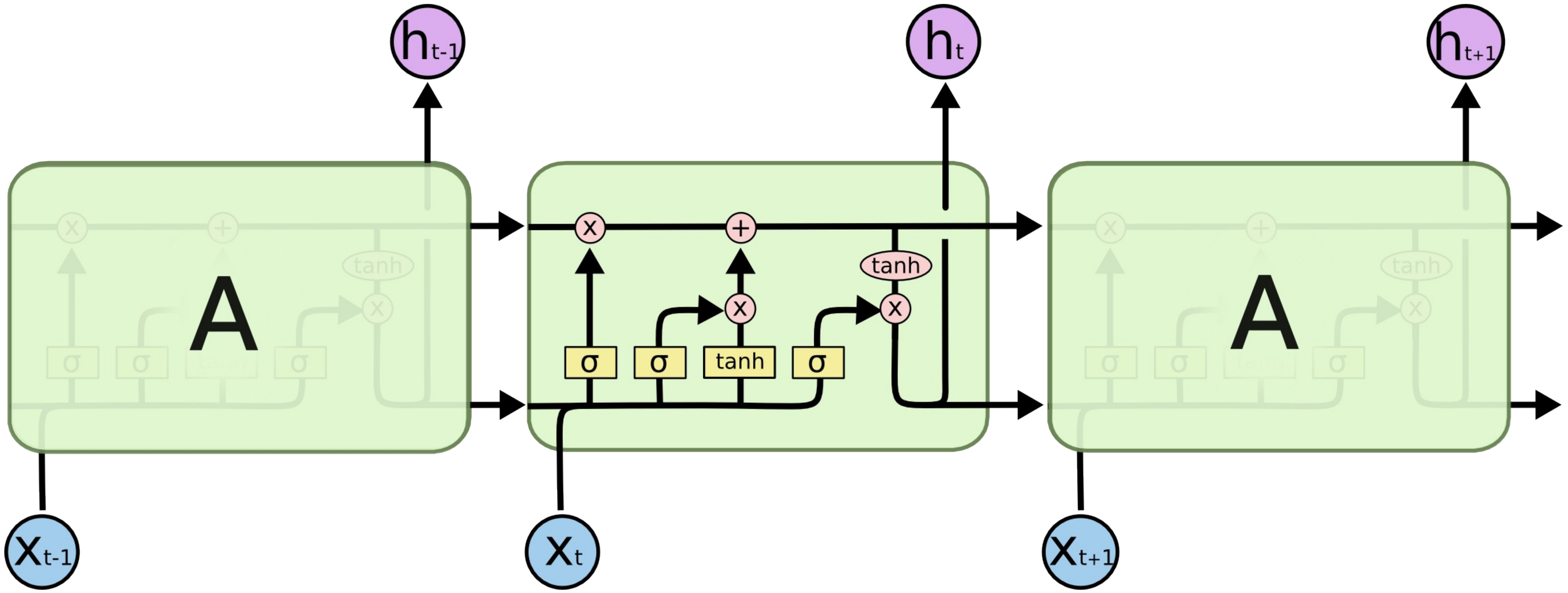
- RNNs have a difficult time with long term dependencies

- Activation functions such as tanh and sigmoid have derivatives of 0 at both ends.

- Gradient contributions from “far away” steps become zero, and the state at those steps doesn’t contribute to what you are learning



LSTM RNNs



LSTM RNN Node Components

Cell State adds or removes information from the LSTM

Multiply the old state with new candidate values

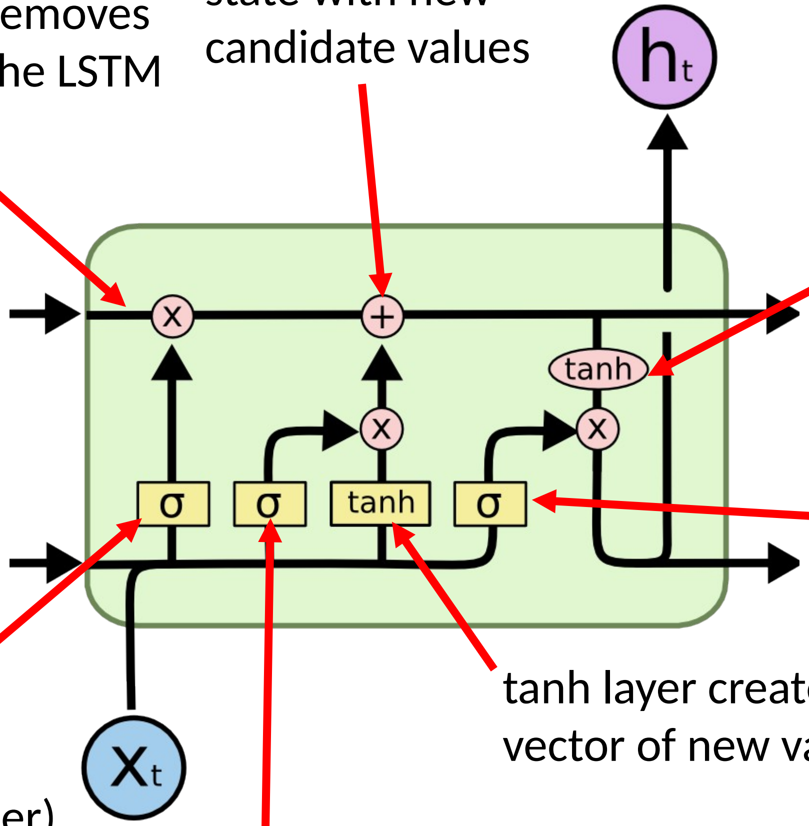
tanh layer multiplies the selected output of previous layers with output of sigmoid gate

Sigmoid layer decides which part of the cell state to output

tanh layer creates a vector of new values

Forget gate includes a sigmoid function that outputs a value between 0 (forget) and 1 (remember)

Input gate includes a sigmoid function that decides which values to update



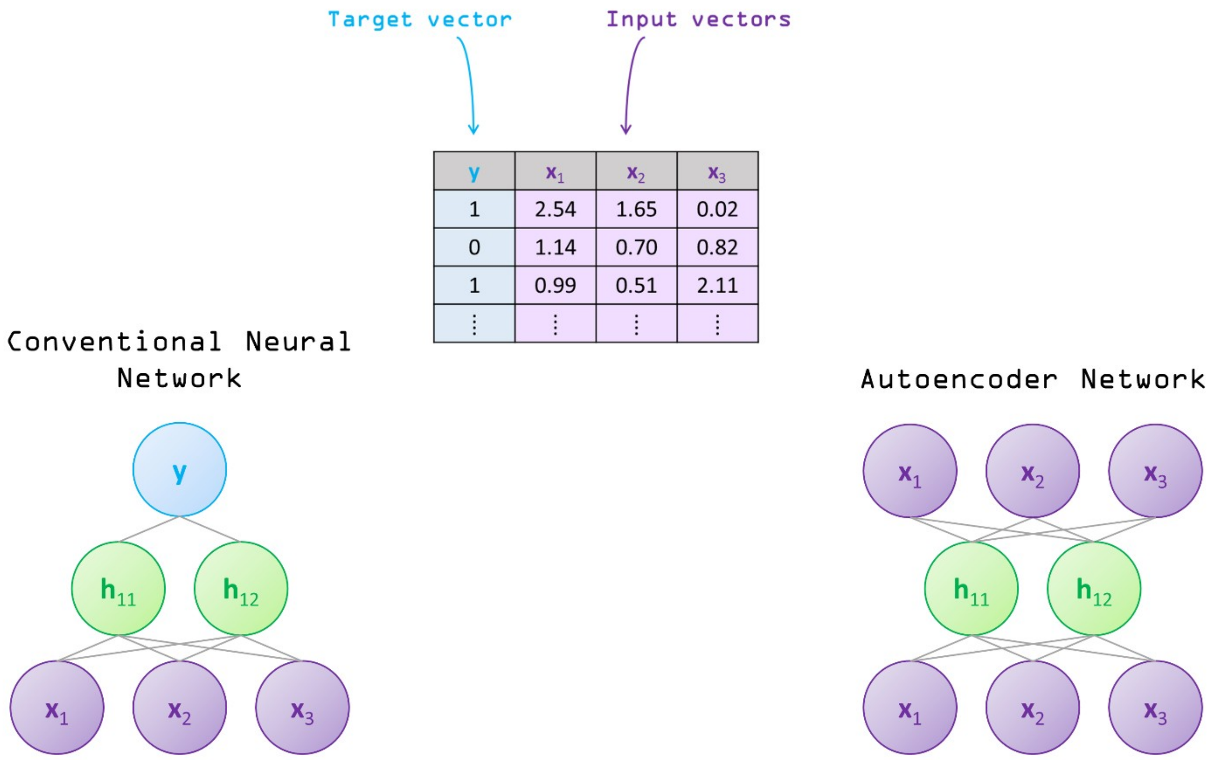
RNN Handwriting Generation Demo

- http://www.cs.toronto.edu/~graves/handwriting.html

Autoencoders for Unsupervised Feature Detection

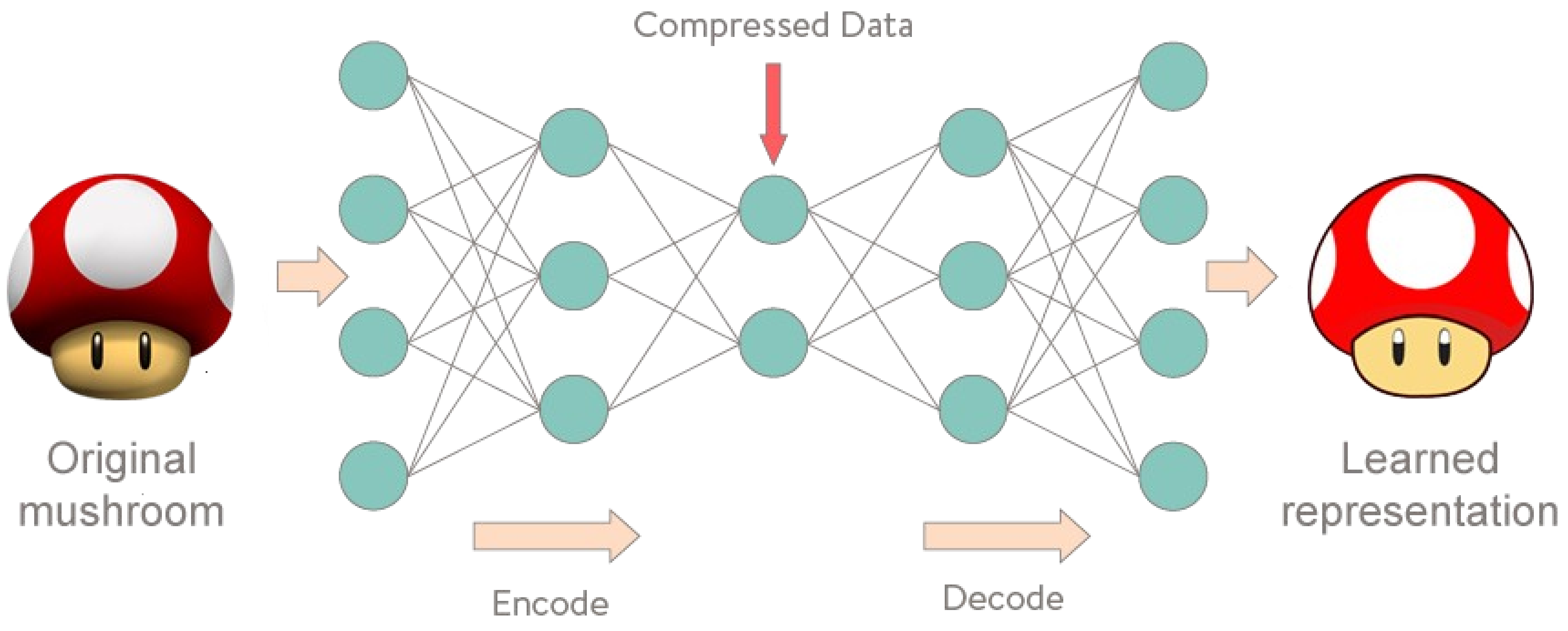
- What about when we don’t have labeled data?
- Neural networks can be used to learn efficient representations or codings of the input data.
- Codings have a much lower dimensionality than input data
- Autoencoders are powerful feature detectors
- Used for pretraining deep neural networks
- Used to generate new data that looks like the input

Autoencoders



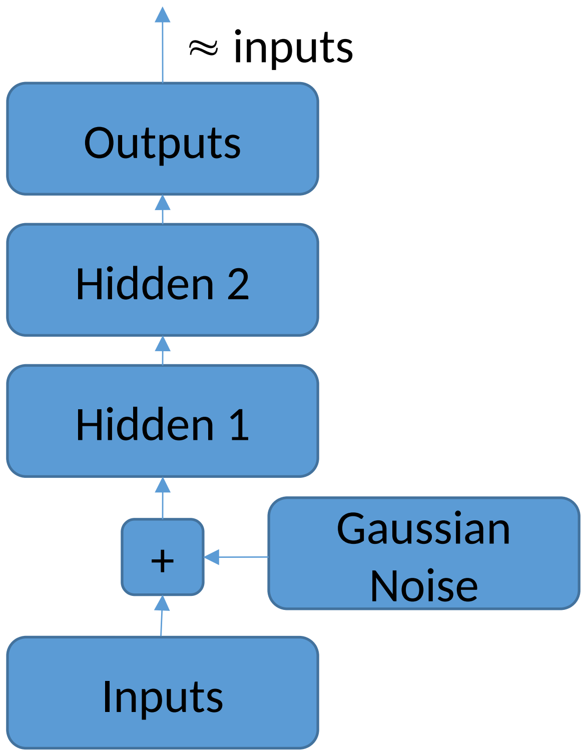
y	x_1	x_2	x_3
1	2.54	1.65	0.02
0	1.14	0.70	0.82
1	0.99	0.51	2.11
⋮	⋮	⋮	⋮

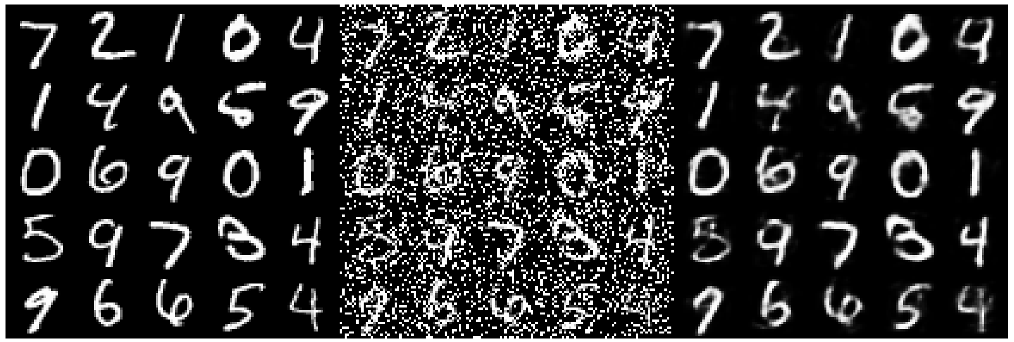
Autoencoders

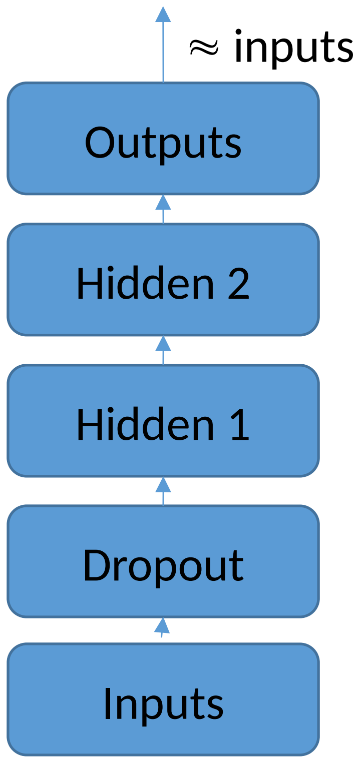


Denoising Autoencoders

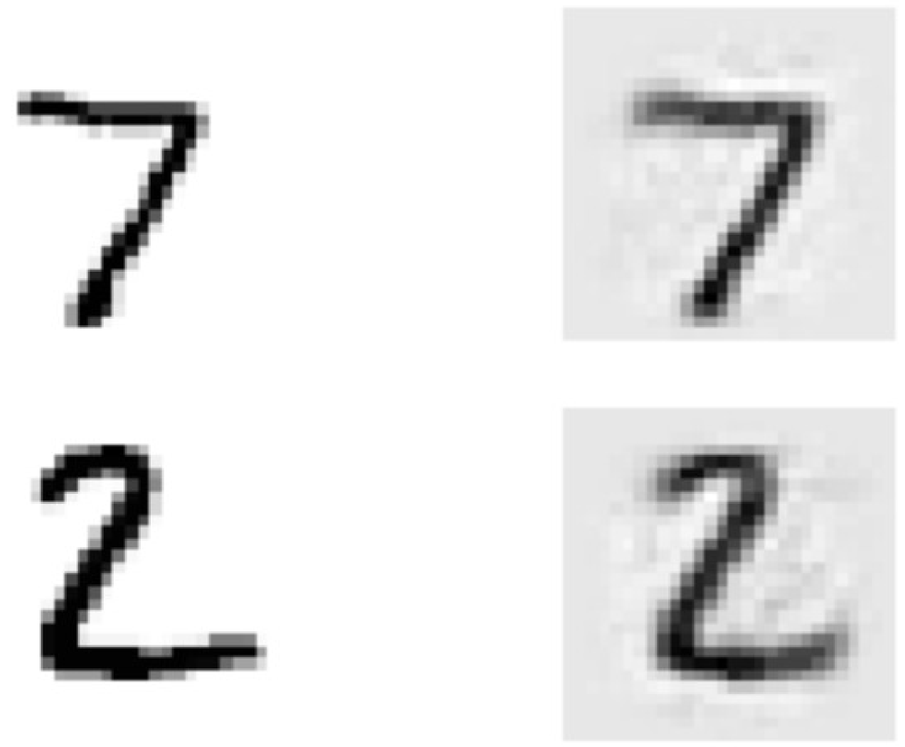
- Denoising autoencoders can be used to learn useful features by adding noise to its inputs

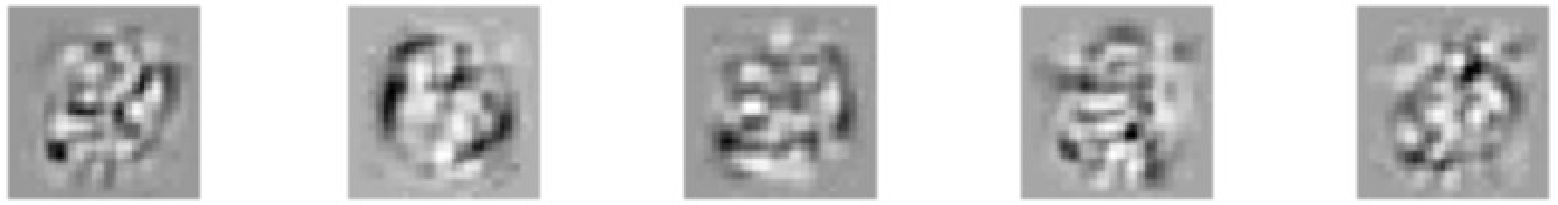






Reconstructed Features and Learned Features



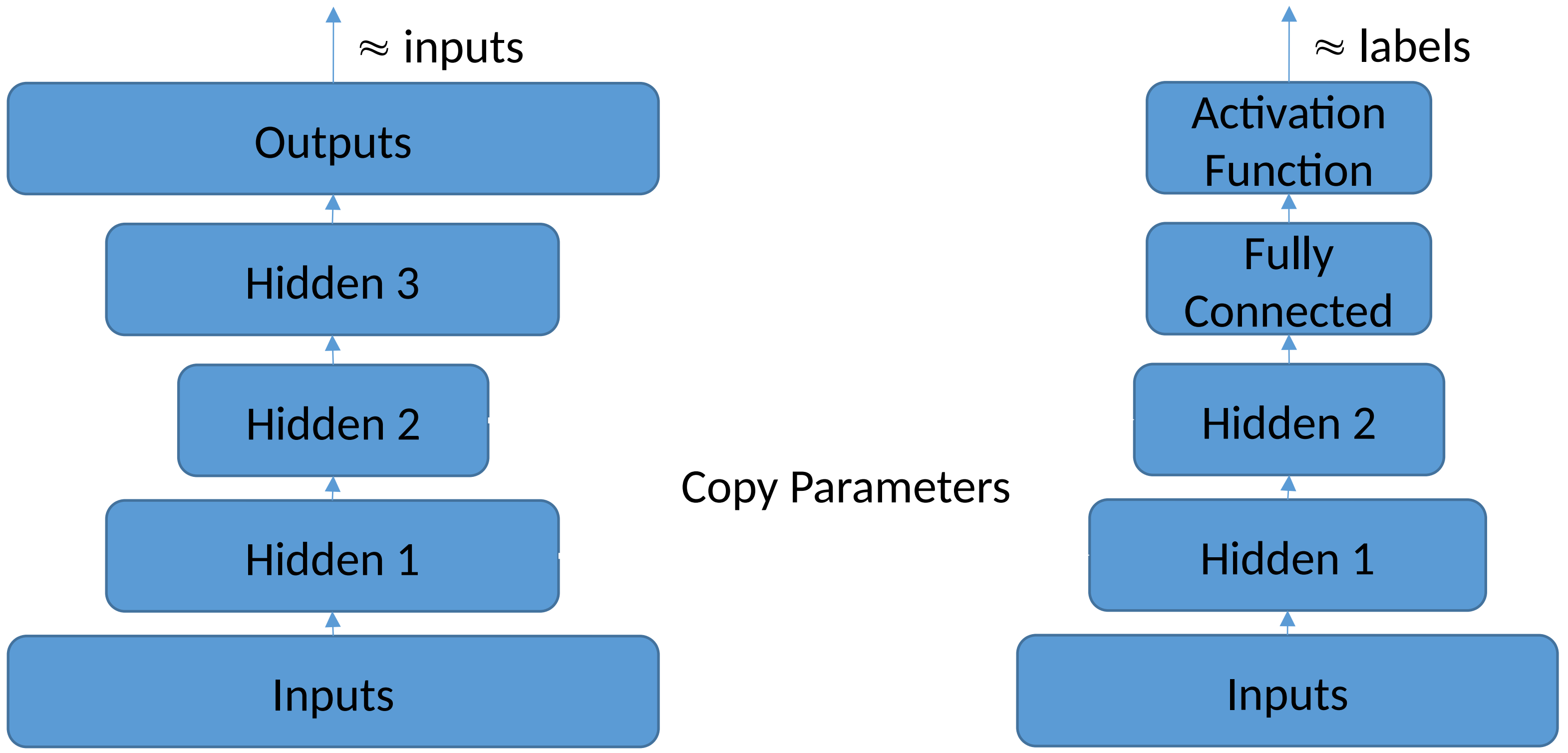


Stacked Denoising Autoencoders

- Uses stacks of autoencoders to learn hidden representations of the data
- As we saw before, denoising autoencoders learn from the data by adding noise and trying to reconstruct it
- Each layer consists of a denoising autoencoder with input from the previous layer’s output
- Trained to minimize reconstruction error in the output

Autoencoders for Unsupervised Pretraining

- Sometimes you have a supervised learning task but limited label data



Phase 1: Train the autoencoder using all the data

Phase 1: Train the autoencoder using all the data