Computational Systems Biology
Deep Learning in the Life Sciences

6.802 6.874 20.390 20.490 HST.506

David Gifford

Lecture 3

February 12, 2019

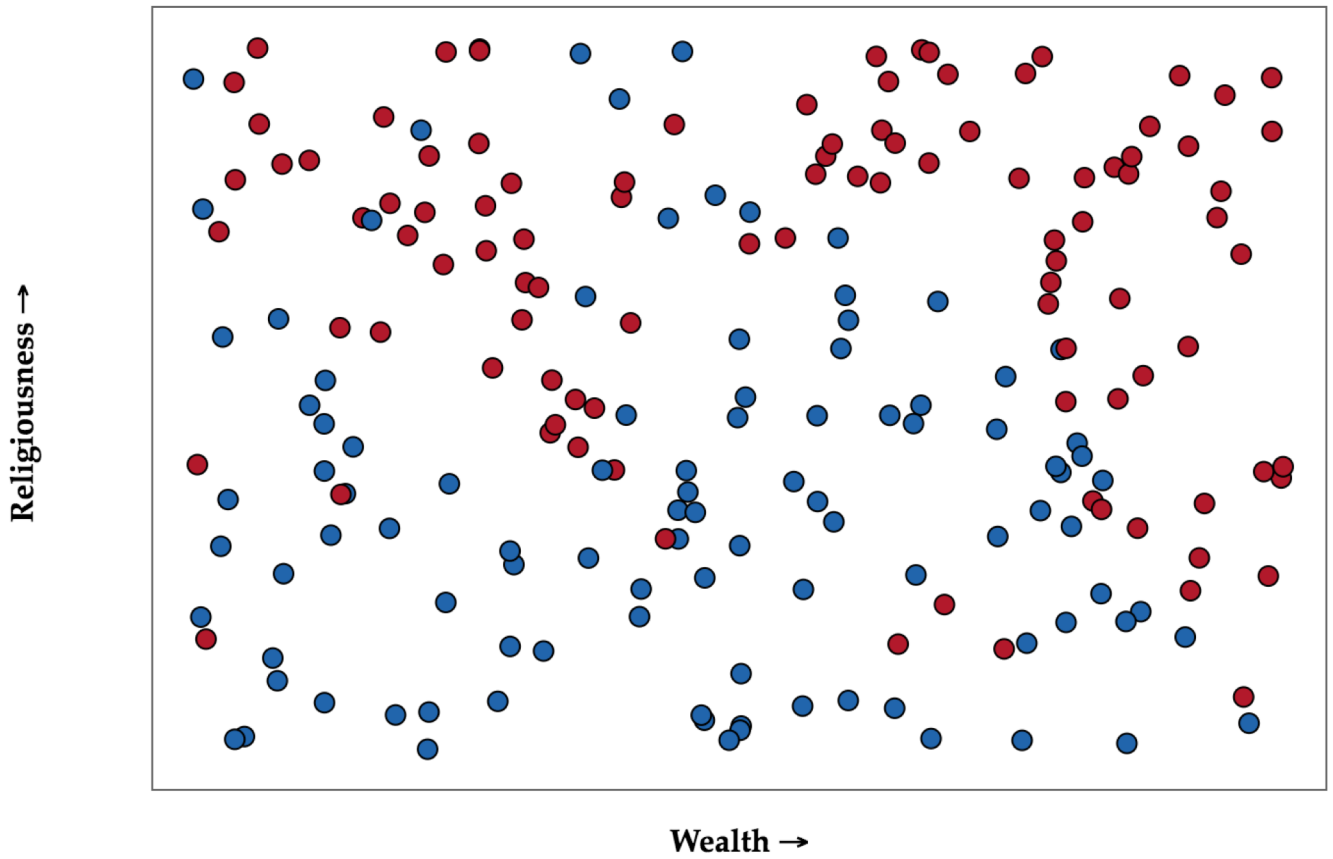
Deep Networks, Convolutional Networks, and Recurrent Networks

http://mit6874.github.io

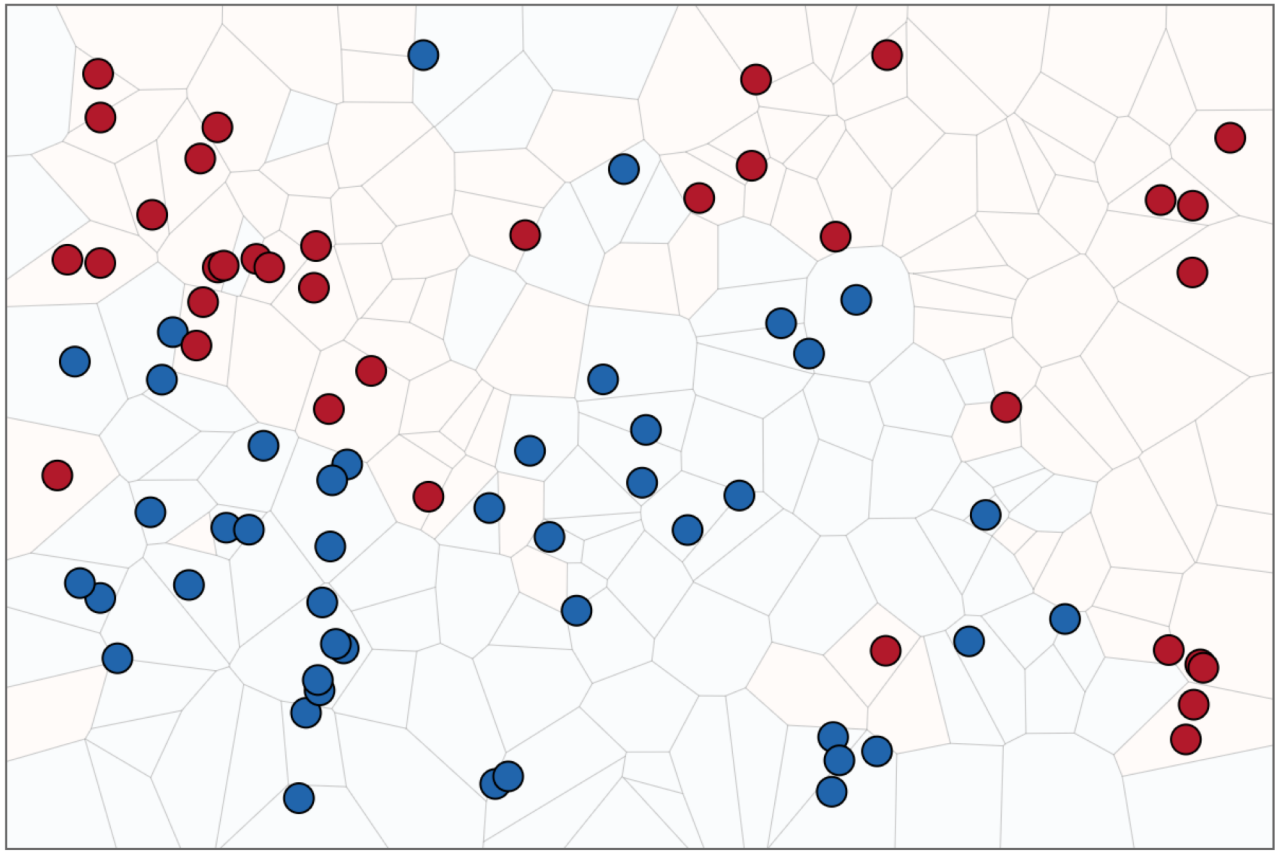
What's on tap today!

- Model Capacity
- K-nearest neighbor classifiers
- Convolutional Networks
  - Dilated networks
- Recurrent neural networks
  - LSTMs

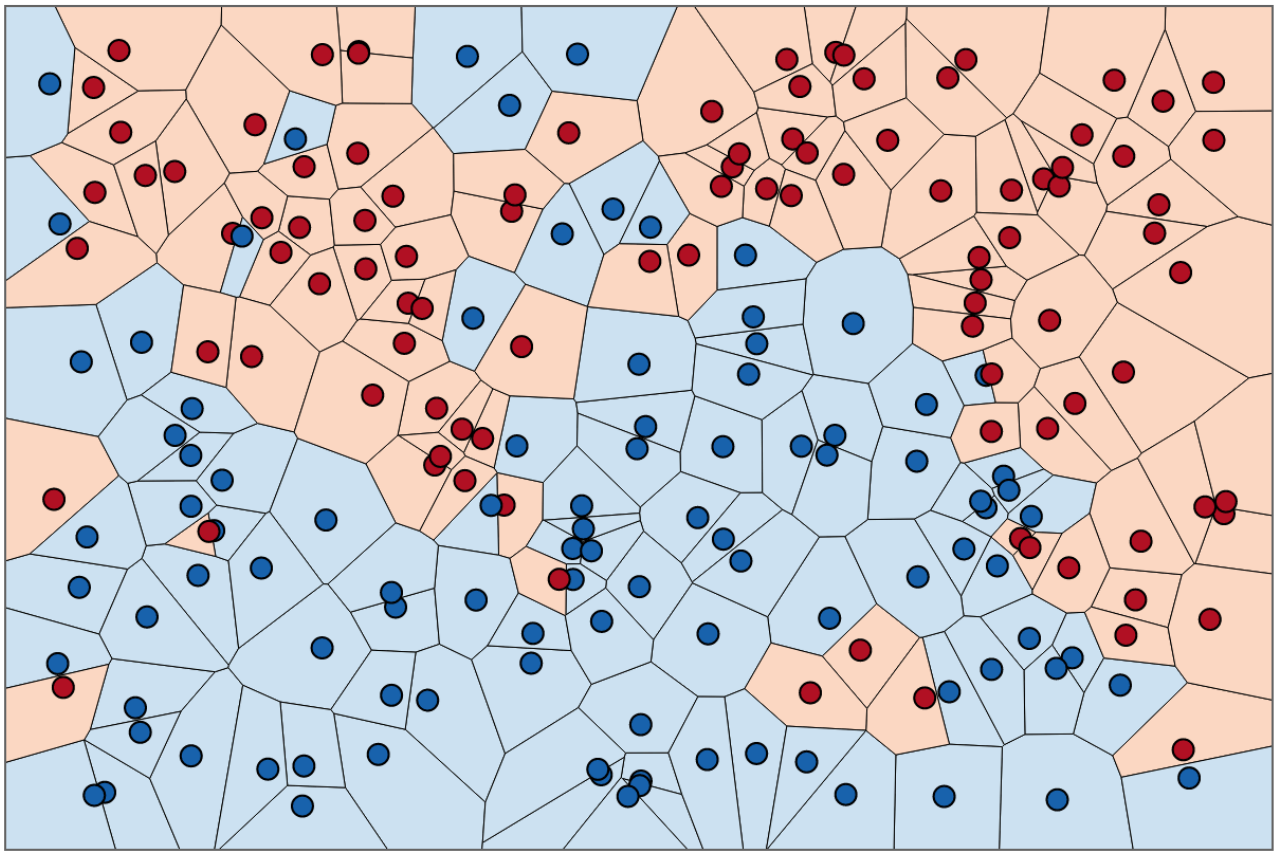
Training set: observations of wealth and religiousness features with #blue or #red labels



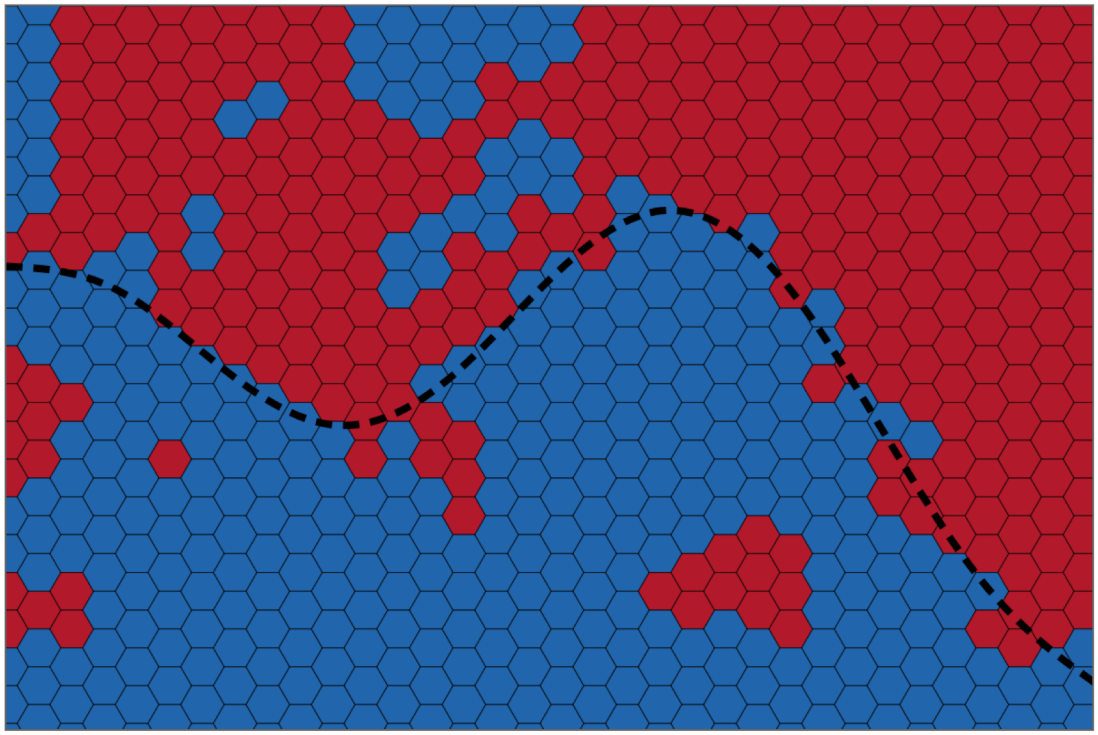
Test set: How well can we do?



K-Nearest Neighbors (KNN) defines neighborhoods

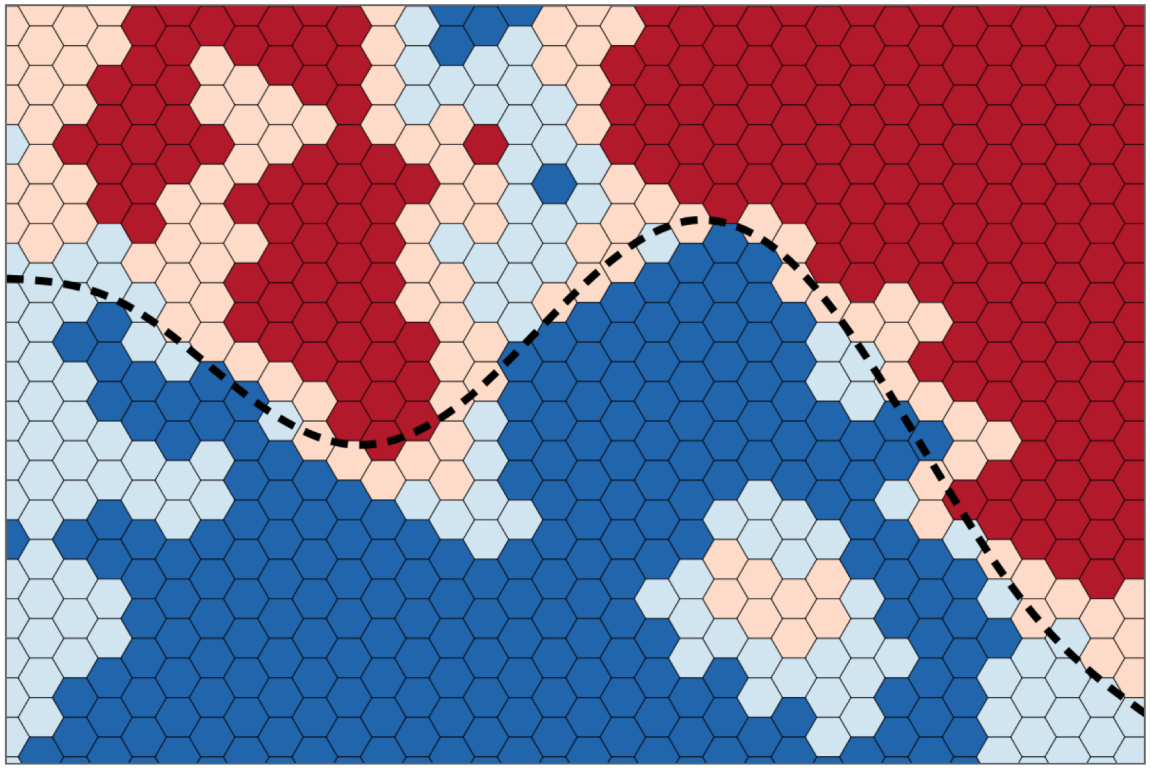


K-Nearest Neighbors (KNN) k=1 classification results

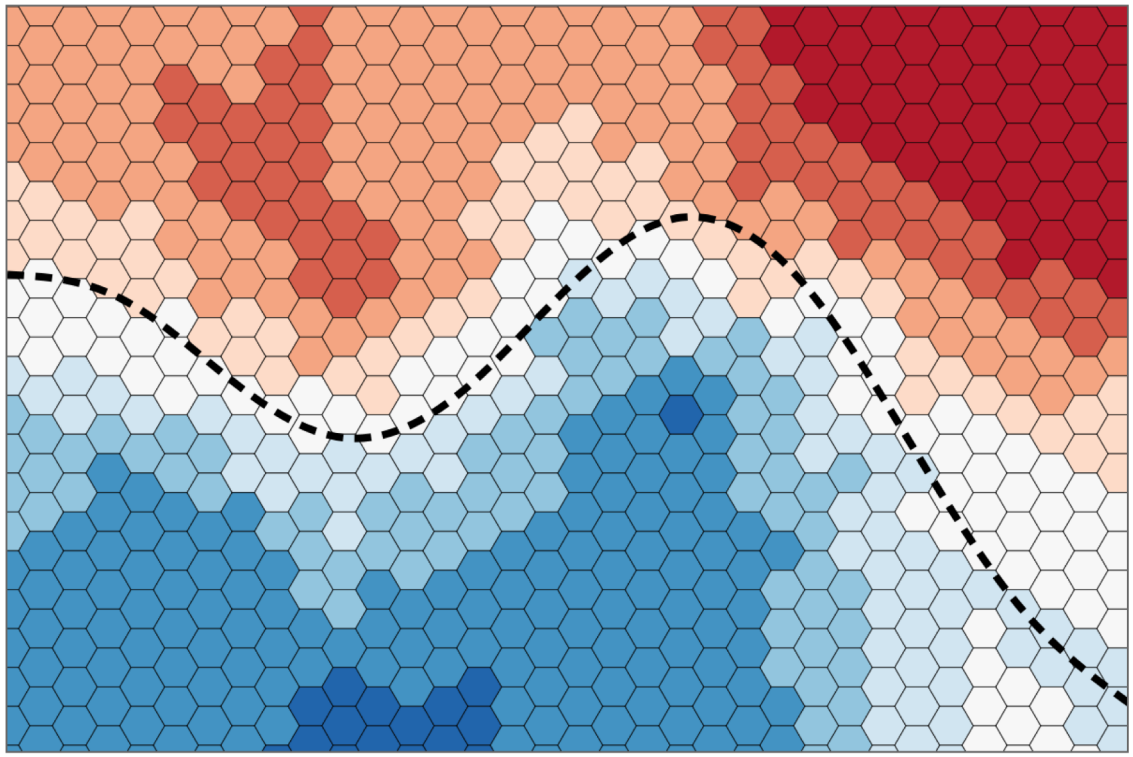


k-Nearest Neighbors: 1

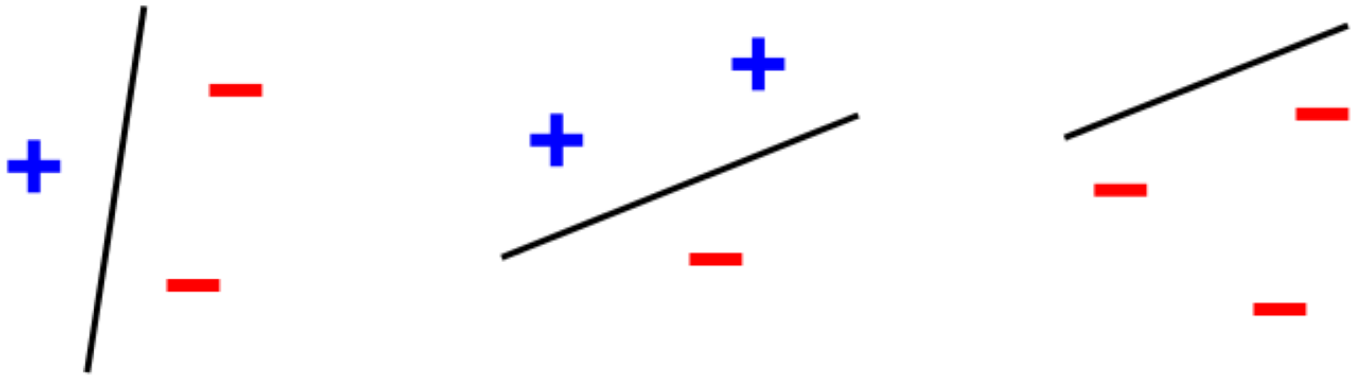
K-Nearest Neighbors (KNN) k=3 classification results



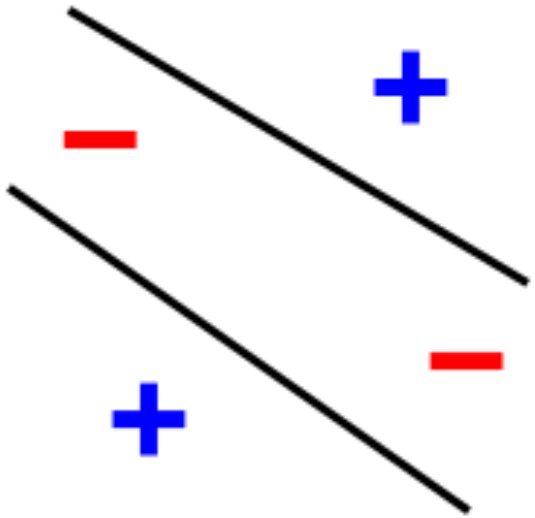
K-Nearest Neighbors (KNN) k=29 classification results



A straight line can classify three points arbitrarily labeled



A straight line can not classify four points arbitrarily labeled



The **capacity** (Vapnik-Chervonenkis dimension) of a model describes how many points can be correctly predicted when they are produced by an adversary

The capacity of non-parametric models is defined by the size of their training set

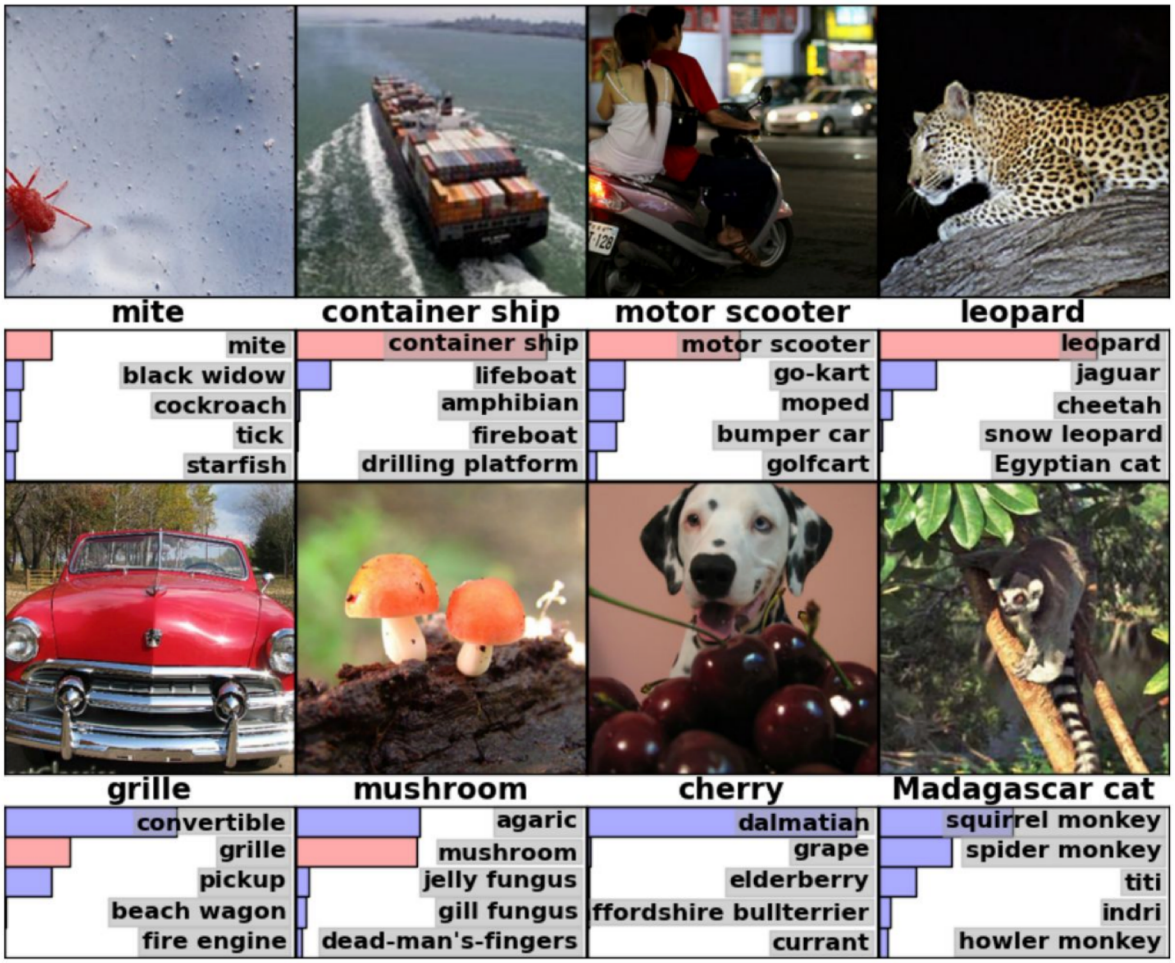
- k-nearest neighbor (KNN) regression computes its output based upon the k “nearest” training examples
- Often the best method, and certainly a baseline to beat

The **generalizability** of a model describes its ability to perform well on previously unseen inputs

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.



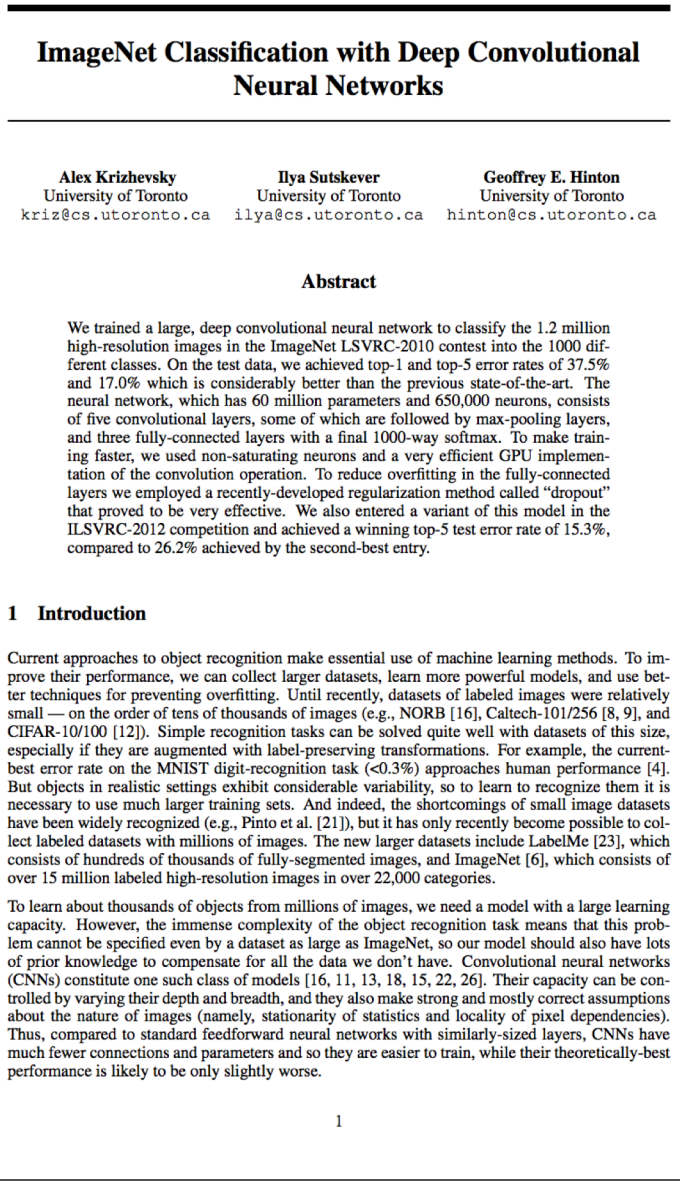
ConvNets changed the world in 2012

ImageNet Large Scale Visual Recognition Challenge: Predict top-5 classes in an image and provide bounding boxes. Training set: 1.2M images with 1000 classes. Validation/test set: 150K.

AlexNet scored top-5 error of 15.3%; 10.% lower than runner up.

One of the early applications of convolutional neural networks (CNNs) and Graphics Processing Units (GPUs)

World started paying attention



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky University of Toronto kriz@cs.utoronto.ca
Ilya Sutskever University of Toronto ilya@cs.utoronto.ca
Geoffrey E. Hinton University of Toronto hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don't have. Convolutional neural networks (CNNs) constitute one such class of models [16, 11, 13, 18, 15, 22, 26]. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse.

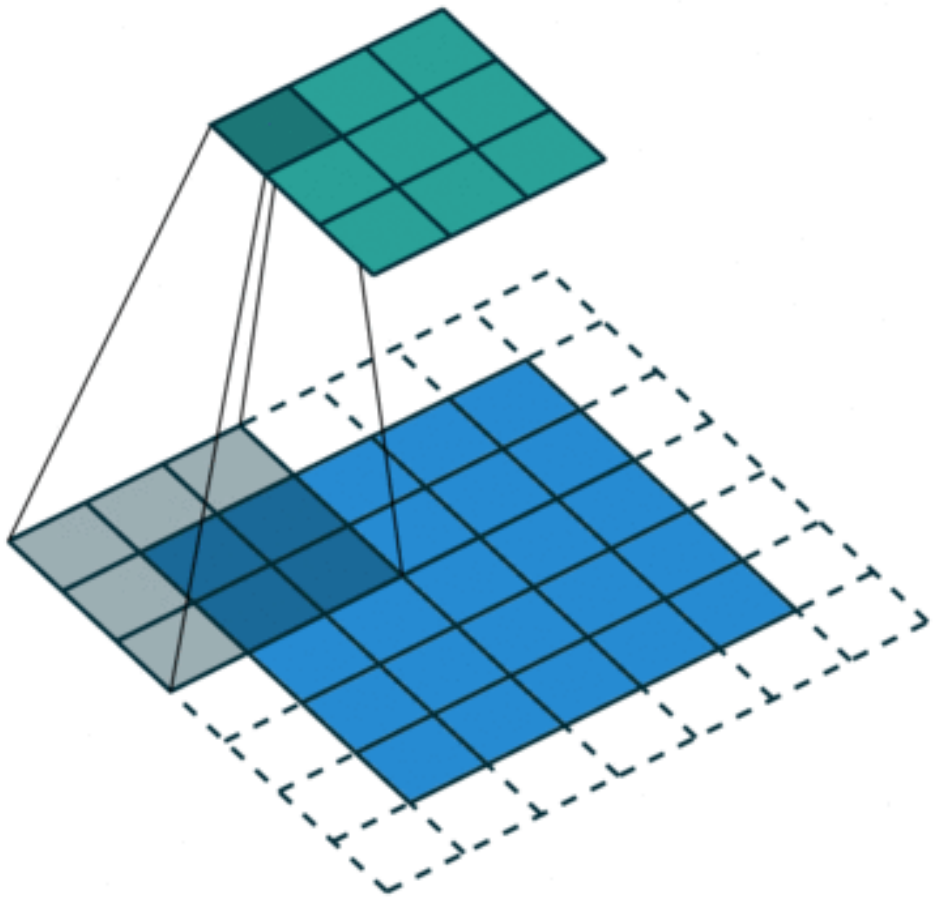
1

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

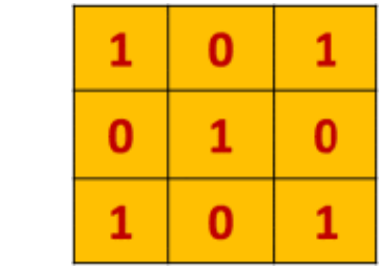
Convolutional Filters learn patterns

When patterns are not aligned we need a way to recognize them

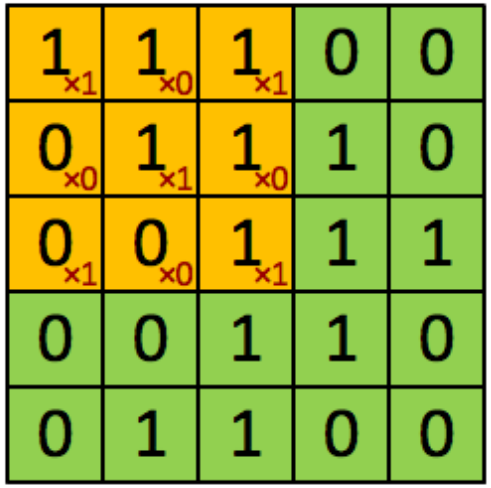
Convolution shares parameters
Example 3x3 convolution on a 5x5 image



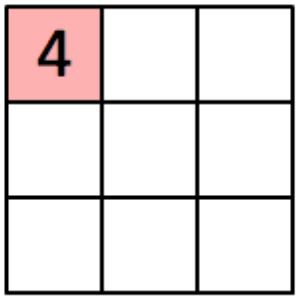
Convolution operation is element wise multiply and add



Filter / Kernel



Image



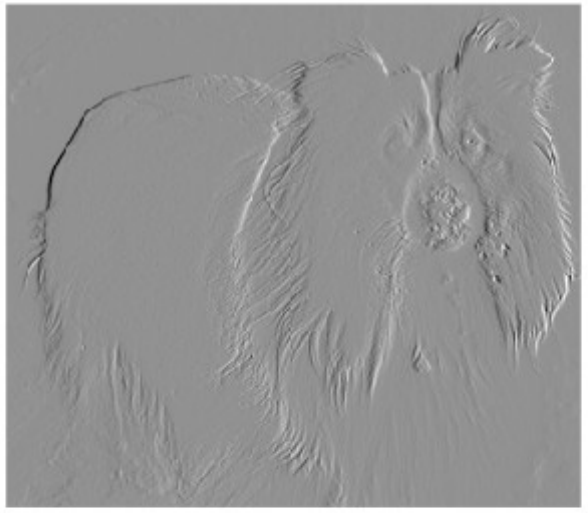
Convolved Feature

A simple pattern: Edges
How can we detect edges with a kernel?

Input

A simple pattern: Edges
How can we detect edges with a kernel?

Input

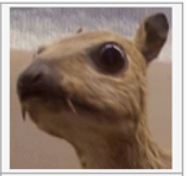
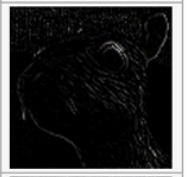
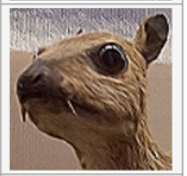
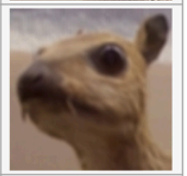
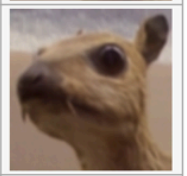


Output

1	-1

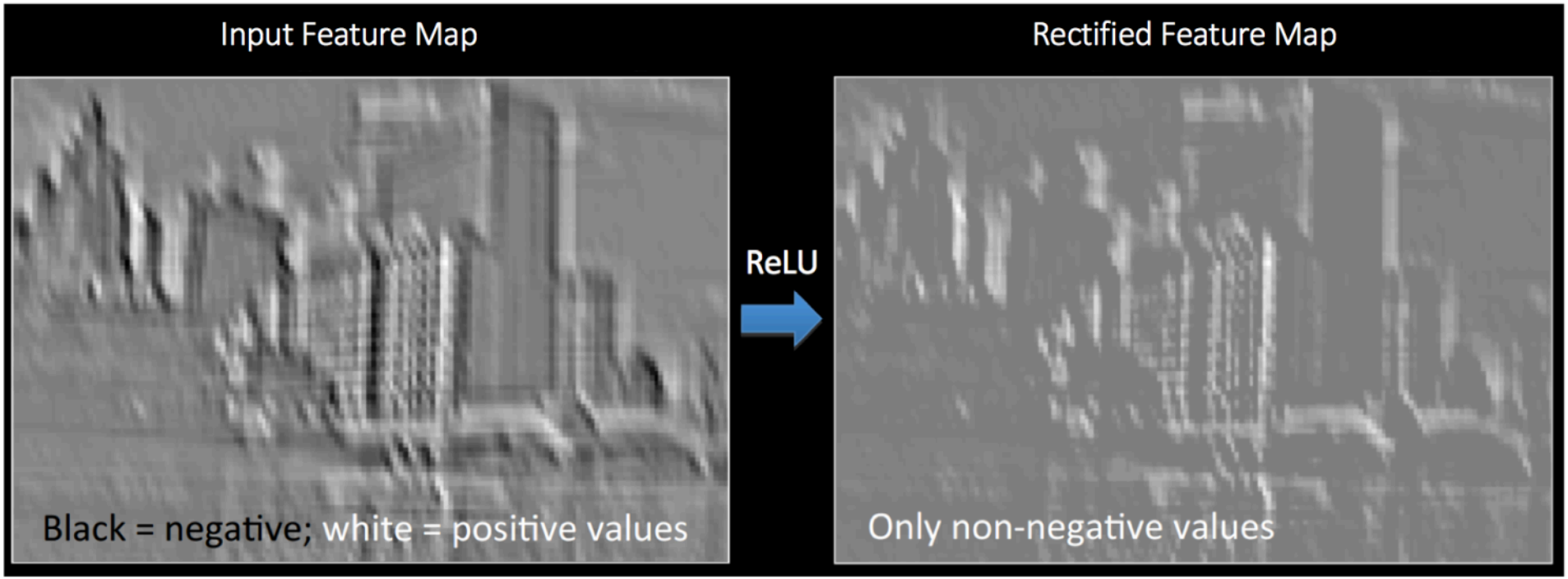
Filter

Simple Kernels / Filters

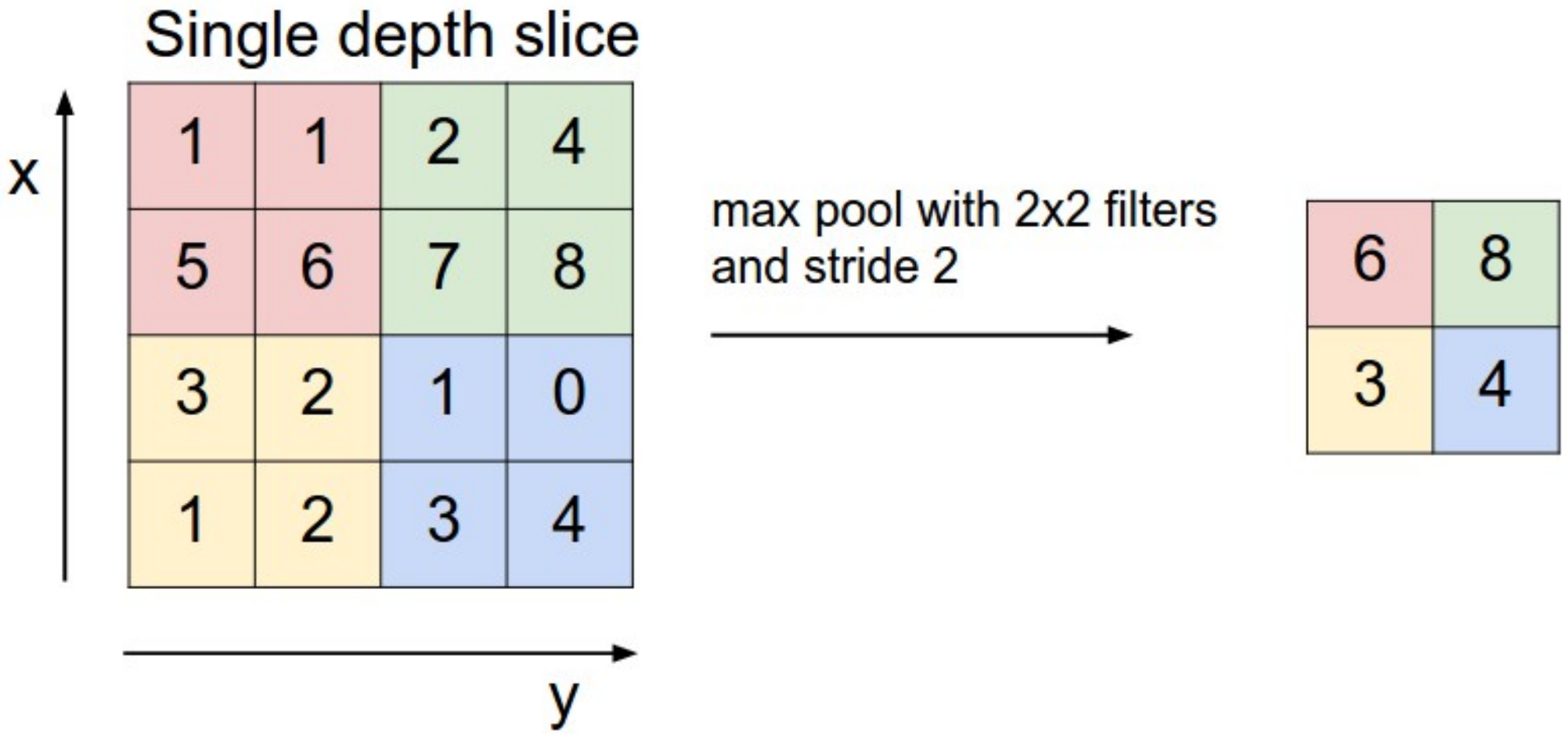
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
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)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Convolution of an image

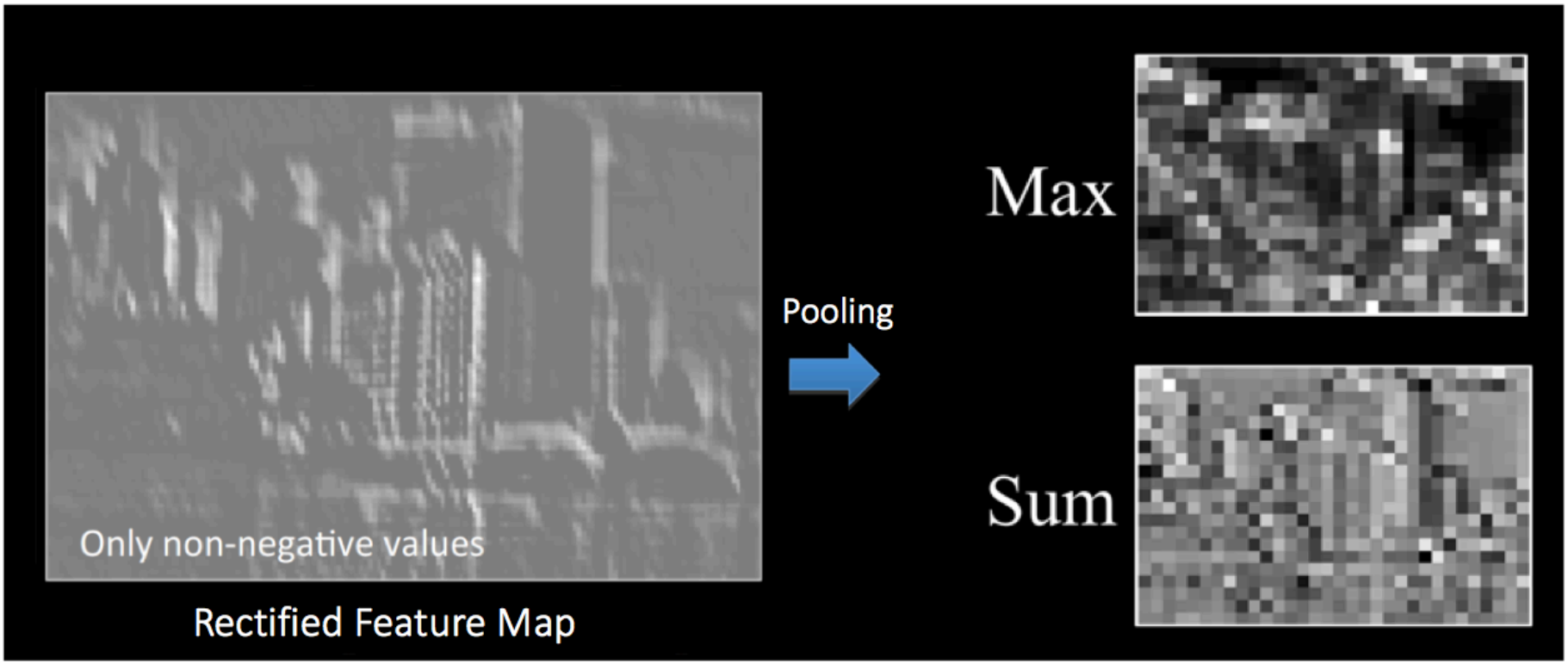
Feature map after RELU



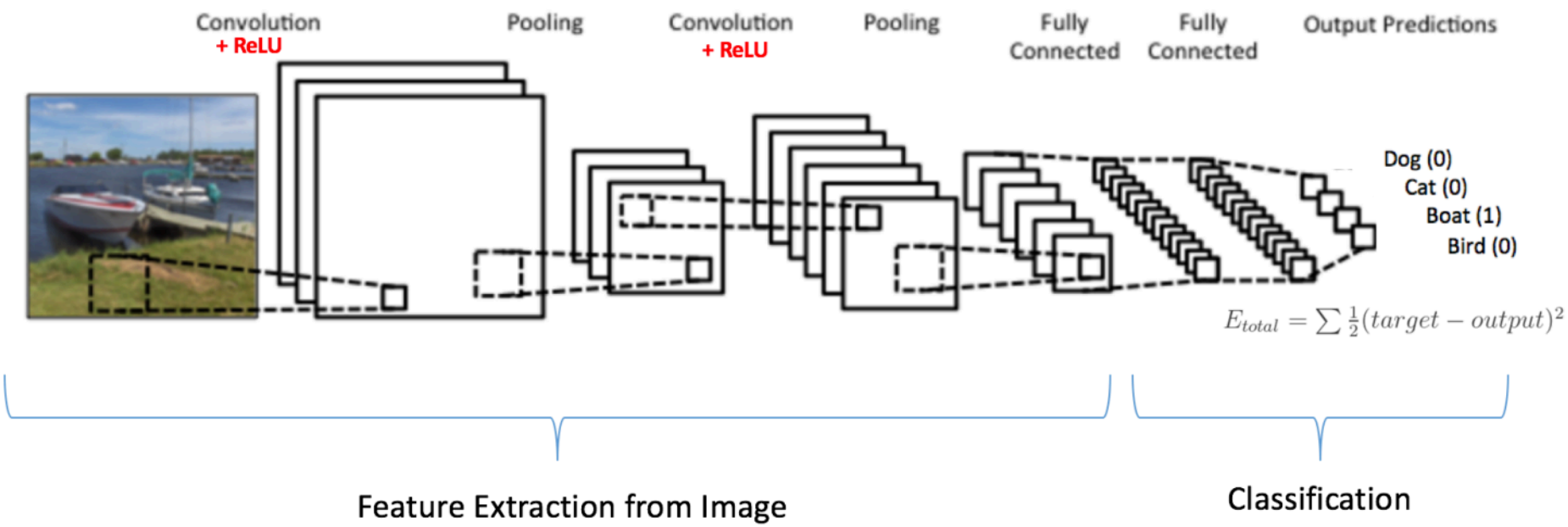
Max pooling reduces resolution with OR like operator



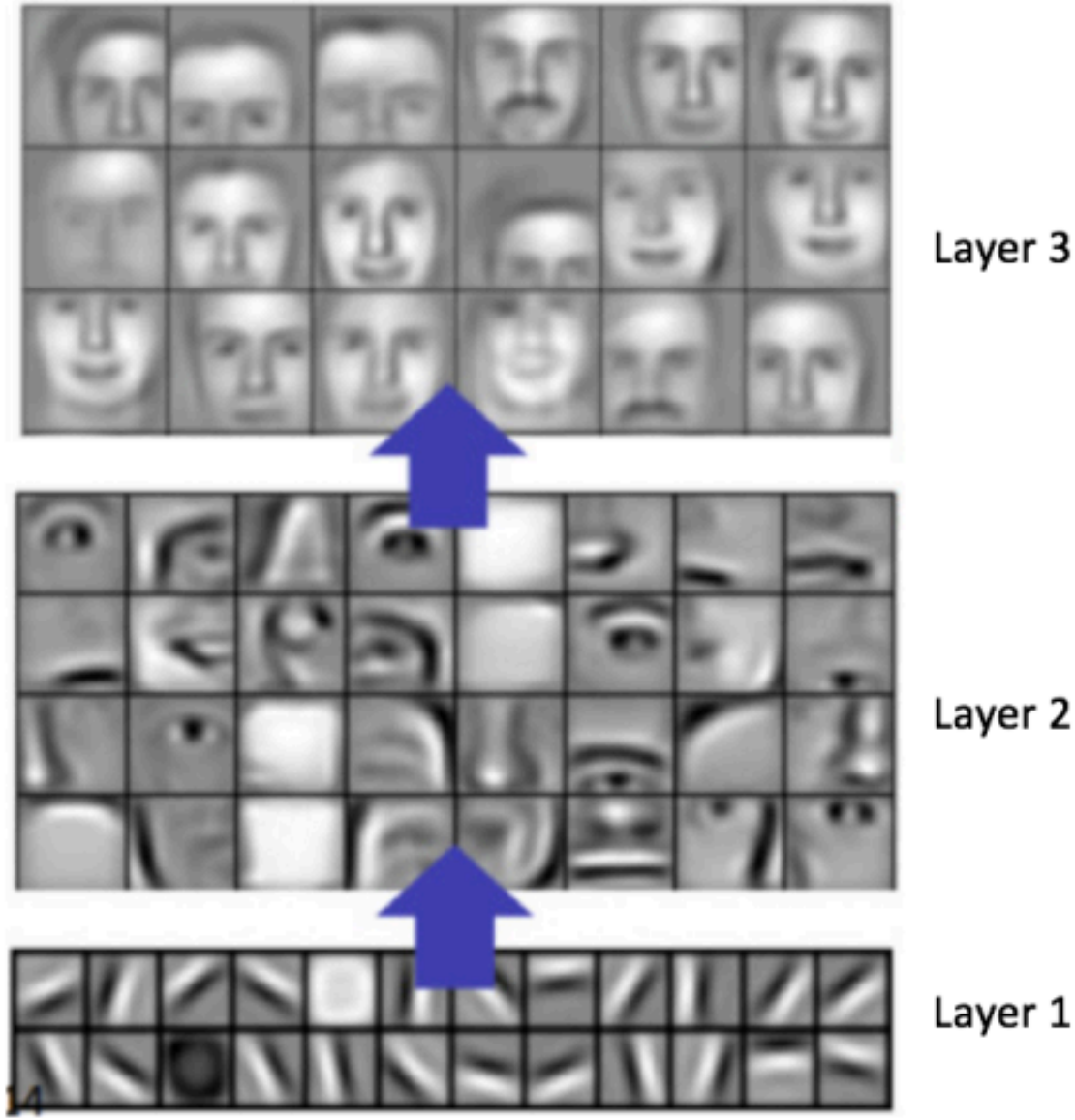
Pooling summarizes key features



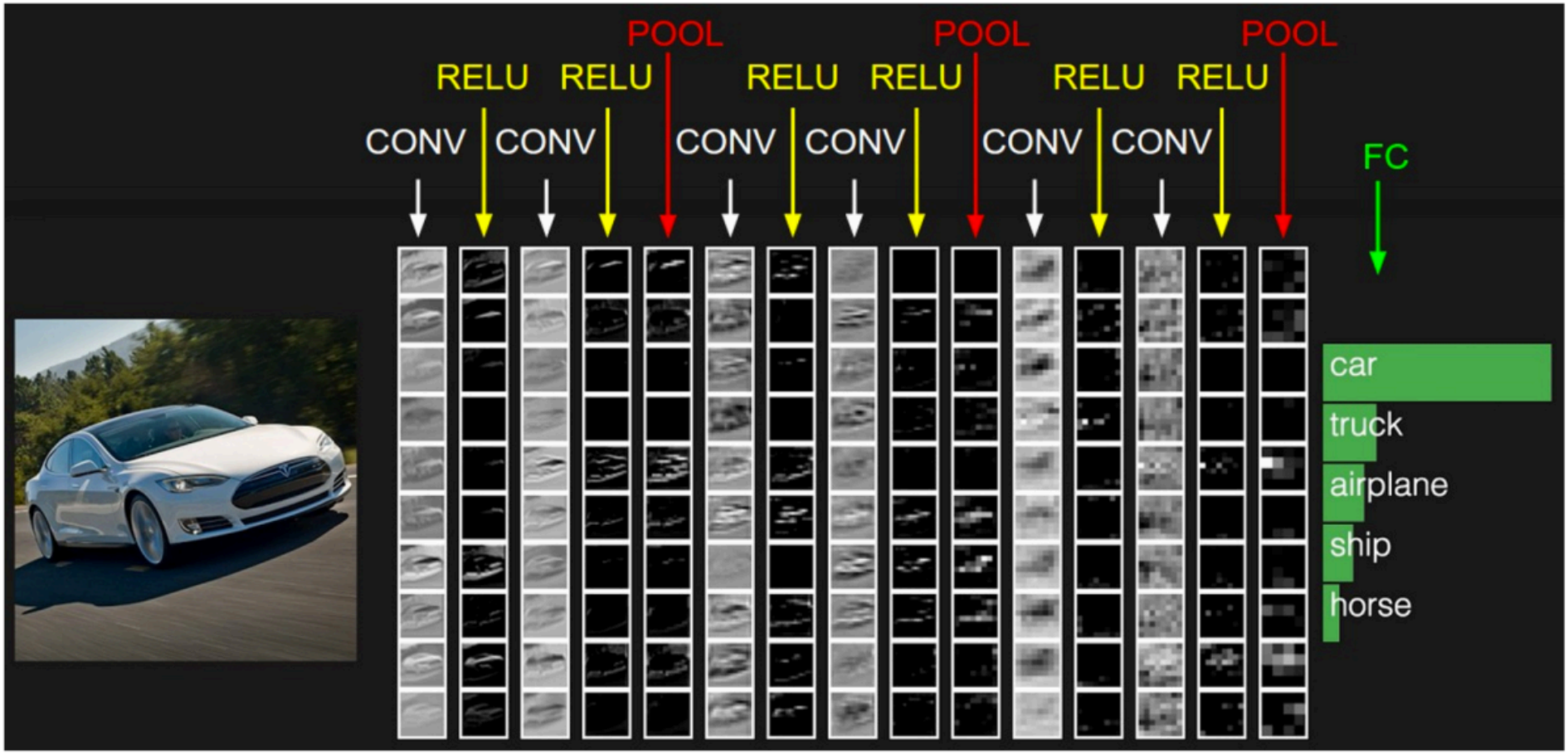
An image classification CNN



Subsequent convolutional layers capture higher levels of abstraction



Example – Six convolutional layers

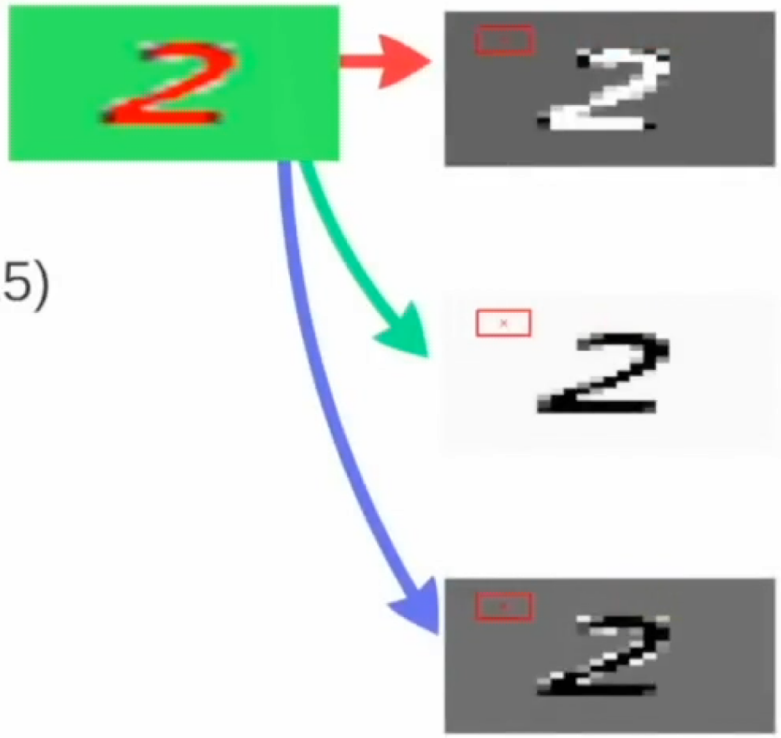


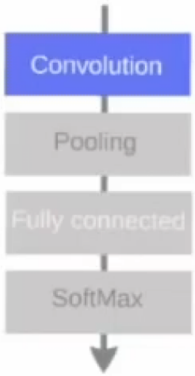
CNN - 1st Convolution

input:
- 3 channels image (25 x 25)

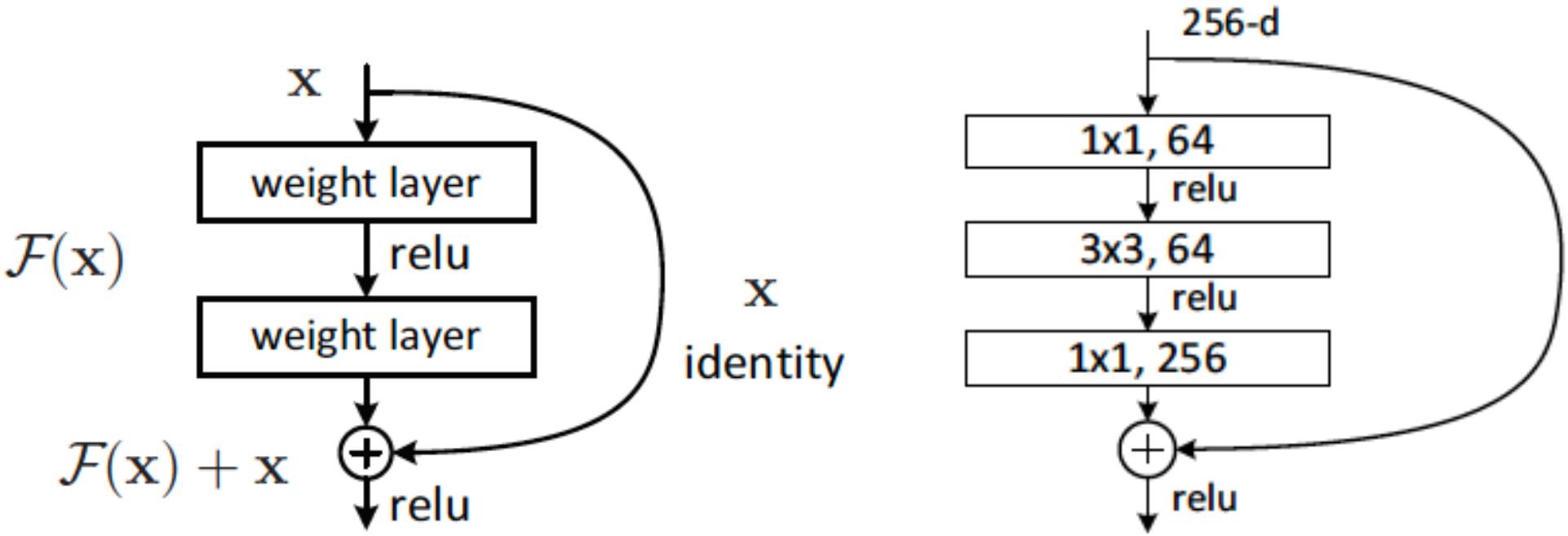
convolution:
- 2 filters 5x5
- ReLU function

output:
- 2 tensors (25 x 25)



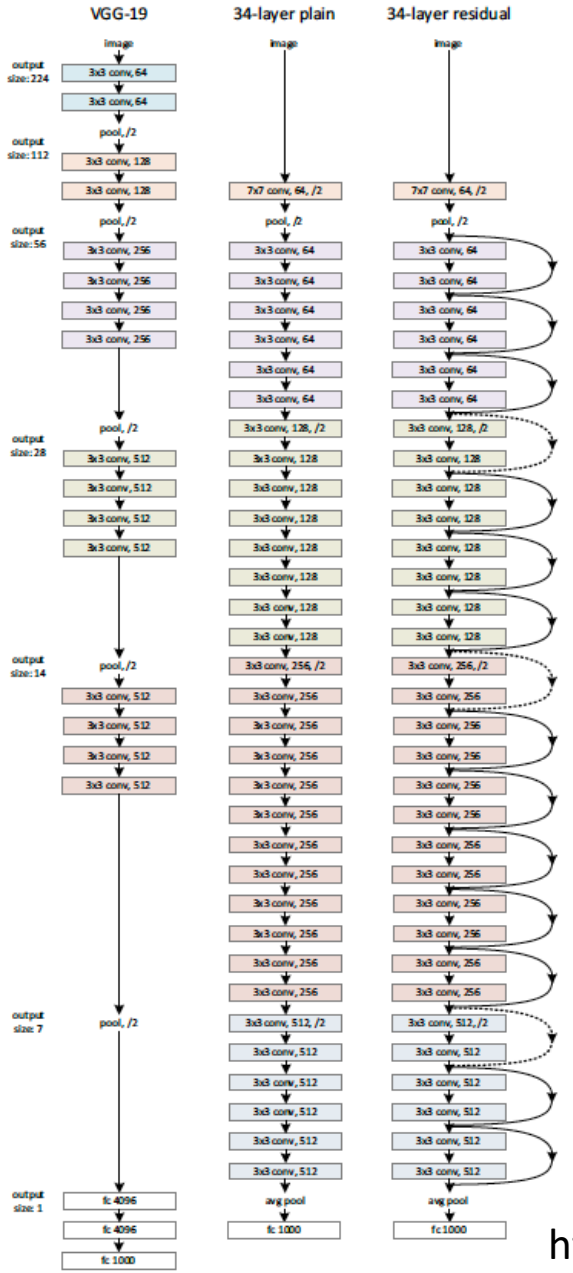


Residual networks permit effective training of deeper networks with “shortcut connections” “Res Nets”



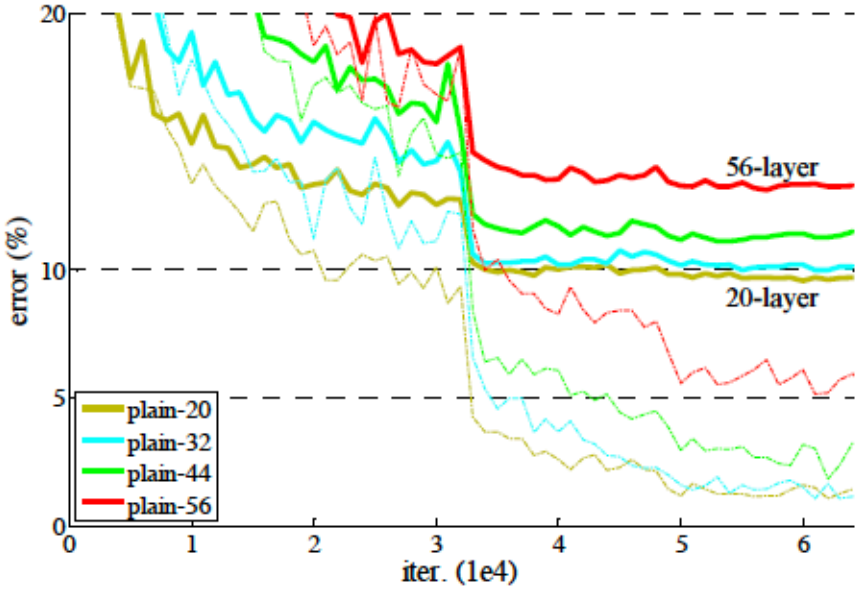
https://arxiv.org/pdf/1512.03385.pdf

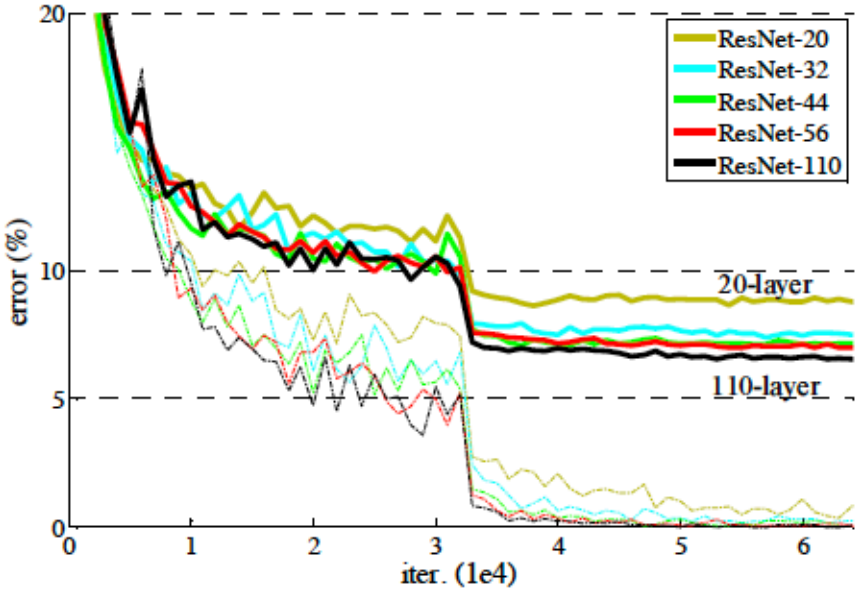
A ResNet won the 2015 ILSVRC compeition



https://arxiv.org/pdf/1512.03385.pdf

Residual networks have superior performance on certain tasks (CIFAR 10 shown here)





https://arxiv.org/pdf/1512.03385.pdf

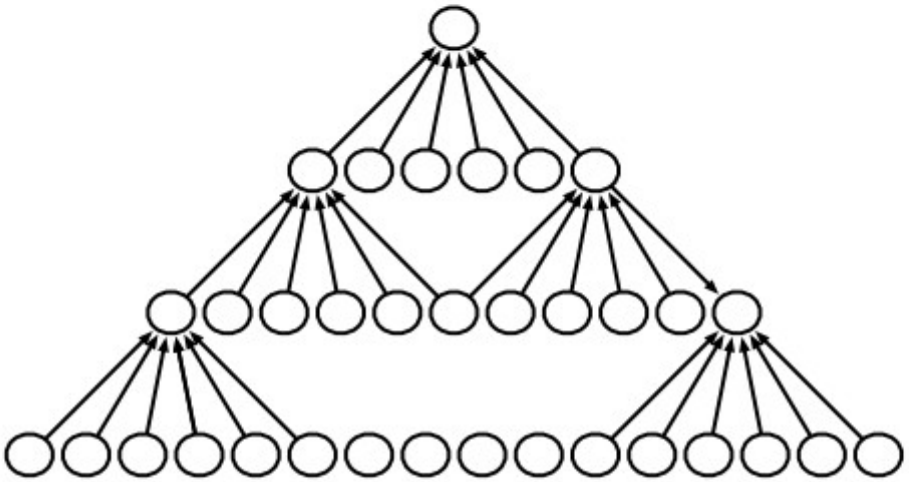
There are three approaches to edge cases in convolution

- **Valid convolution**: output only when entire kernel is contained in input (shrinks output)
- **Same convolution**: zero pad input so output is same size as input dimensions
- **Full convolution**: zero pad input so output is produced whenever an output value contains at least one input value (expands output)

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n).$$

Zero Padding Controls Output Size

Valid -
no padding

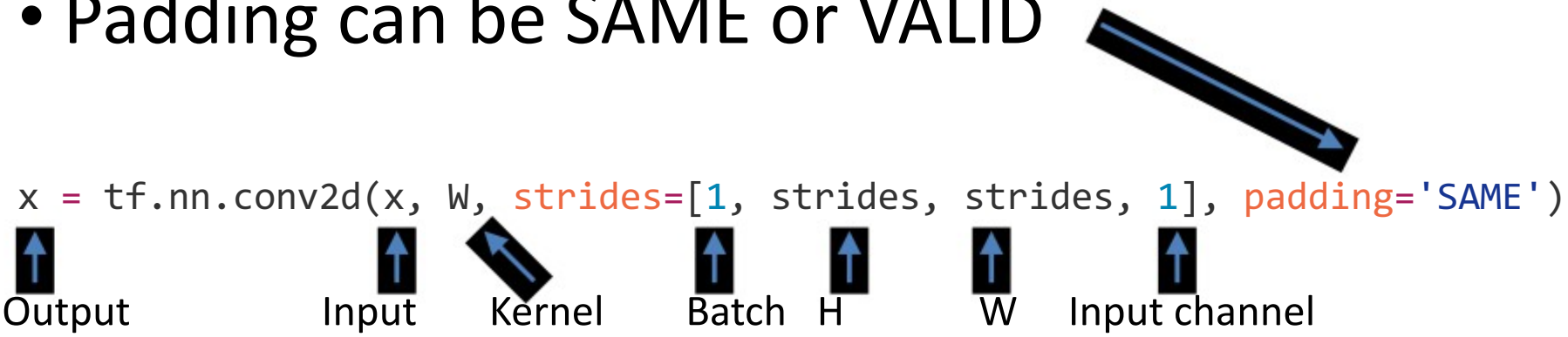


Same -
add zeros

Figure 9.13

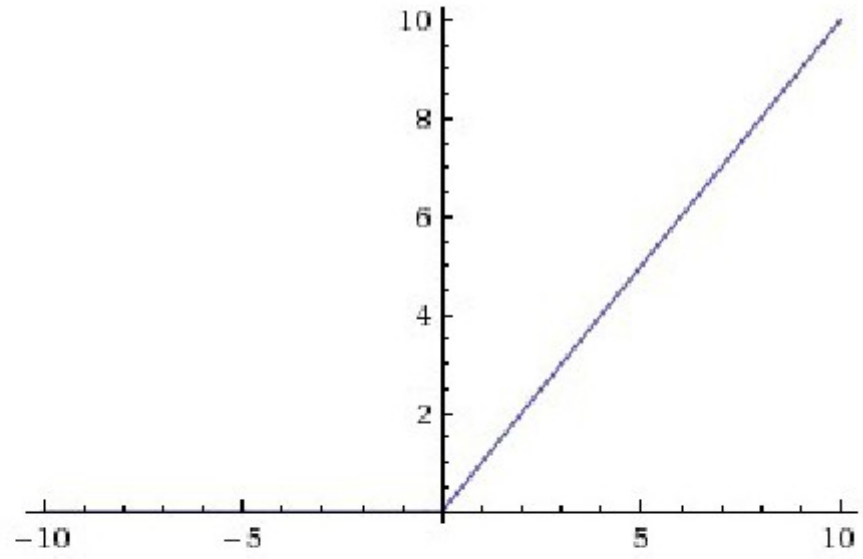
TF convolution operator takes stride and zero fill option as parameters

- Stride is distance between kernel applications in each dimension
- Padding can be SAME or VALID



The REctified Linear Unit (RELU) is a common non-linear **detector** stage after convolution

```
x = tf.nn.conv2d(x, W, strides=[1, strides, strides, 1], padding='SAME')
x = tf.nn.bias_add(x, b)
x= tf.nn.relu(x)
```



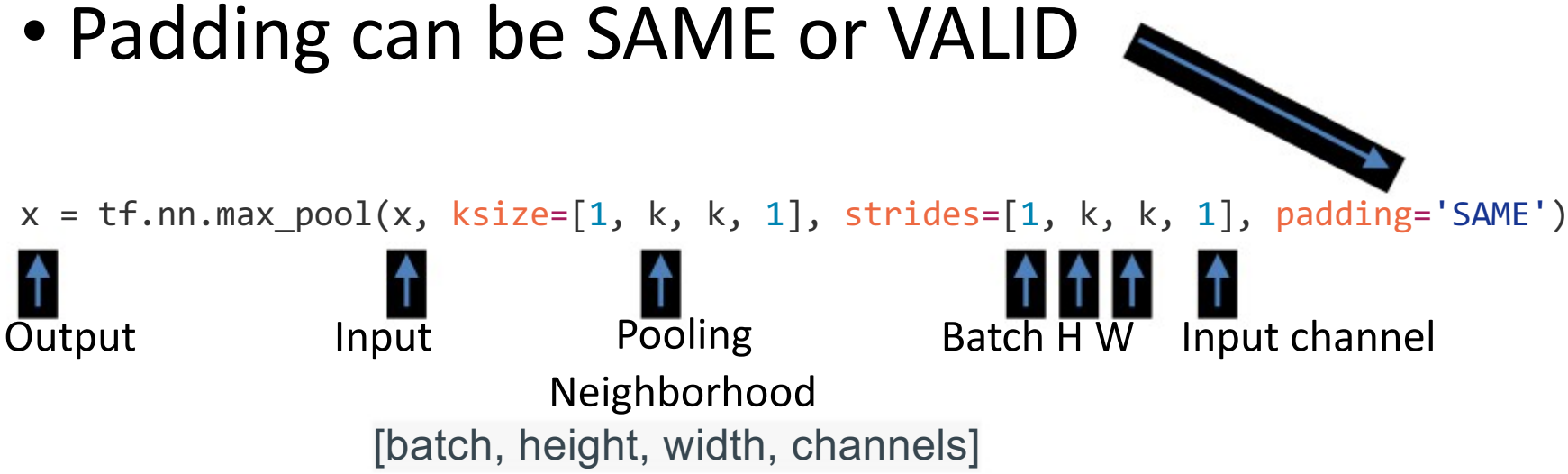
f(x) = max(0, x)

When will we backpropagate through this?

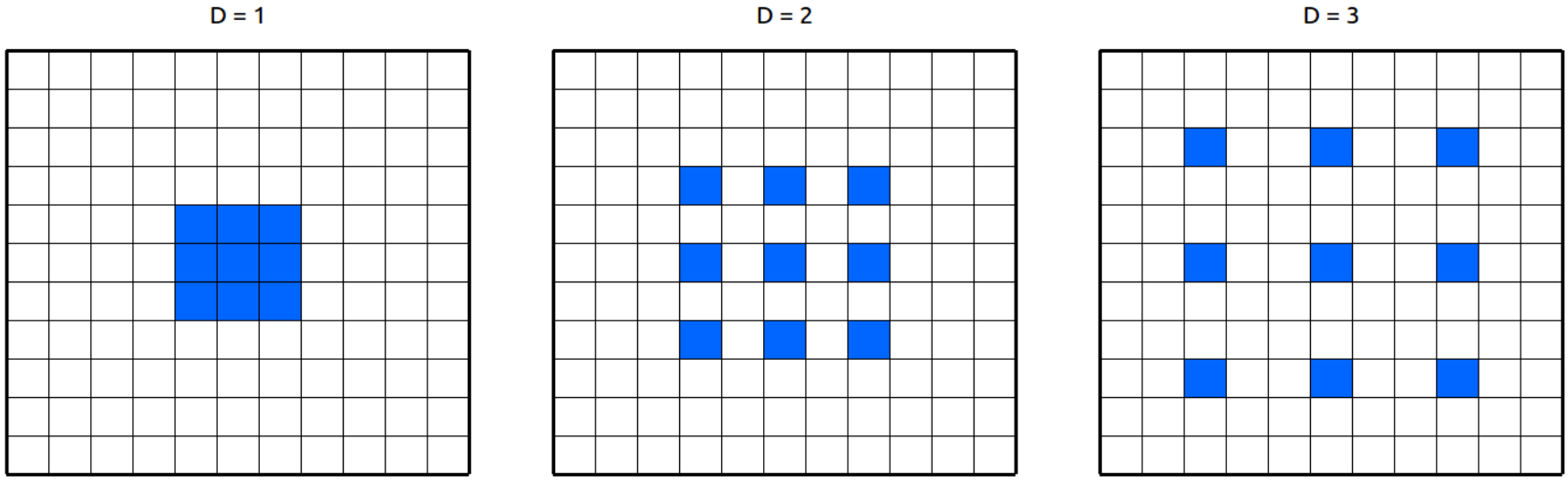
Once it “dies” what happens to it?

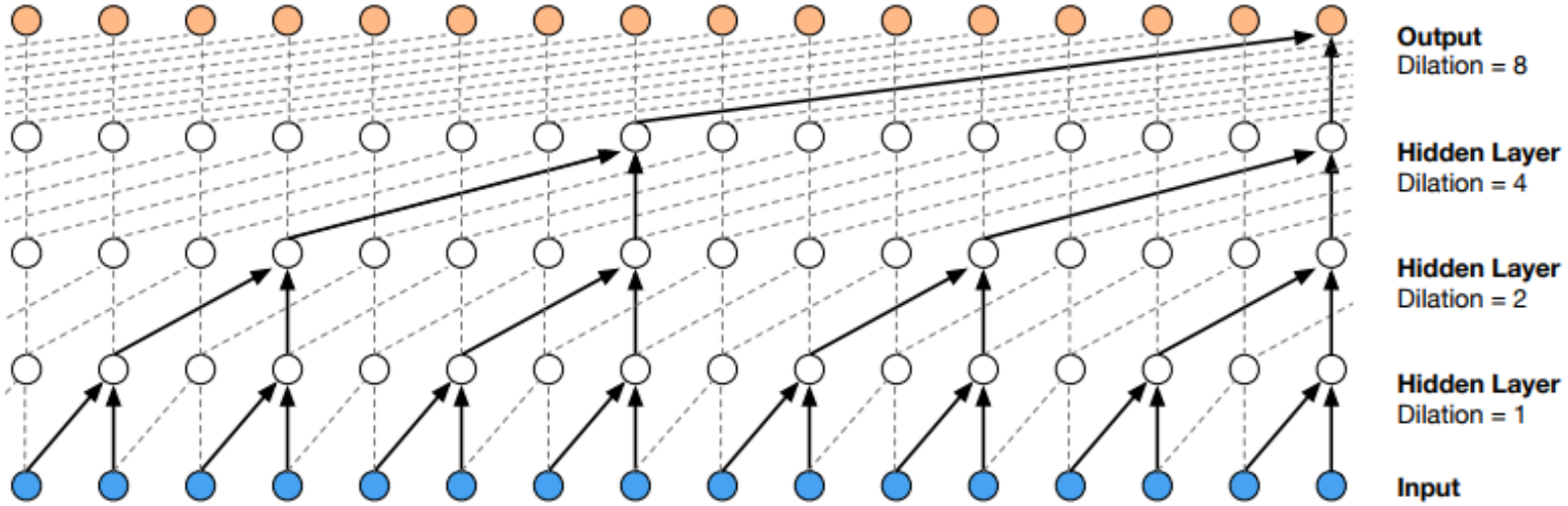
Pooling reduces dimensionality by giving up spatial location

- **max pooling** reports the maximum output within a defined neighborhood
- Padding can be SAME or VALID



Dilated Convolution





How can we control the complexity of our networks?

Dropout regularizes a broad class of models by testing lots of alternative structures

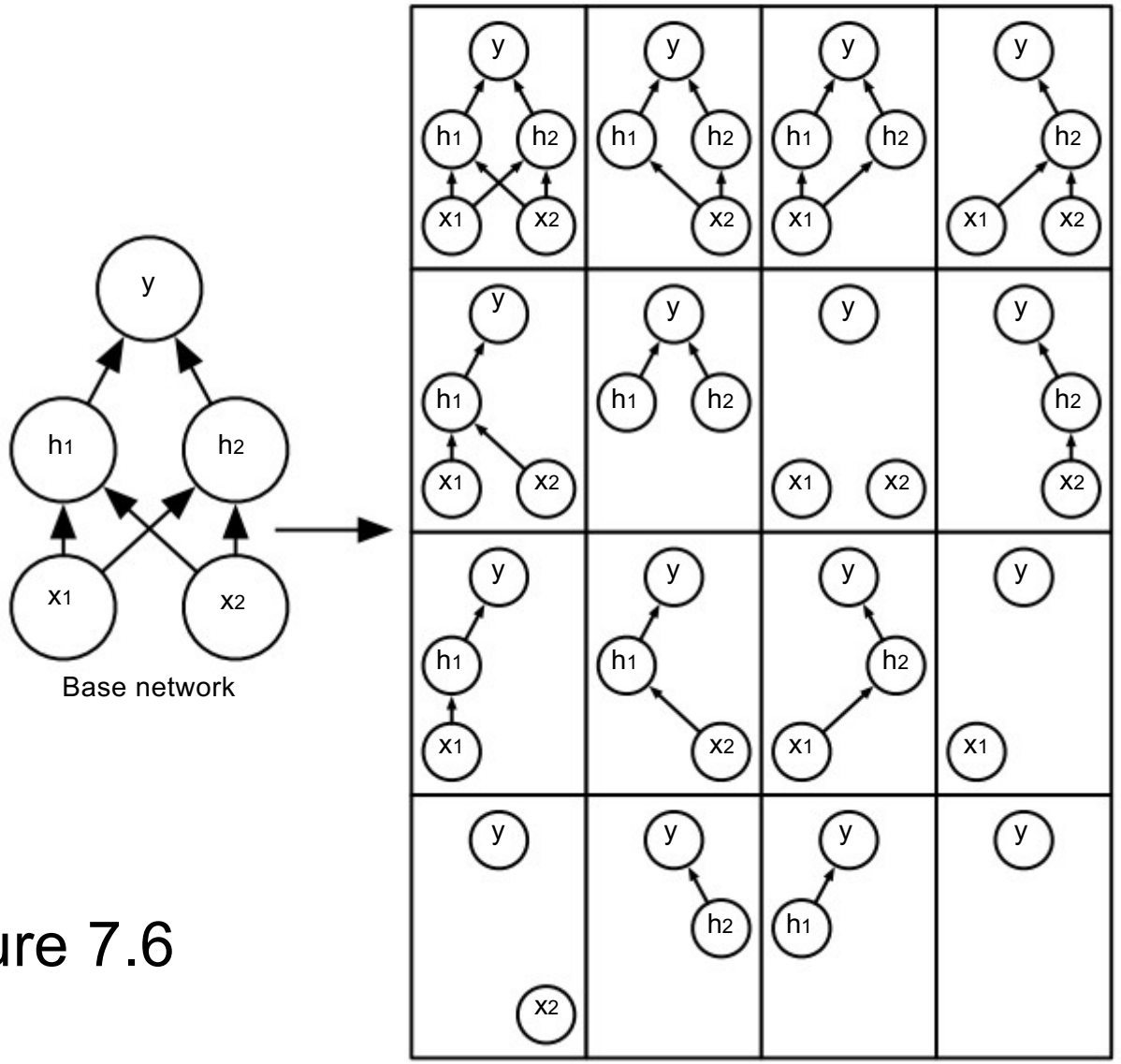


Figure 7.6

TF has a builtin dropout operator

```
# Apply Dropout
fc1 = tf.nn.dropout(fc1, dropout)
```

Probability of link dropout

Regularization is an important aspect of learning a stable model that generalizes well

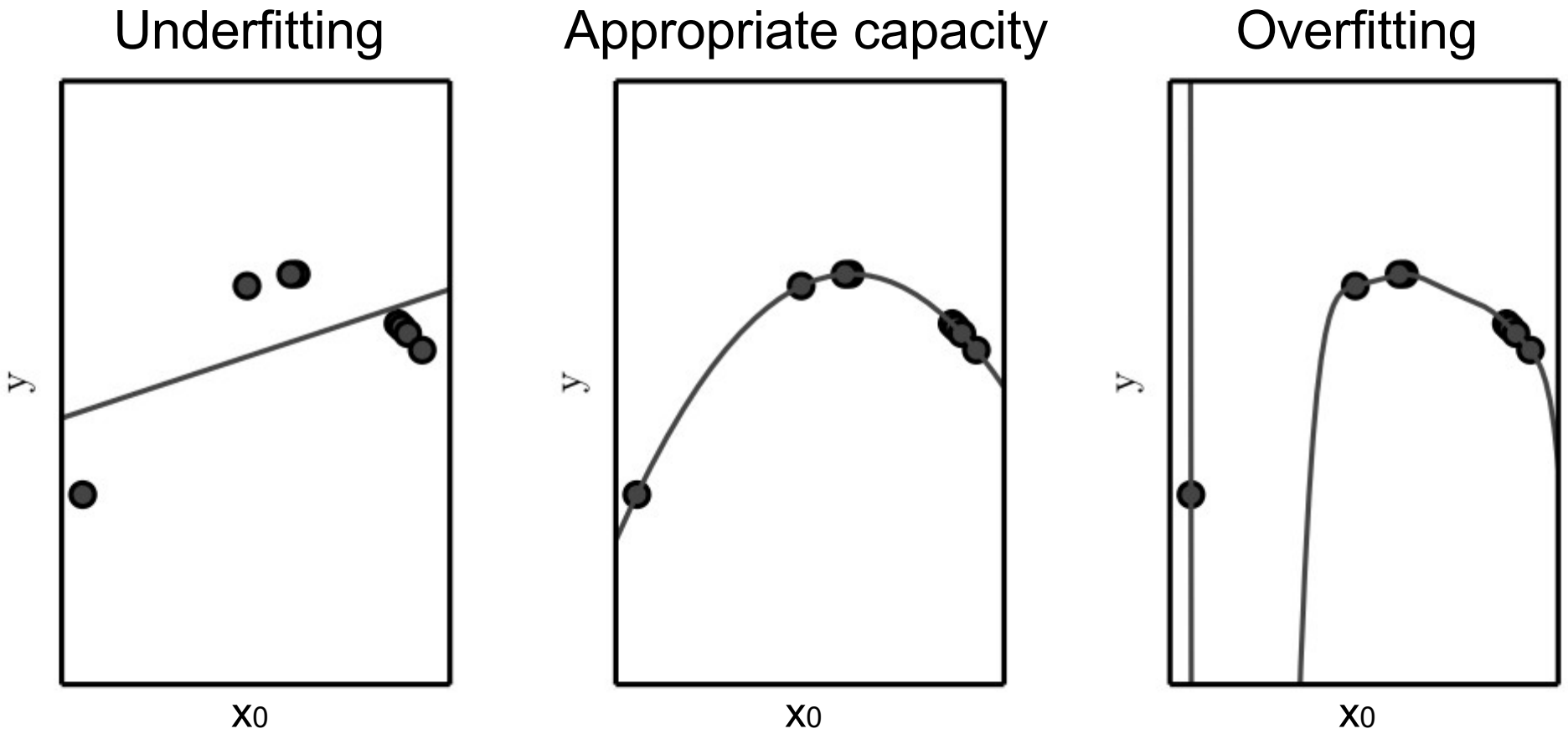


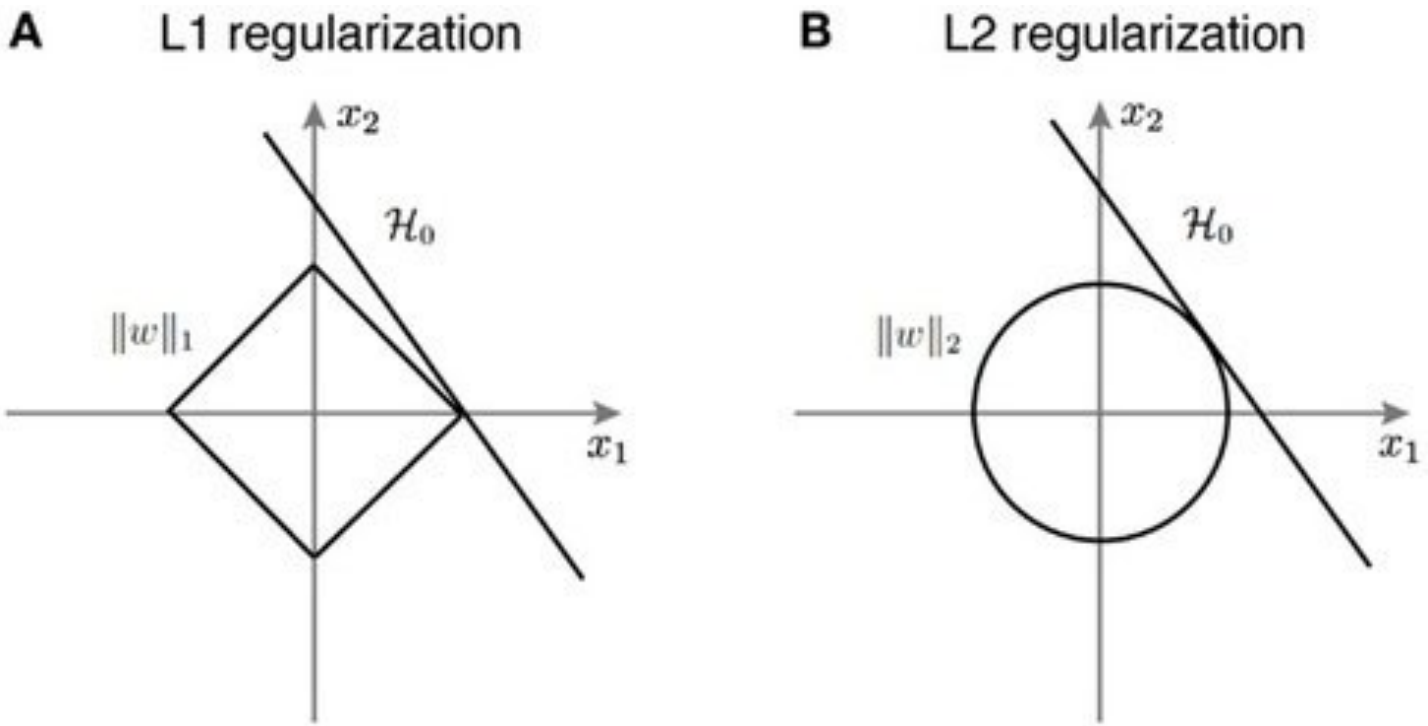
Figure 5.2

Penalizing Weights

- L1 Norm attempts to drive weights to 0 (make weight vector sparse)(p = 1)
- L2 Norm attempts to minimize magnitude of weights (p = 2)

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Visualizing L1 and L2 regularization



Weight Decay

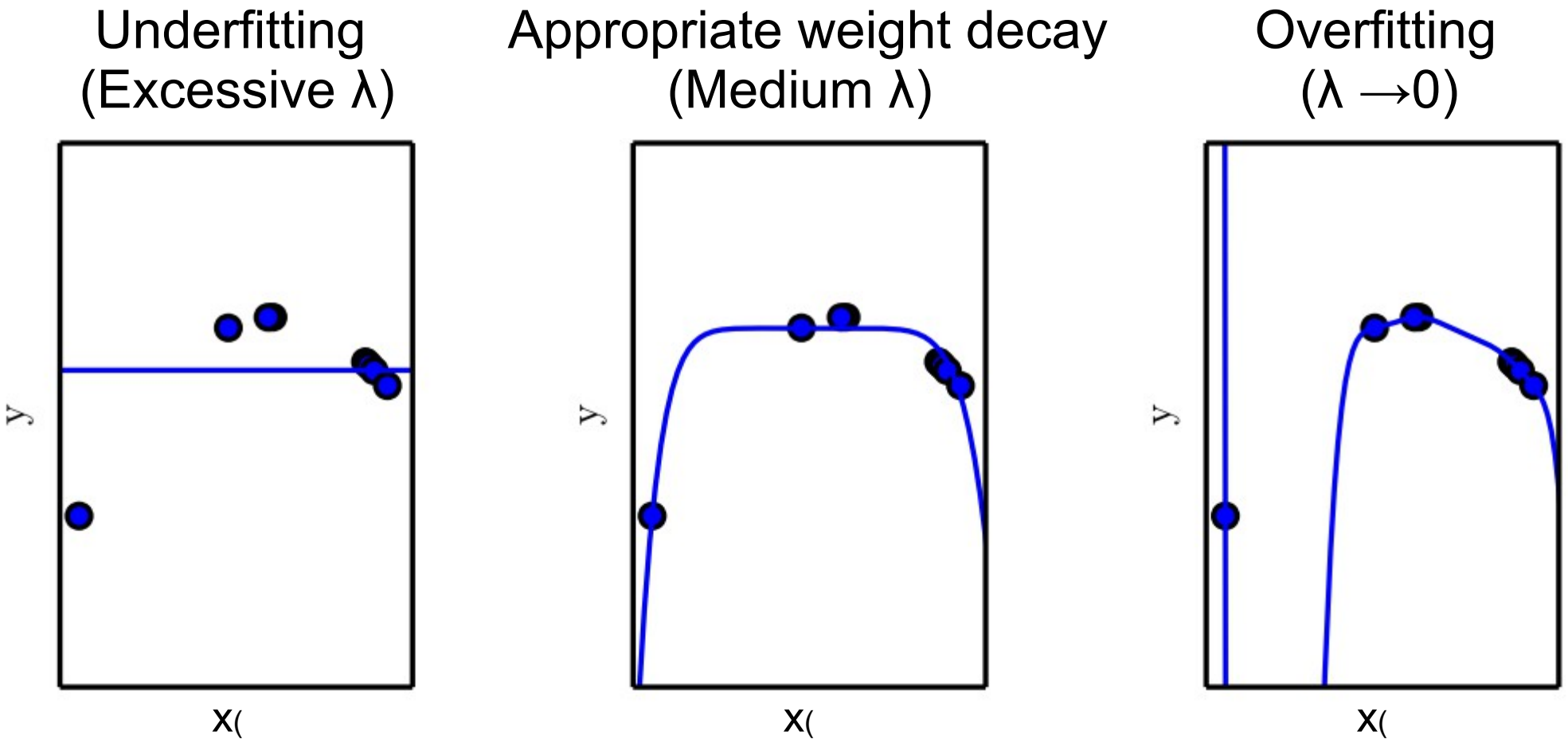


Figure 5.5

Getting the right “fit” to the training data is important

- Underfitting - high error on training set
- Overfitting - large gap between training and test error
- Correct fit is important so the model generalizes to new examples

Model capacity

- Capacity - ability to fit a wide range of functions (hypothesis space)
- Vapnik-Chervonenkis dimension - size of largest unique training set of examples a binary classifier can label arbitrarily is a measure of capacity

Generalization and Capacity

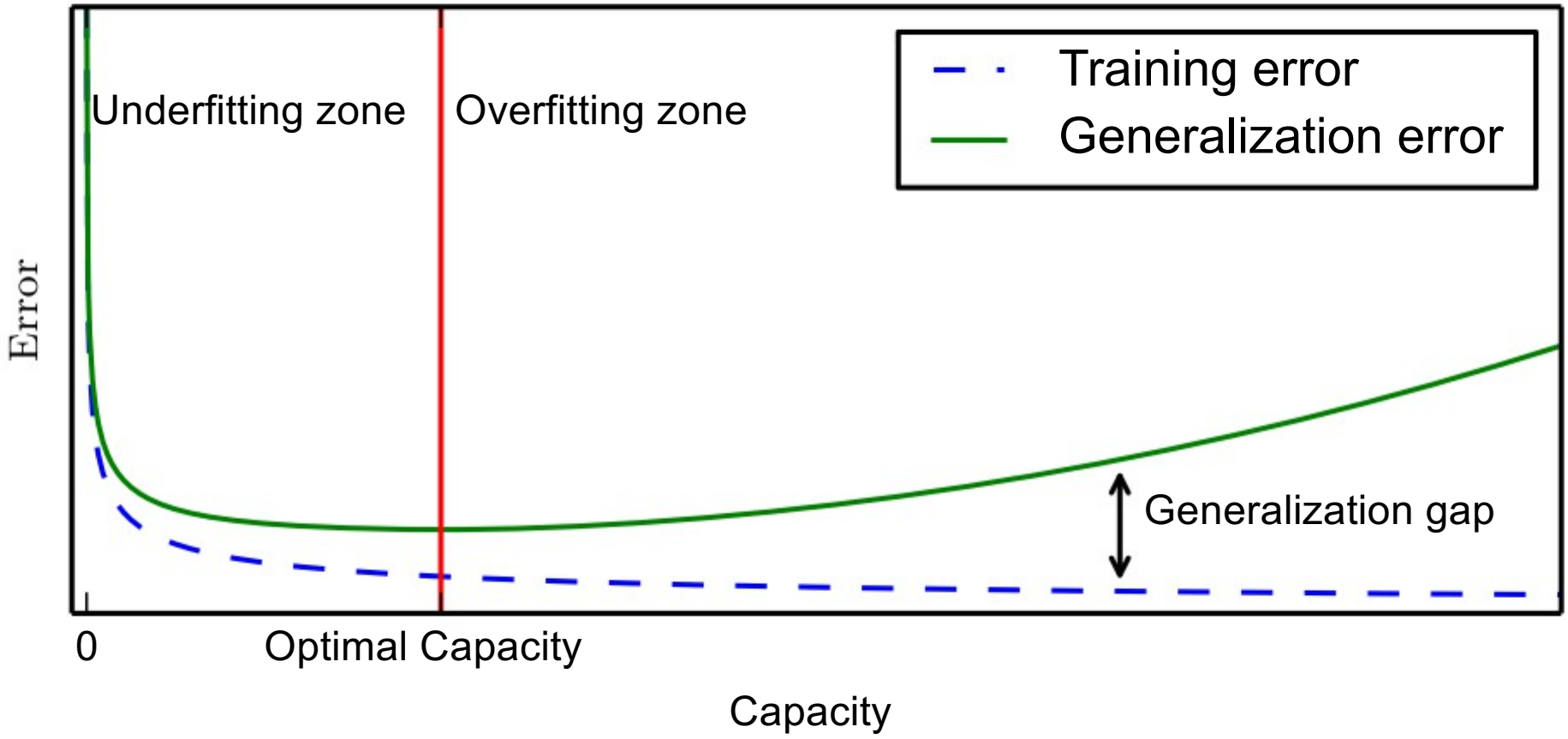


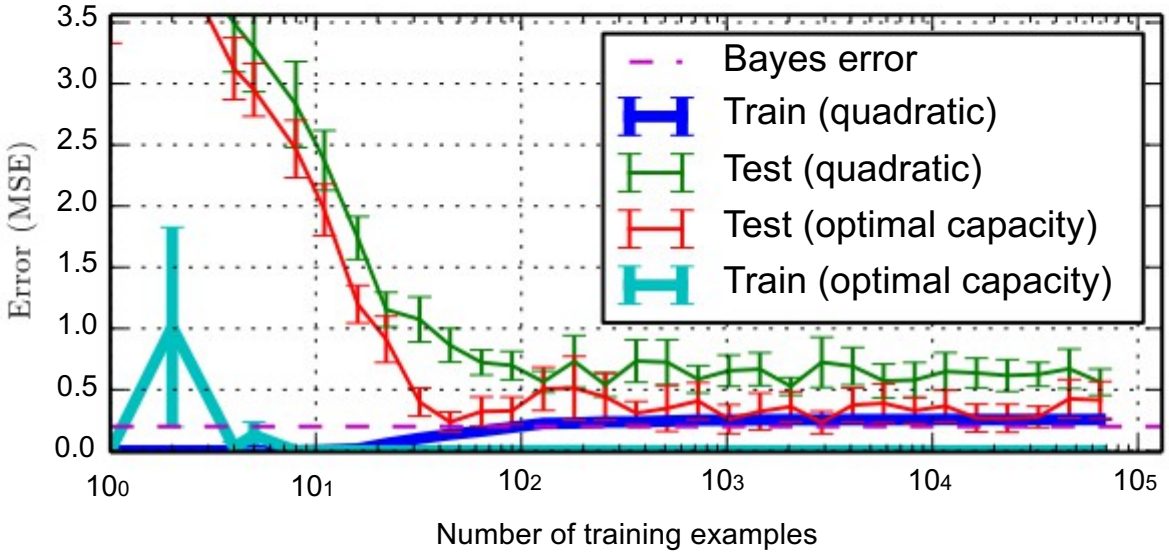
Figure 5.3

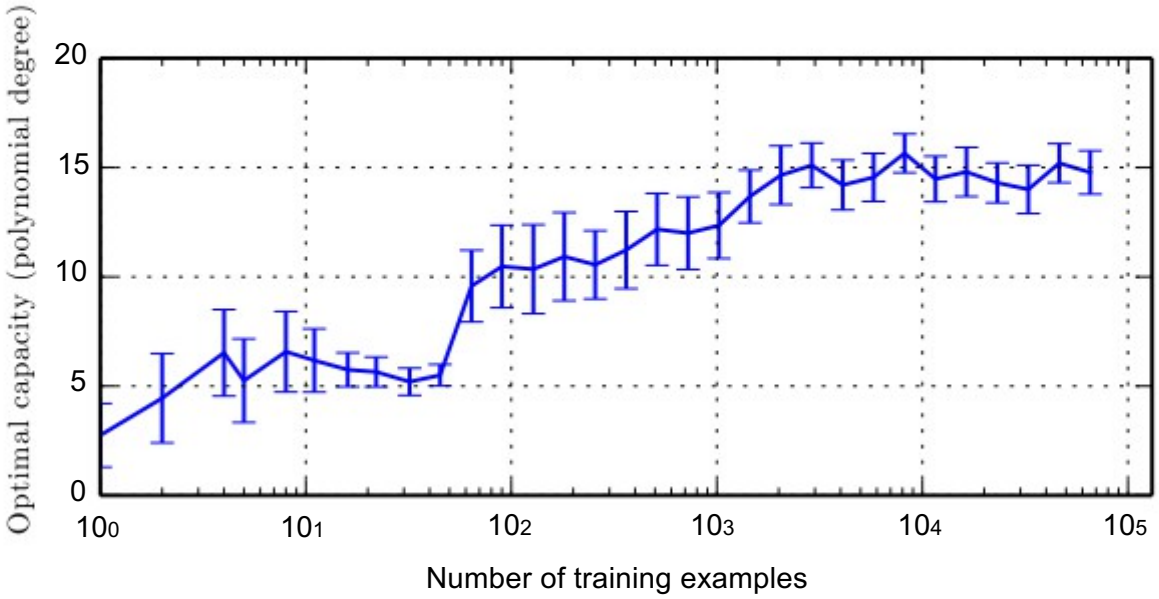
Bayes error is residual noise from confounders and observation noise

- Bayes error is the error made by an Oracle given the training set
- For example, if there is an unobserved variable that affects the labels the Oracle's predictions will be noisy

Sufficient training data are necessary to generalize well

Figure 5.4





Machine Learning has limits

- The *no free lunch* theorem: You can only obtain generalization from finitely many training examples if the algorithm searches a limited hypothesis space.
- We desire a learning algorithm to perform *generalization* and to be *stable*. It generalize if the training error on a data set will converge to the expected error. It is stable if small perturbations in the data result in only small perturbations in the output hypothesis.

Recurrent neural networks (RNNs) model sequences

Recurrent networks share parameters across time indexes

- Recurrent networks have inter-time dependencies with each time-step sharing parameters.
- For example, such dependencies can include the value of the hidden units or outputs of the previous stage
- Examples: speech understanding, language translation

We can unfold a simple one layer network to handle sequential data with all parameters shared between time points

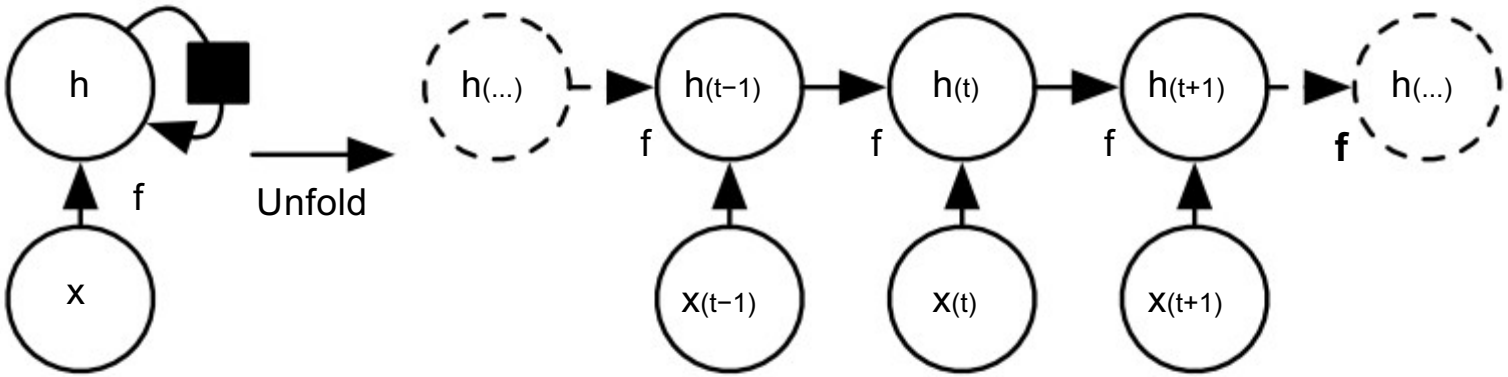
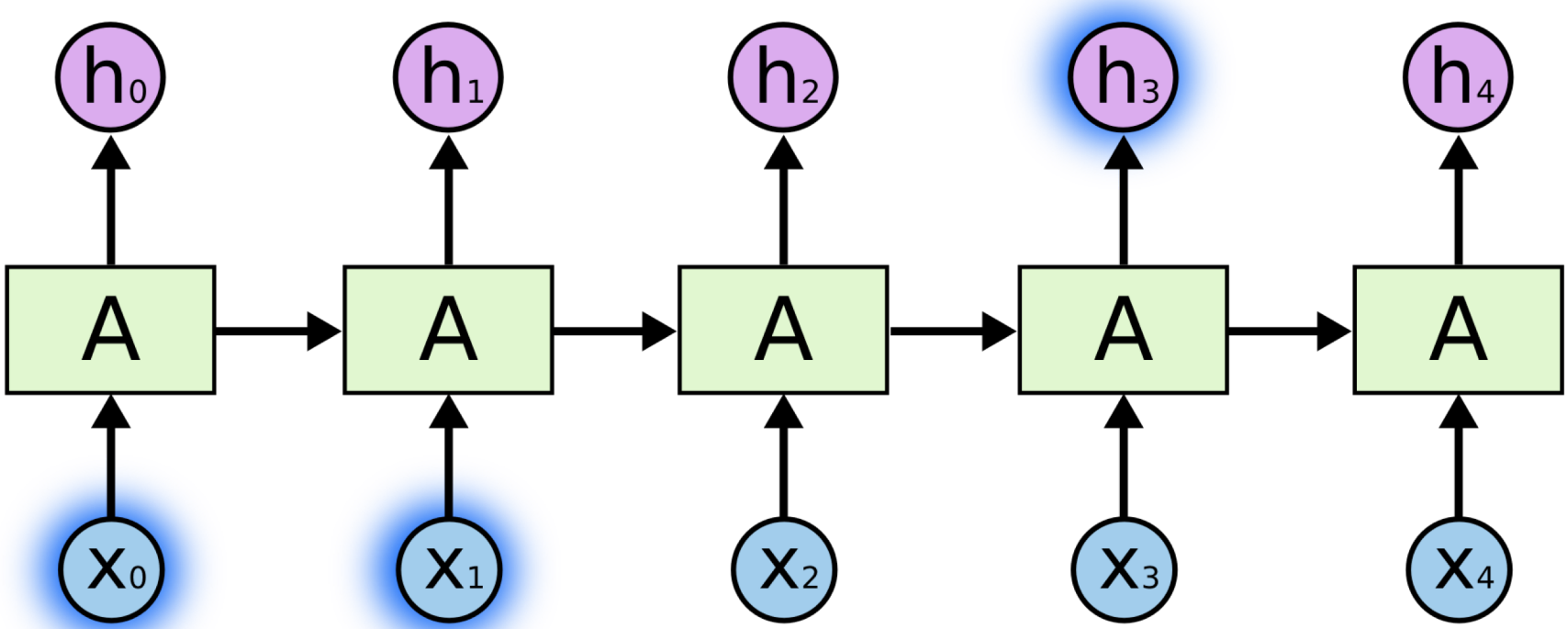
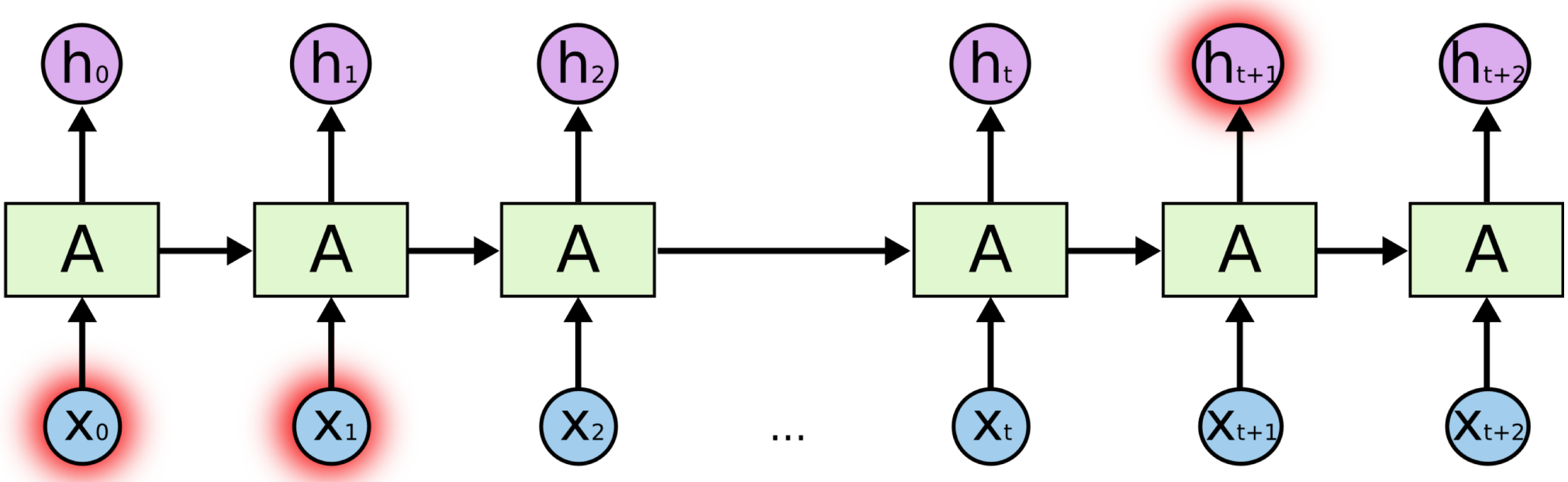


Figure 10.2

RNNs can propagate information across time steps



Increasing distance makes this more difficult



We can train an RNN based upon data likelihood as before

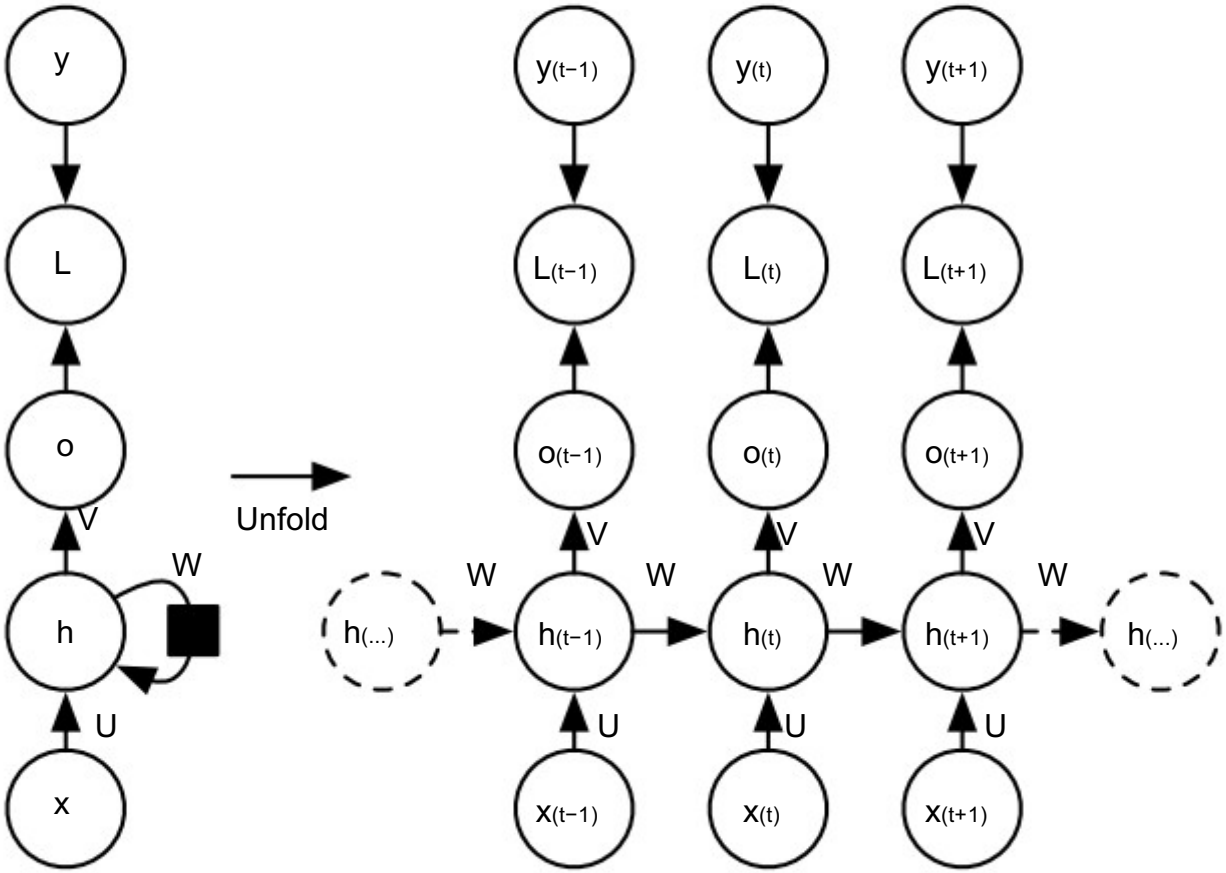


Figure 10.3

Vanishing and exploding gradients during training are an issue with RNNs

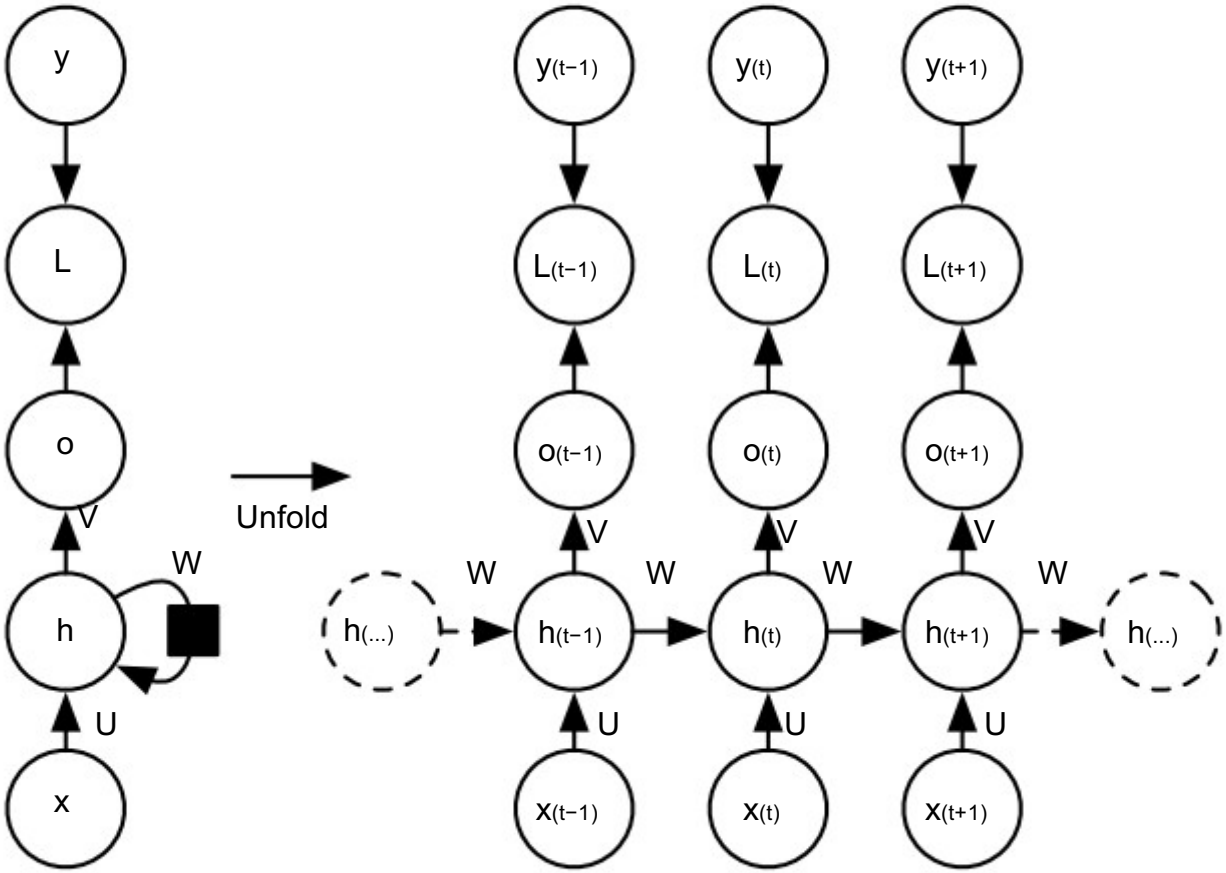
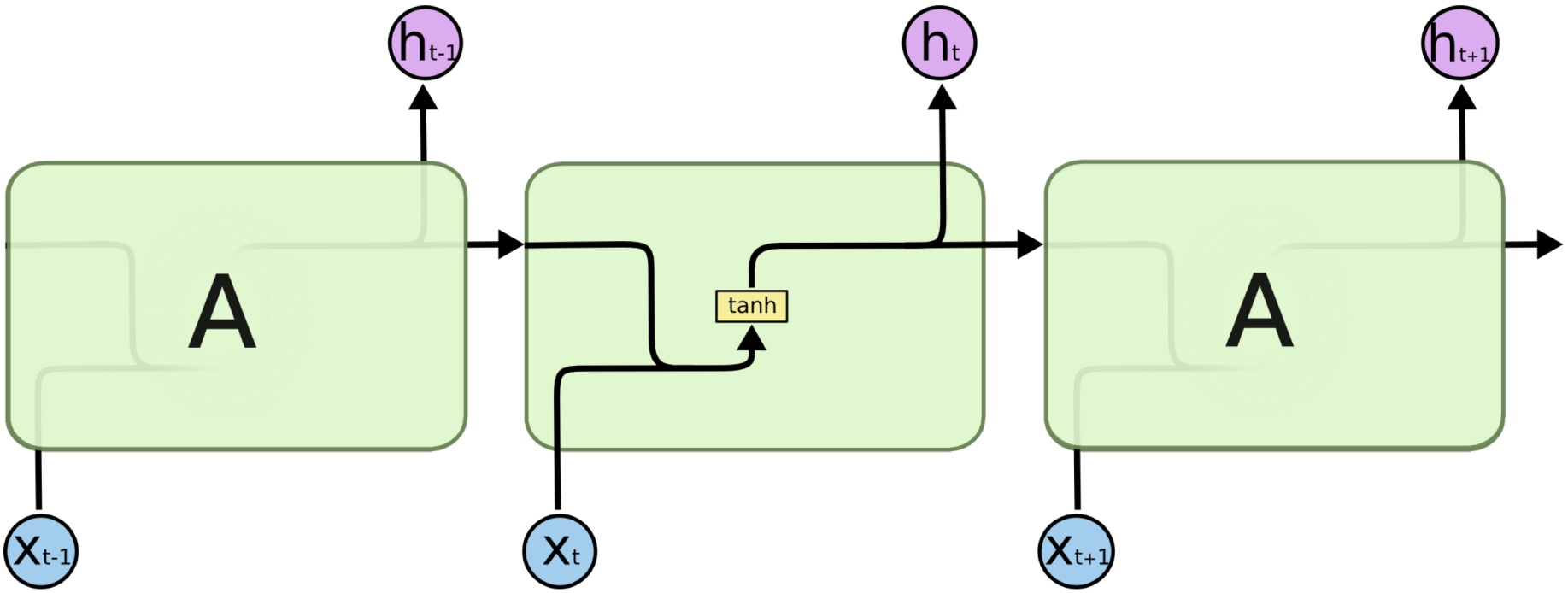


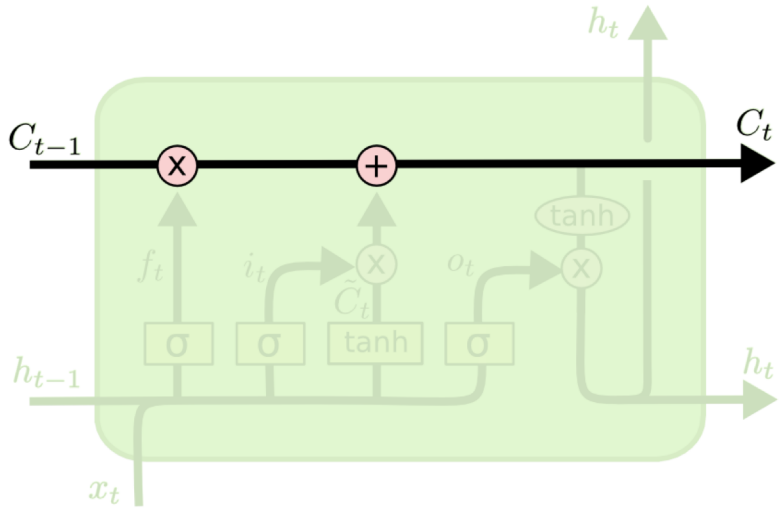
Figure 10.3

A simple RNN



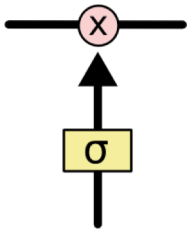
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

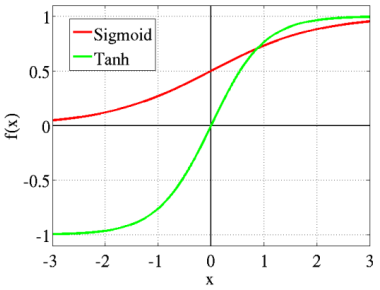
Key innovation of an LSTM is a cell state "conveyor belt"

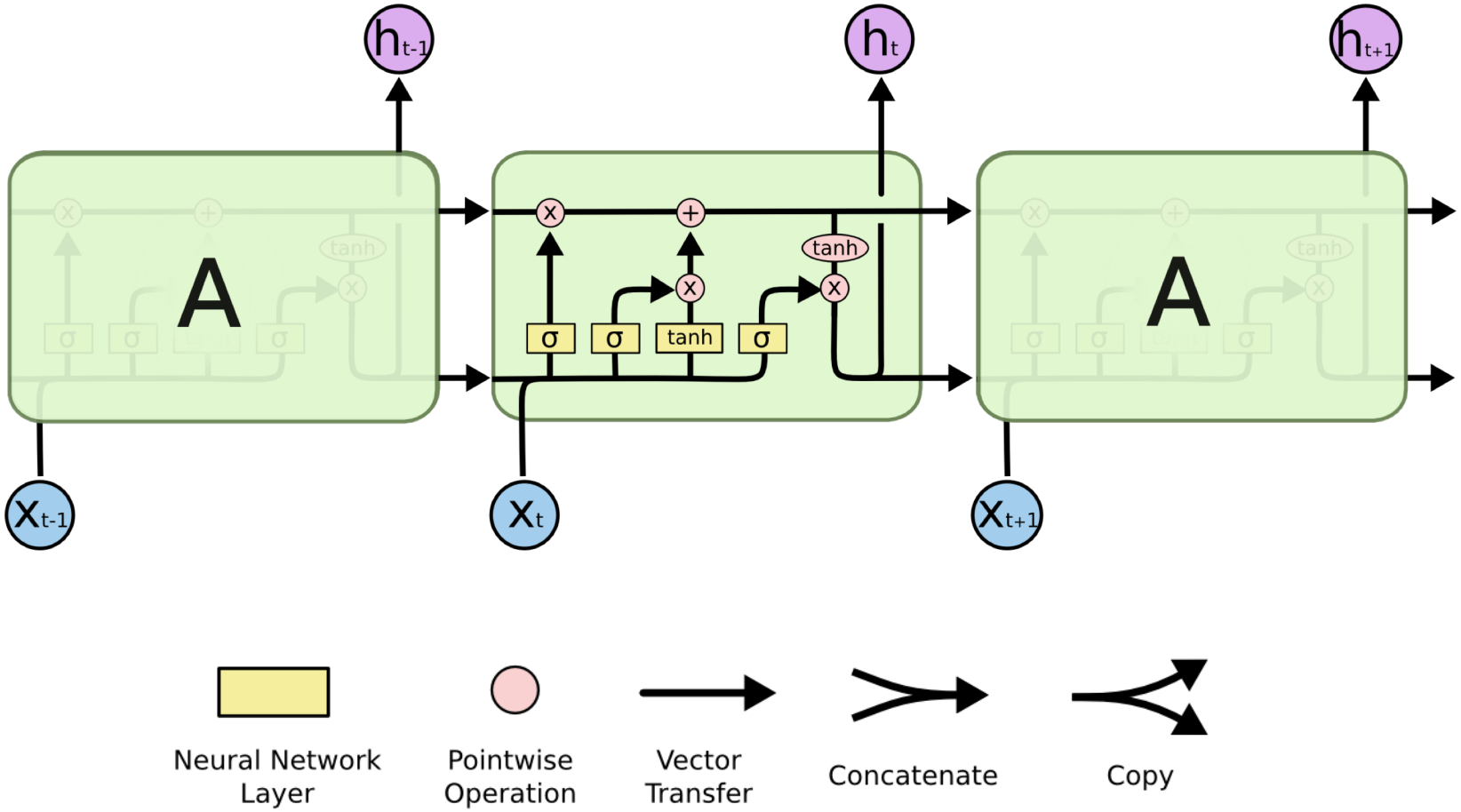


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

An LSTM includes three gates

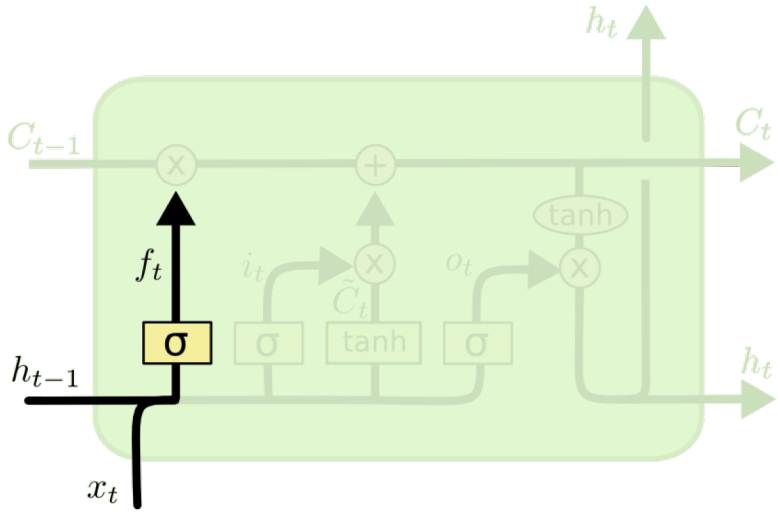






http://colah.github.io/posts/2015-08-Understanding-LSTMs/

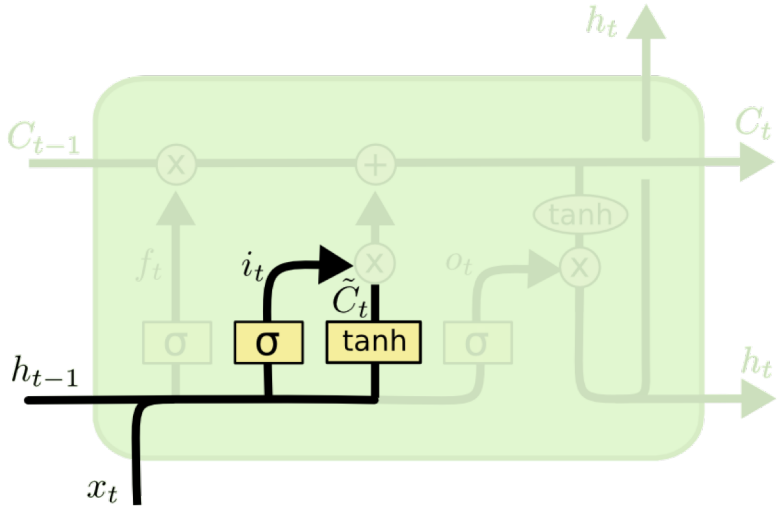
The LSTM forget gate can erase previous cell state – “forget gate”



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

An LSTM can create new memory state "input gate"

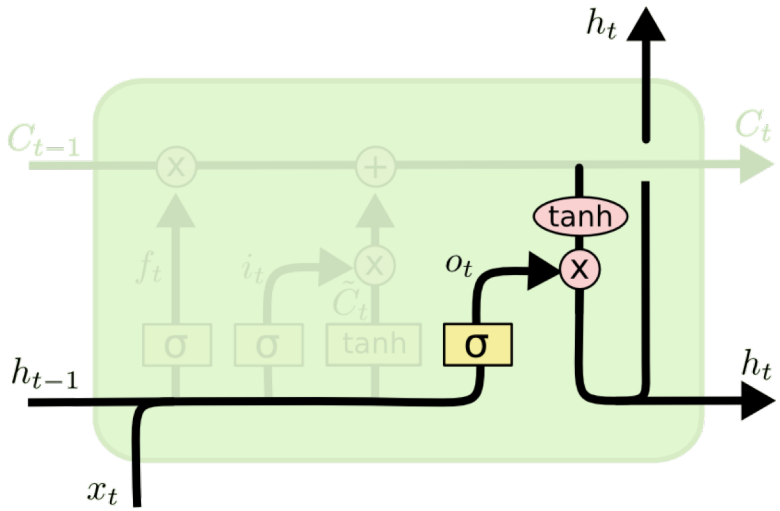


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

An LSTM produces an output state "output gate"

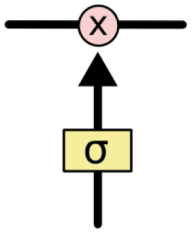


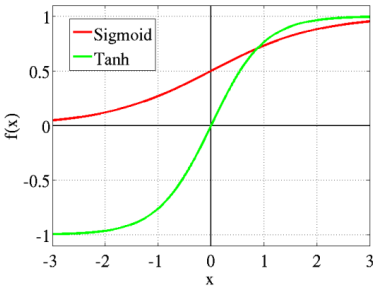
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

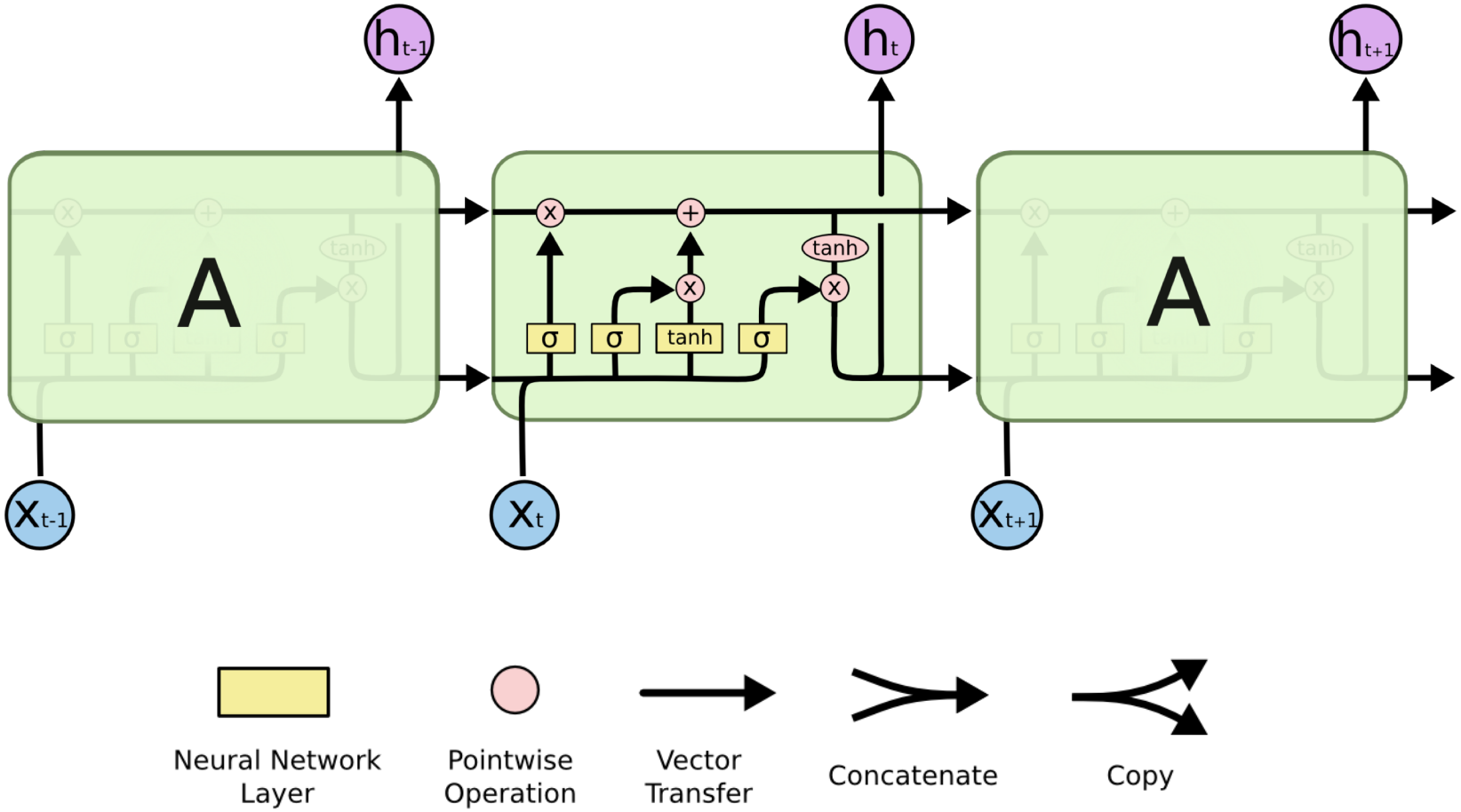
$$h_t = o_t * \tanh\left(C_t\right)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

An LSTM includes three gates



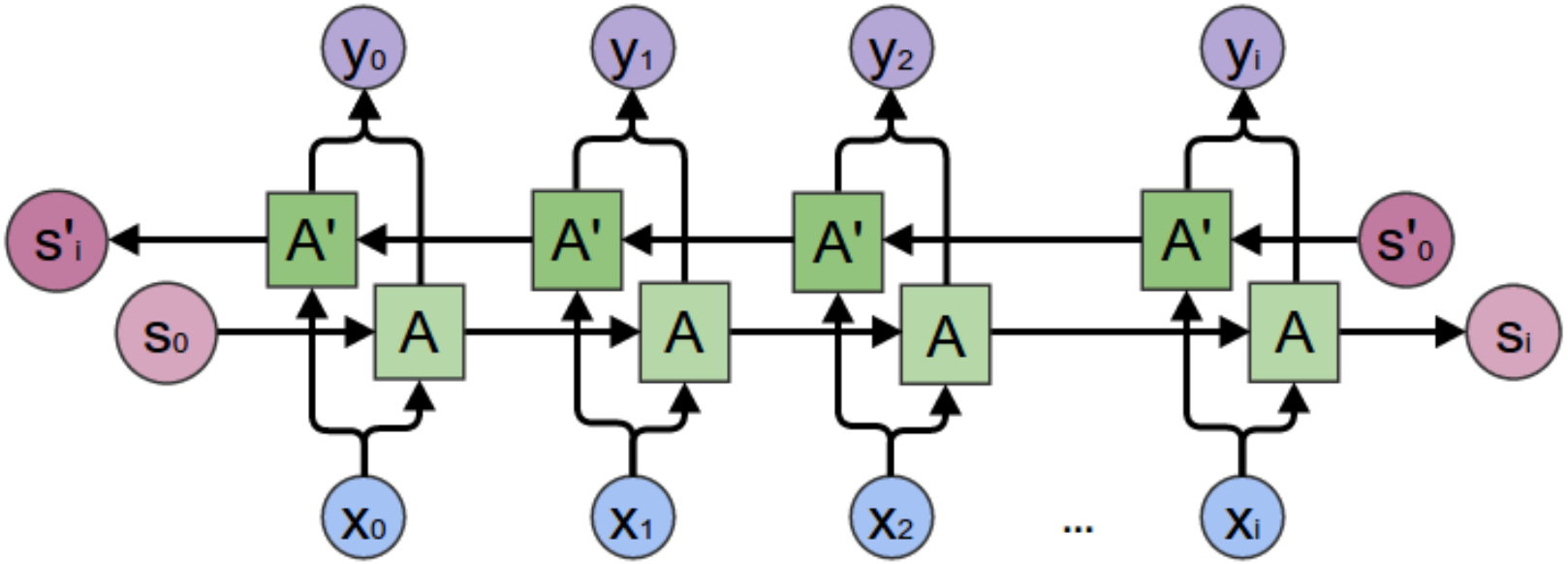




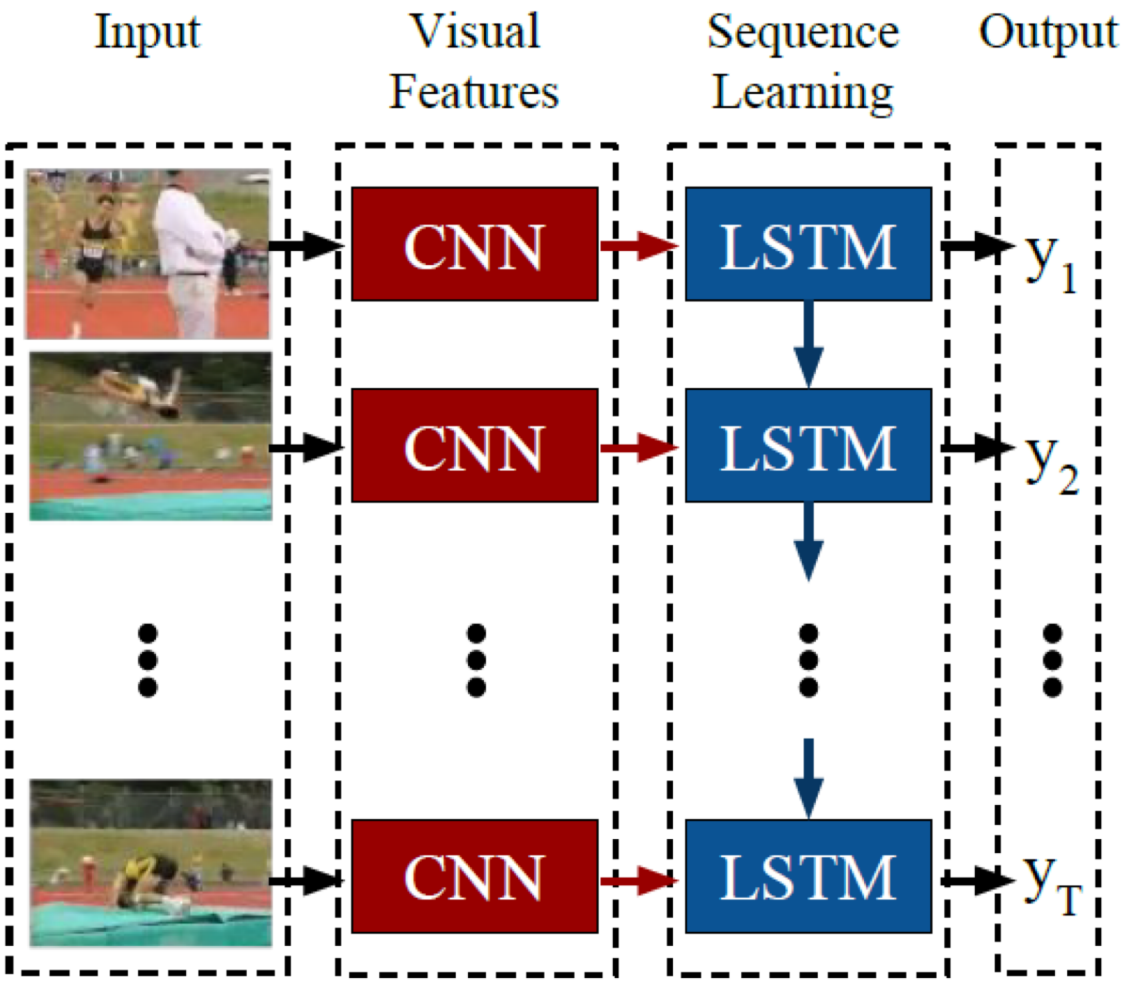
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

How can we propagate state in both directions in a sequence?

Bidirectional LSTMs



CNNs and LSTMs can be combined



Observations

- Convolution and pooling permit parameter sharing
  - Network architecture is key
  - Backprop works with convolution and pooling
  - Need to consider network capacity for stable, generalizable results
- Recurrent networks unroll a network to permit the processing of variable length sequential data
  - Text is typically processed by an RNN
  - LSTMs are a form of RNN that is optimized for the communication of information through time

FIN - Thank You