Neural Networks 2

CS446 Machine Learning

Administrative

HW 5 is out. Due on April 11

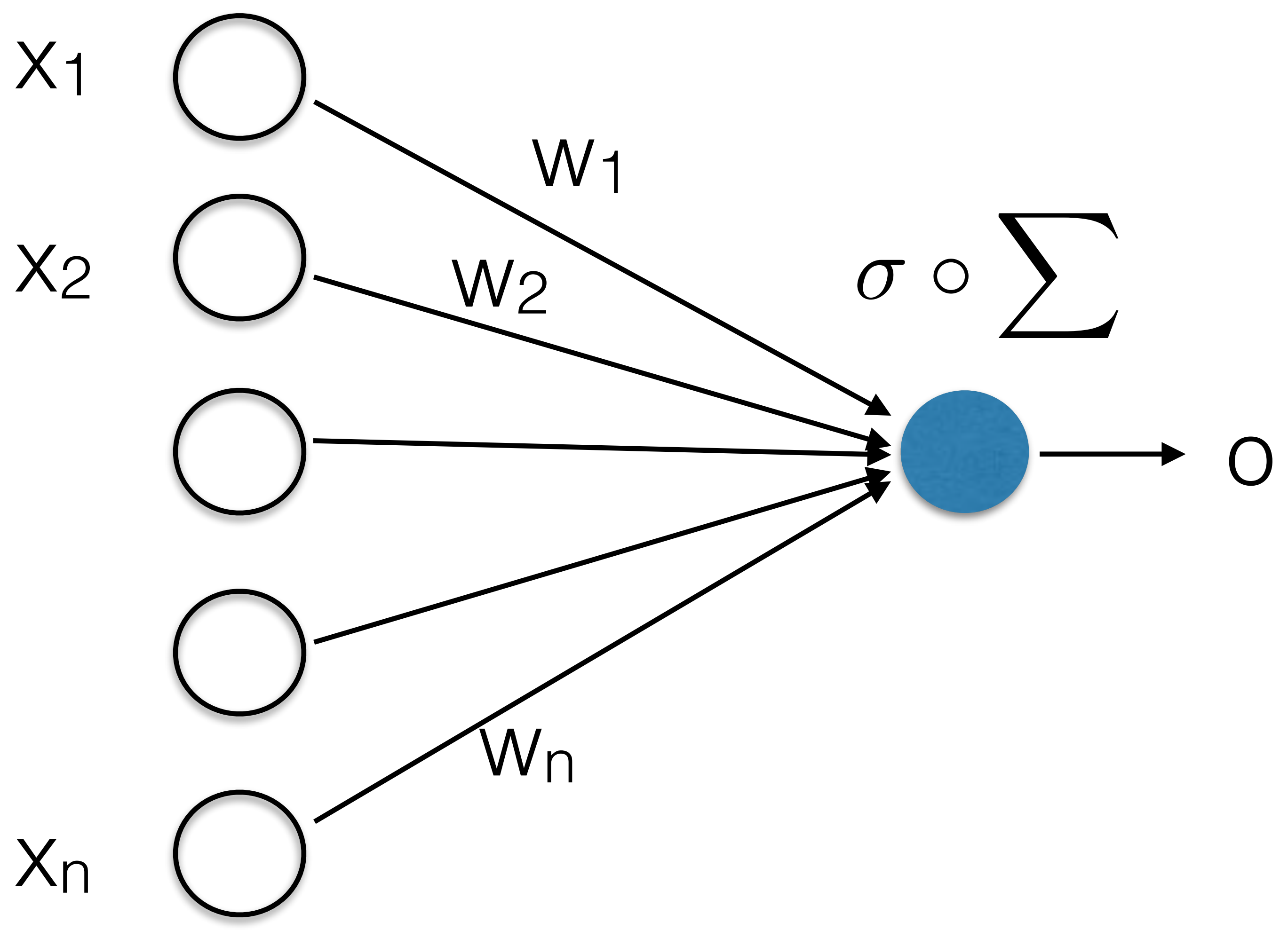
No class on Thursday

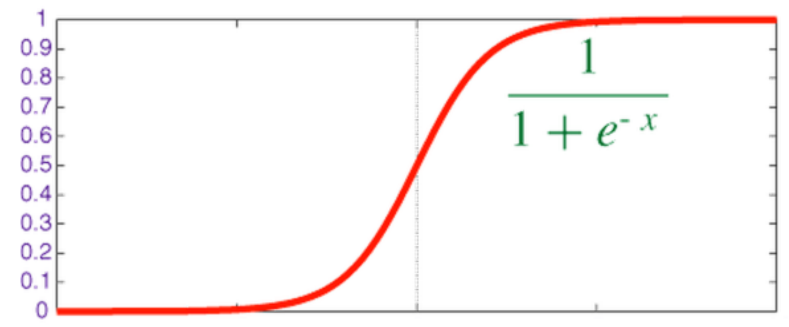
Slides are released. Try to follow the pdf.
(since these slides are made in keynote, and exporting to ppt does not work well)

Corrections regarding derivative in the last slide

What did we learn in last class ?

Basic Unit : Neuron



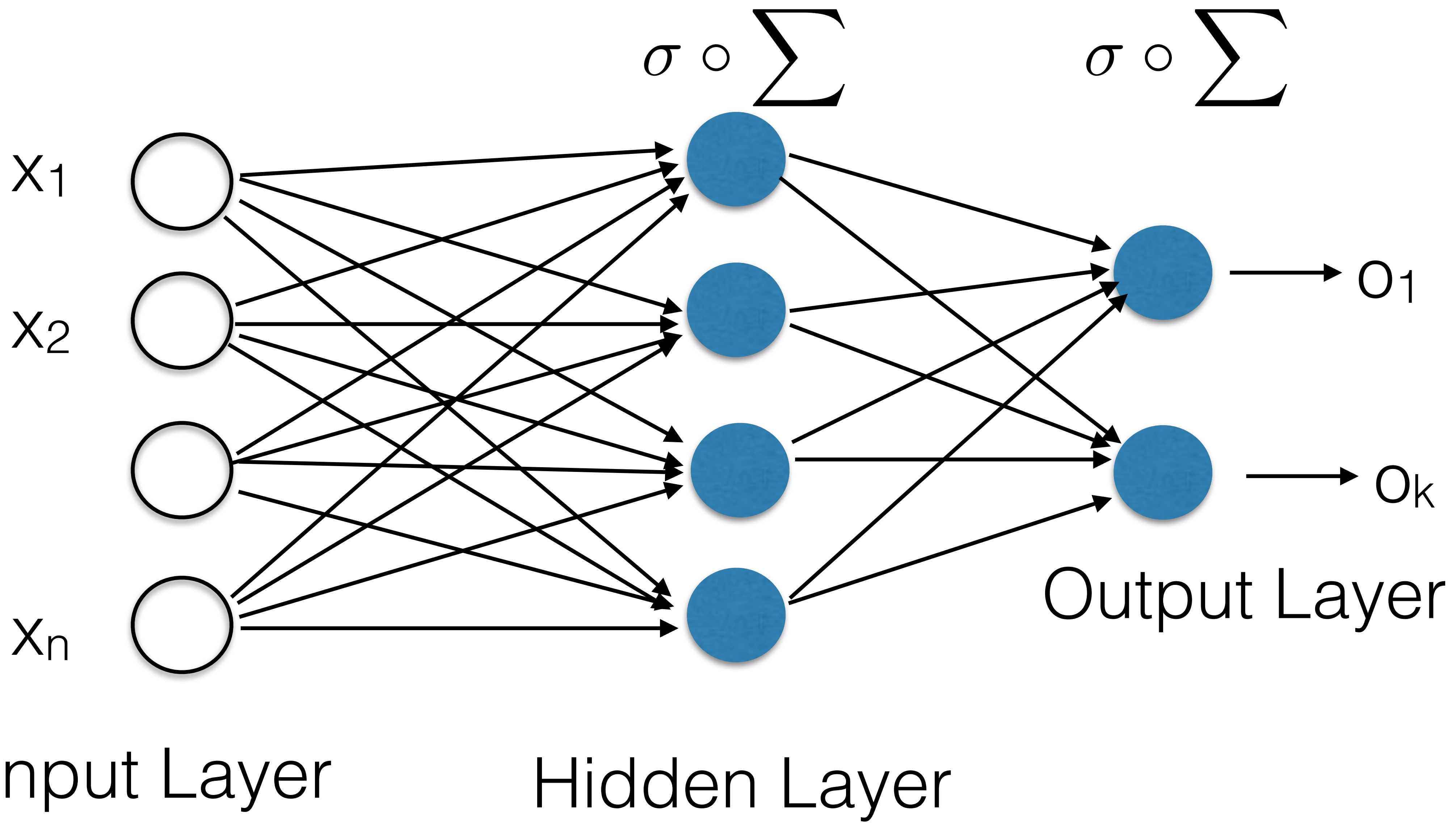


$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$o = \sigma(\sum_i w_i x_i - T)$$

Note that we can write this as $o = \sigma(W\mathrm{x})$

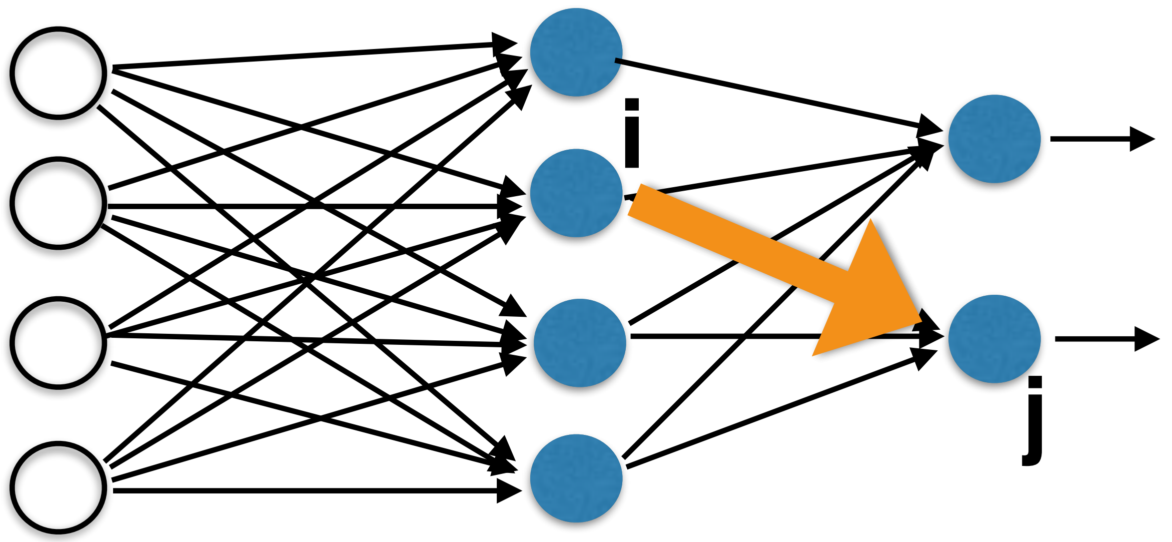
Stack them up



Feedforward Neural Network

This allows us to capture complex non-linear functions of the input data

Derivation of Learning Rule



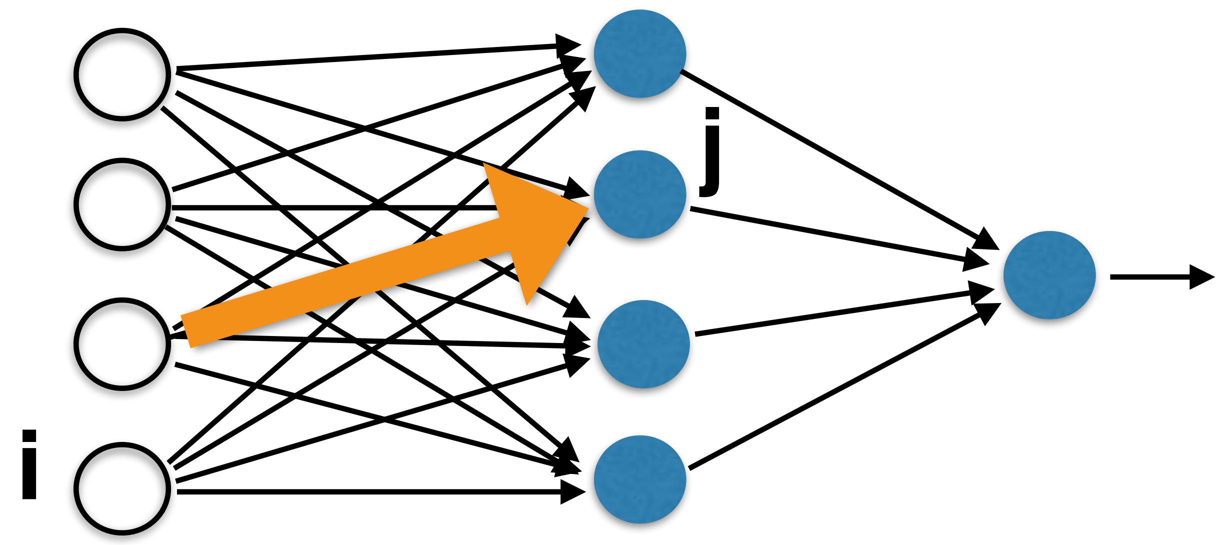
Connection $\mathbf{w}_{ij}$ between hidden and output layer is updated as

$$\Delta w_{ij} = R(t_j - o_j)o_j(1 - o_j)\, x_i$$
$$= R\delta_j\, x_i$$

where

$$\delta_j = (t_j - o_j)o_j(1 - o_j) \quad = -\frac{\delta E_d}{\delta net_j}$$

Derivation of Learning Rule



Connection $\mathbf{w_{ij}}$ between input and hidden layer is updated as

$$\Delta w_{ij} = R o_j(1 - o_j)\left(\sum_{k \in downstream(j)} \delta_k w_{jk}\right) x_i = R\delta_j x_i$$

where

$$\delta_j = o_j(1 - o_j)\left(\sum_{k \in downstream(j)} \delta_k w_{jk}\right)$$

First determine the error for the output units. Then, back propagate this error layer by layer through the network, changing weights appropriately in each layer

The Backpropagation Algorithm

For each example in the training set, do:

1. Compute the network output for this example
2. Compute the error term between he output and target values
3. For each output unit **j**, compute error term:

$$\delta_j = (t_j - o_j)o_j(1 - o_j)$$

4. For each hidden unit, compute error term

$$\delta_j = o_j(1 - o_j)\left(\sum_{k \in downstream(j)} \delta_k w_{jk}\right)$$

5. Update weights by $\Delta w_{ij} = R\delta_j x_i$

Questions ??

Today’s Plan

Recurrent Neural Networks
(and applications to NLP)

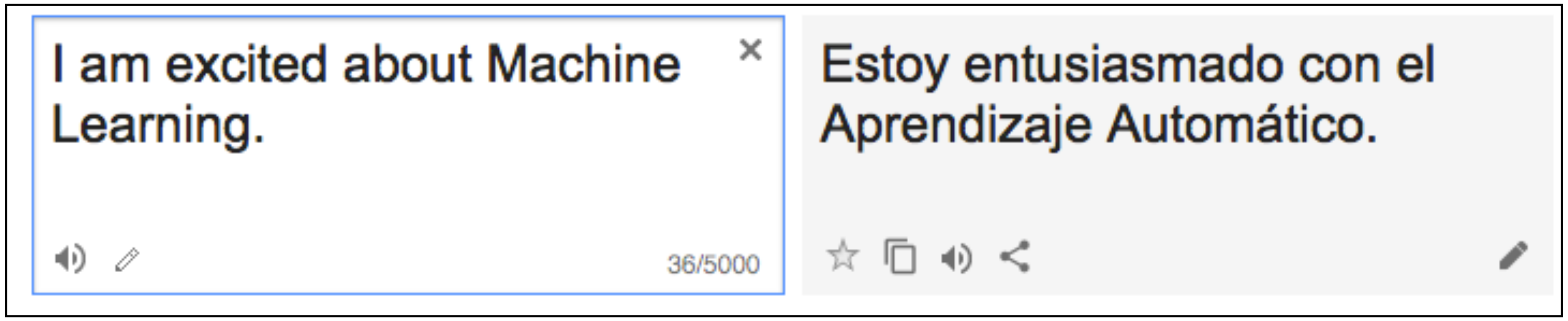
Convolutional Neural Networks
(and applications to Vision)

Recurrent
Neural Networks

Motivation

Not all problems can be converted into one with fixed length inputs and outputs

Machine Translation : translating English sentence to other language


I am excited about Machine Learning.

36/5000

Estoy entusiasmado con el Aprendizaje Automático.

Speech Recognition : translating spoken sentence to written sentence

Hard or impossible to choose a large enough window, there can always be a new sentence longer than anything seen

Recurrent Neural Networks

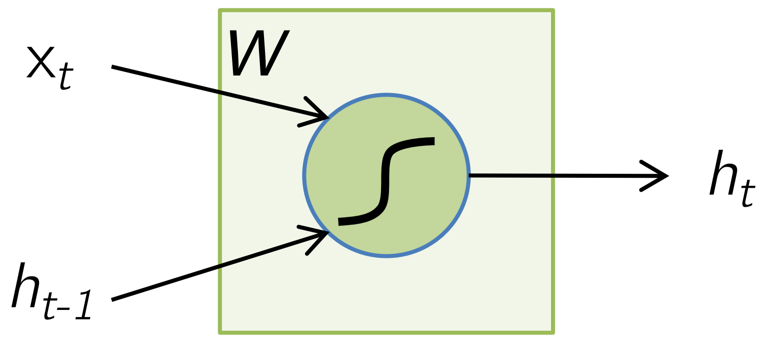
- Recurrent Neural Networks take as input each element of a sequence one by one. (Think of this as taking input $\mathbf{x_1}$ at time step 1, $\mathbf{x_2}$ at time step 2, etc)

- Recurrent Neural Networks take the previous output or hidden states as inputs.

- The composite input at time t has some historical information about the happenings at time $T < t$

- RNNs are useful as their intermediate values (state) can store information about past inputs for a time that is not fixed a priori

RNN cell



Input at time t is $\mathbf{x_t}$

State at the end of time (t-1) is $\mathbf{h_{t-1}}$

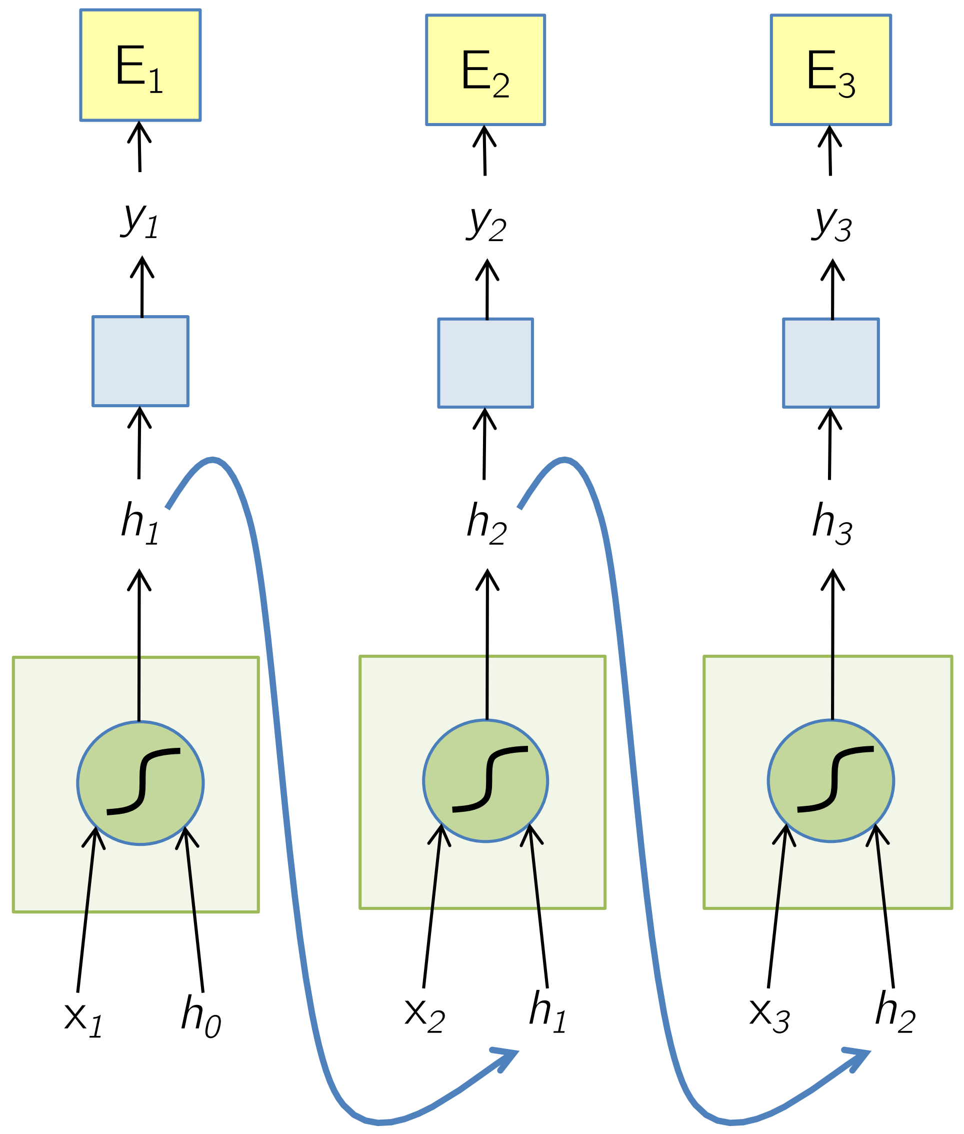
State at the end of time t is $\mathbf{h_t}$

$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

You can obviously replace **tanh** by your favorite non-linear differentiable function

How is it different from a standard neuron ??

Feeding a sequence to an RNN



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

You can output **y**

$$y_t = F(h_t)$$

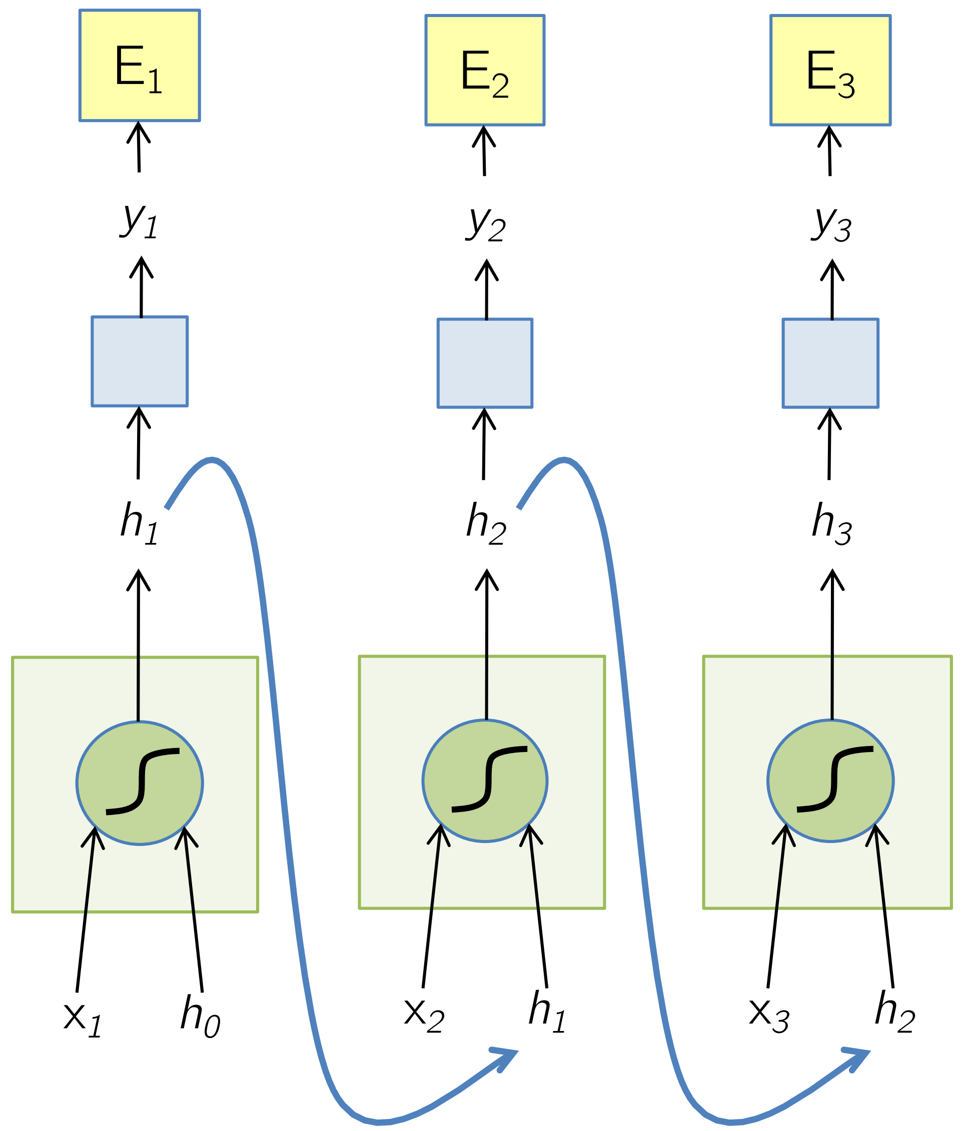
often linear function of $\mathbf{h_t}$

Finally define error $\mathbf{E_t}$ based on $\mathbf{y_t}$ and the target output at time **t**

$$E_t = Loss(y_t, GT_t)$$

$\mathbf{GT_t}$ is target at time **t**

RNNs



- Note that the weights are shared over time
- Essentially, copies of the RNN cell are made over time (unrolling/unfolding), with different inputs at different time steps

Sentiment Classification

Classify a restaurant review from Yelp or movie review from IMDB as positive or negative

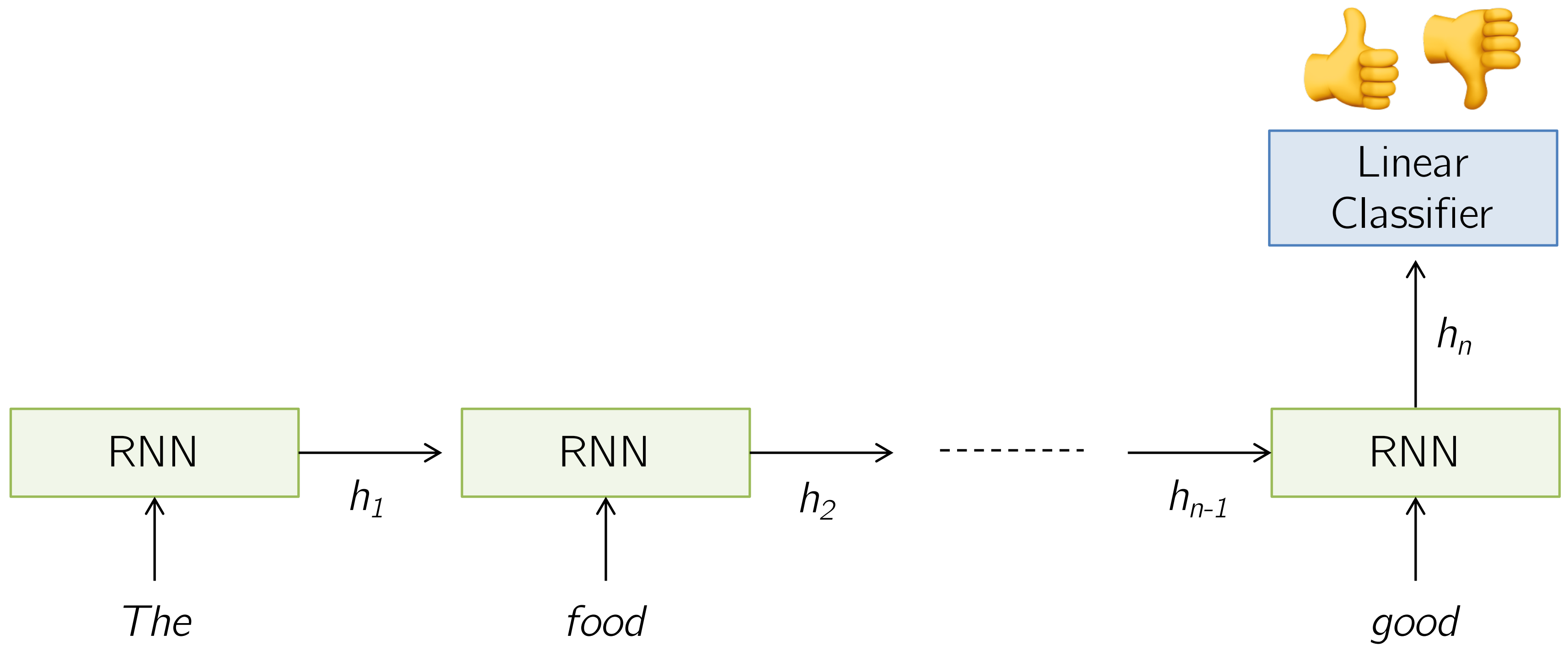
Inputs : Multiple words, one or more sentences

Outputs : Positive or negative classification

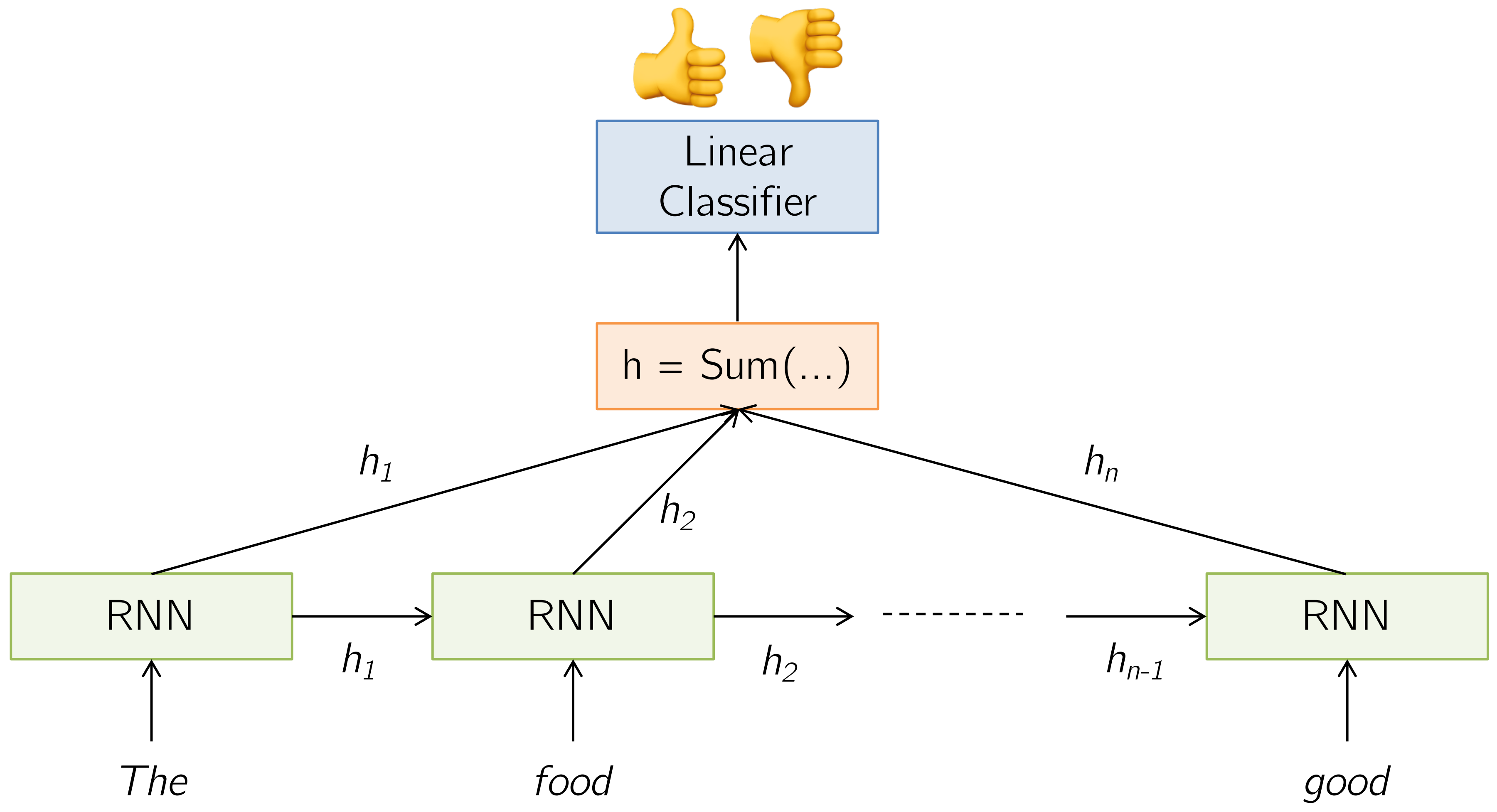
“The food was really good”

“Don’t try the pizza, its so good you will come back everyday. I hate this place. ”

Sentiment Classification



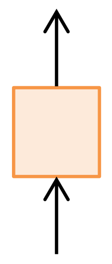
Sentiment Classification



http://deeplearning.net/tutorial/lstm.html

Input Output Scenario

Single - Single



Feed-forward network

Single - Multiple

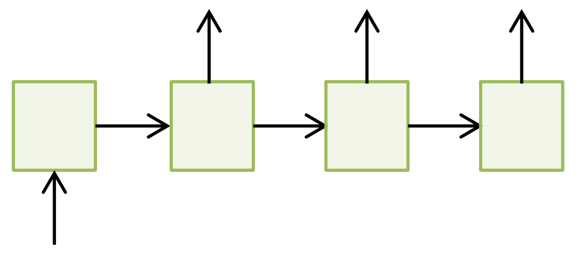
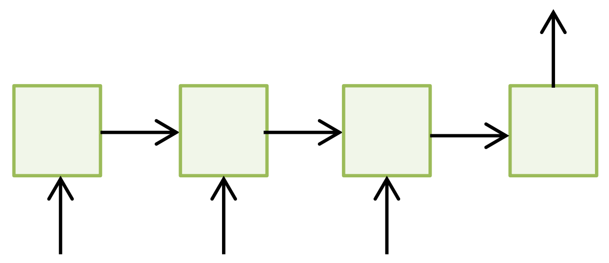


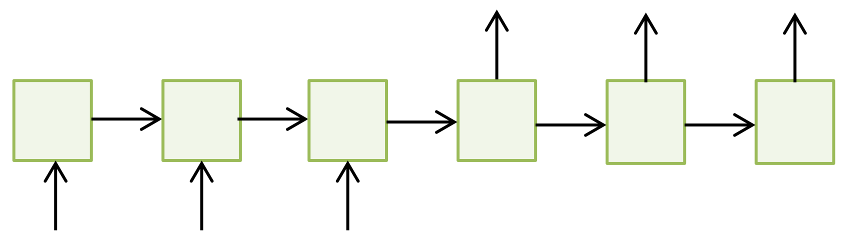
Image captioning

Multiple - Single



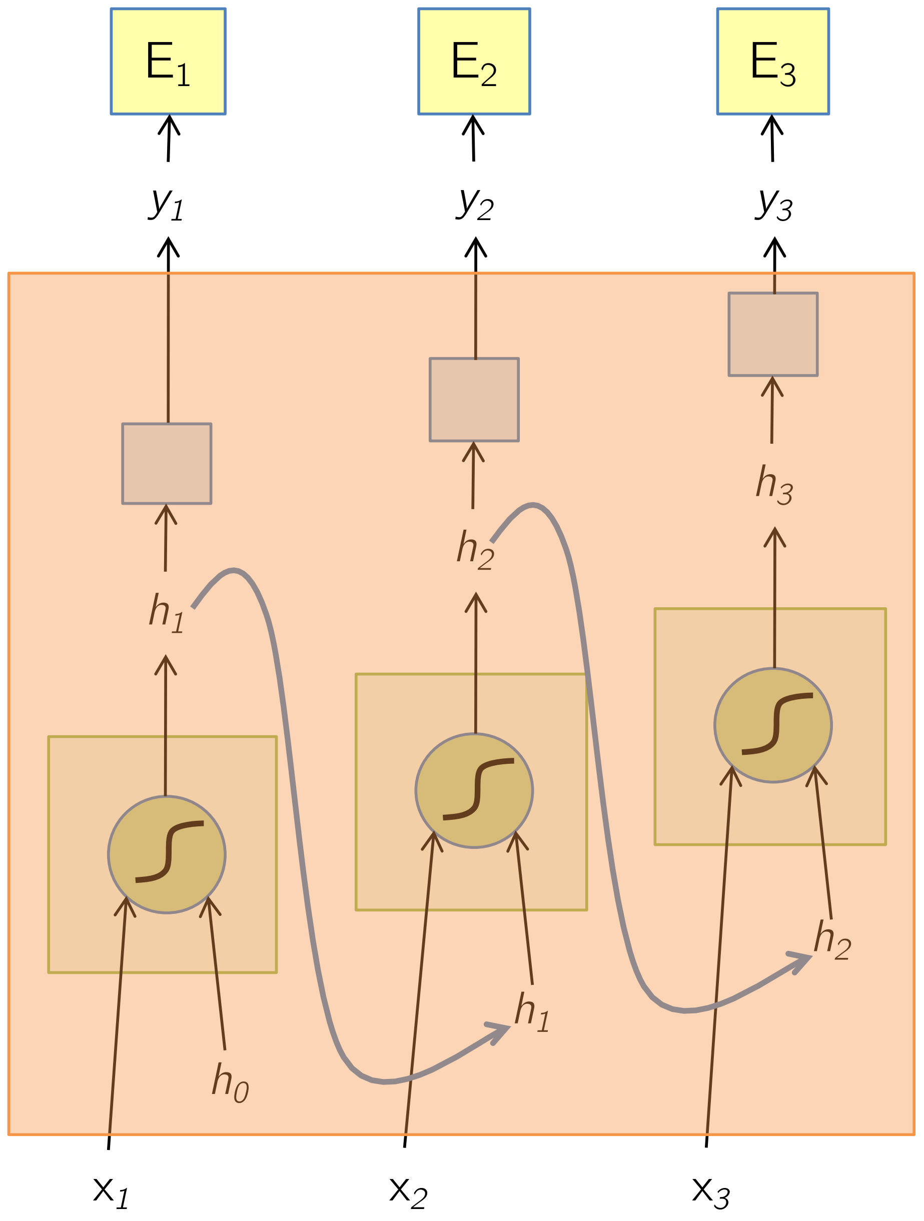
Sentiment classification

Multiple - Multiple



Translation

Backpropagation though Time



- Treat the unfolded network as one big feed-forward network
- This unfolded network accepts the whole time series as input
- The weight update is computed for each copy in the unfolded network, then summed (or averaged) and then applied to RNN weights

An Issue With RNNs

Recall, back propagation requires computation of gradients with respect to each variable. For RNNs, consider the gradient of **E$_t$** with respect to **h$_1$**

$$\begin{aligned}\frac{\delta E_t}{h_1} &= \left(\frac{\delta E_t}{\delta y_t}\right)\left(\frac{\delta y_t}{\delta h_1}\right)\\ &= \left(\frac{\delta E_t}{\delta y_t}\right)\left(\frac{\delta y_t}{\delta h_t}\right)\left(\frac{\delta h_t}{\delta h_{t-1}}\right)\cdots\left(\frac{\delta h_2}{\delta h_1}\right)\end{aligned}$$

Product of a lot of terms can shrink to zero (**vanishing gradient**) or explode to infinity (**exploding gradient**). Exploding gradients are often controlled by clipping the values to a max value. One way to handle vanishing gradient problem is to try to have some relationship between **h$_t$** and **h$_{t-1}$** such that each $\left(\frac{\delta h_t}{\delta h_{t-1}}\right) = 1$

Long Short Term Memory (LSTM) try to do that, but vanishing gradients still a problem. As a result, capturing long range dependencies is still challenging.

Summary

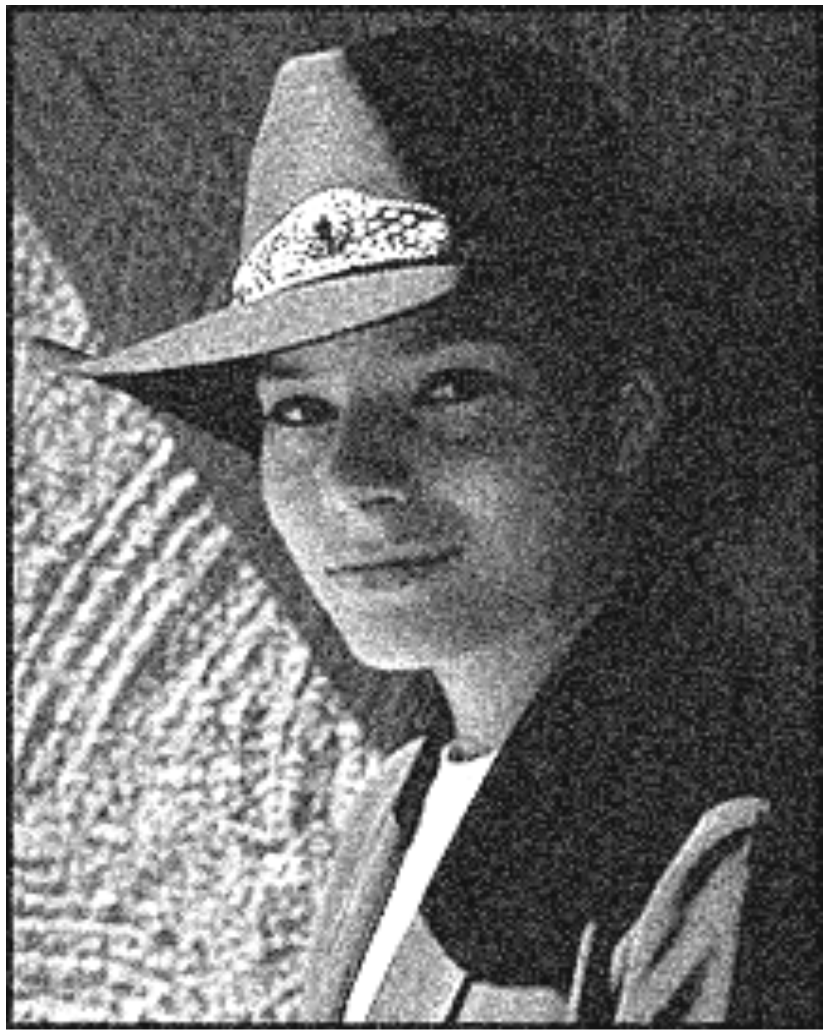
- RNNs allow for processing of variable length inputs and outputs by maintaining state information across time steps
- Various input / output scenarios possible (Single / Multiple)
- Exploding gradients are handled by gradient clipping, LSTMs are a way towards handling vanishing gradients.

You can read more about LSTMs at http://arunmallya.github.io/ . A major part of RNN slides were taken from there.

Questions ??

Convolutional Neural Networks

Motivation



Grayscale Image is rectangular matrix of pixel values (intensity values)

How to remove the noise in the photograph ?

This picture and many others taken from slides of Prof. Lana Lazebnik

Moving Average

- Lets replace each pixel with a weighted average of its neighborhood
- The weights are called the filter kernel
- What are the weights for the average of a 3 * 3 neighborhood ?

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

"box filter"

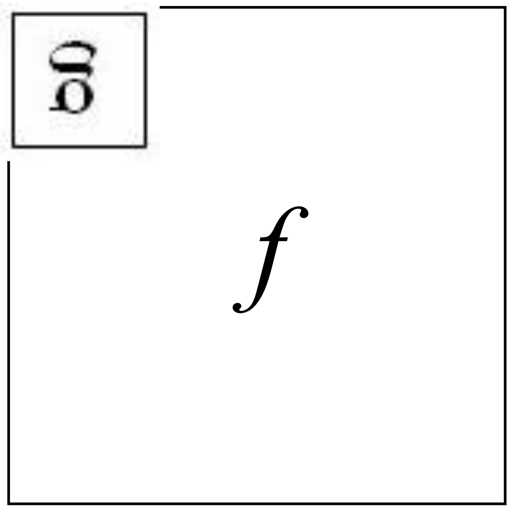
For each pixel **p**, place this matrix with the pixel **p** at the center. Perform point wise multiplication of matrix with pixel values, and average them. Make this the new value of pixel **p.**

Convolution

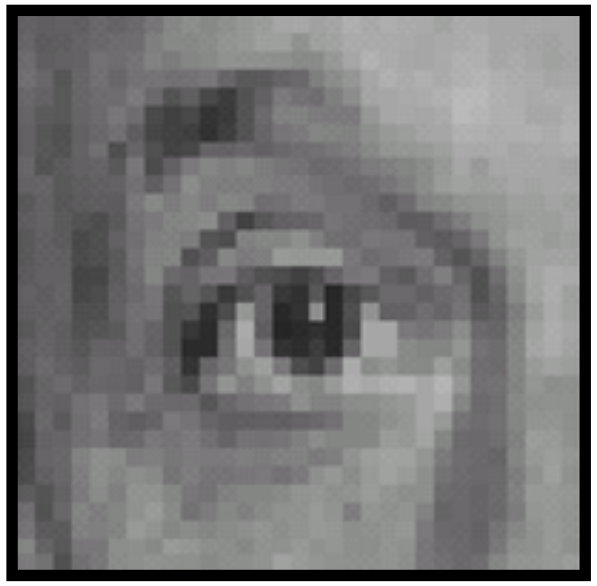
Let **f** be the image and **g** be the kernel. The output of convolving **f** with **g** is denoted **f*g**

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l]\, g[k,l]$$

Convention:
kernel is flipped

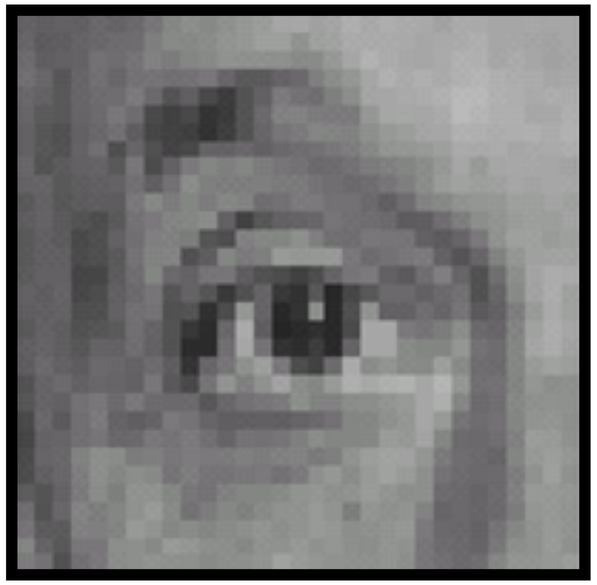


Practice with Linear Filters



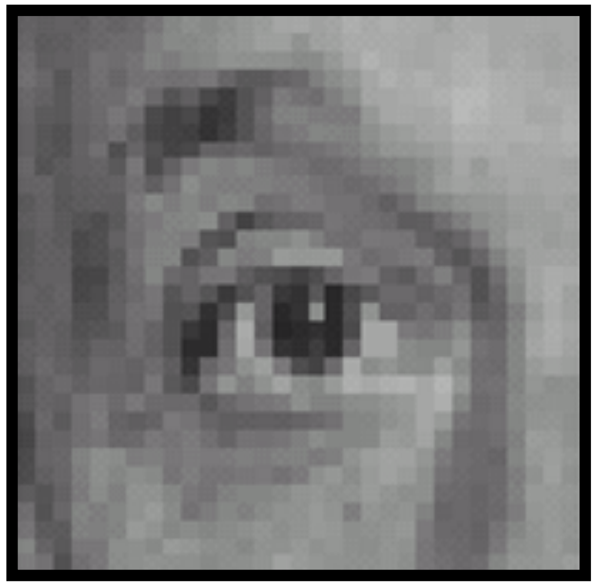
Original

0	0	0
0	1	0
0	0	0



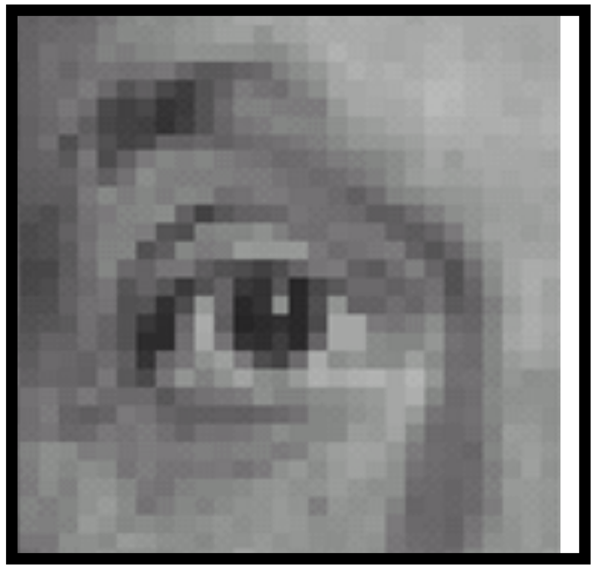
Filtered (No change)

Practice with Linear Filters



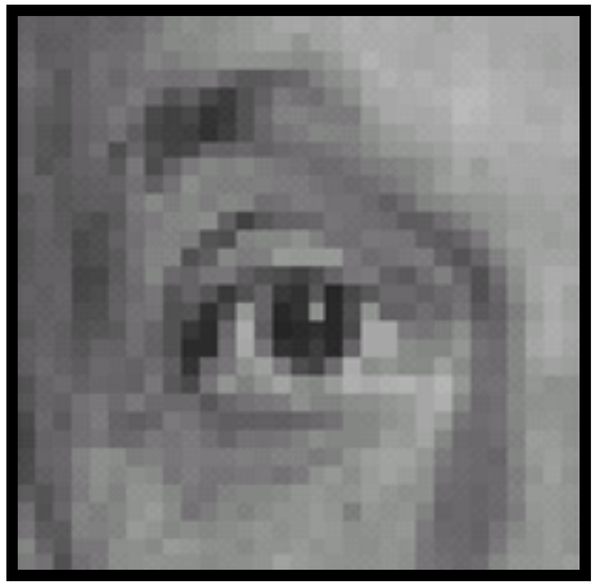
Original

0	0	0
0	0	1
0	0	0



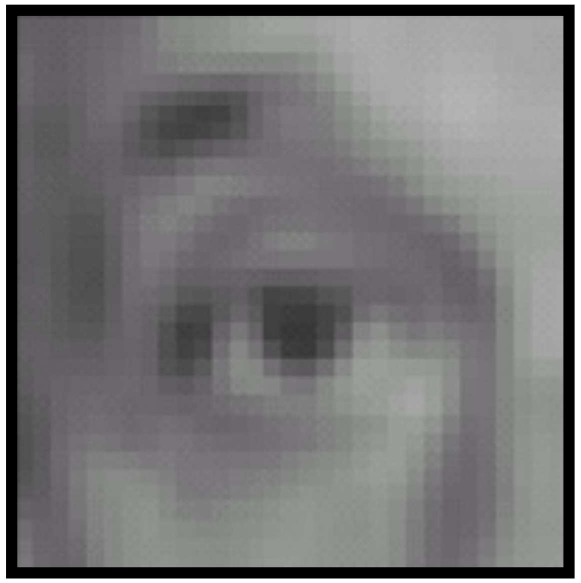
Shifted left by 1 pixel

Practice with Linear Filters



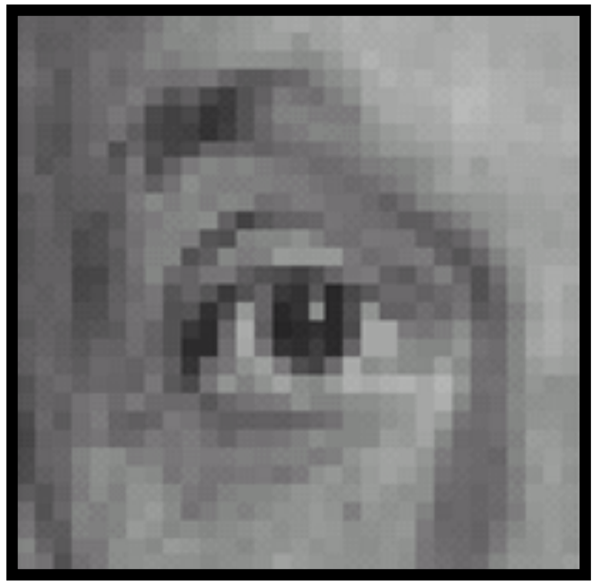
Original

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

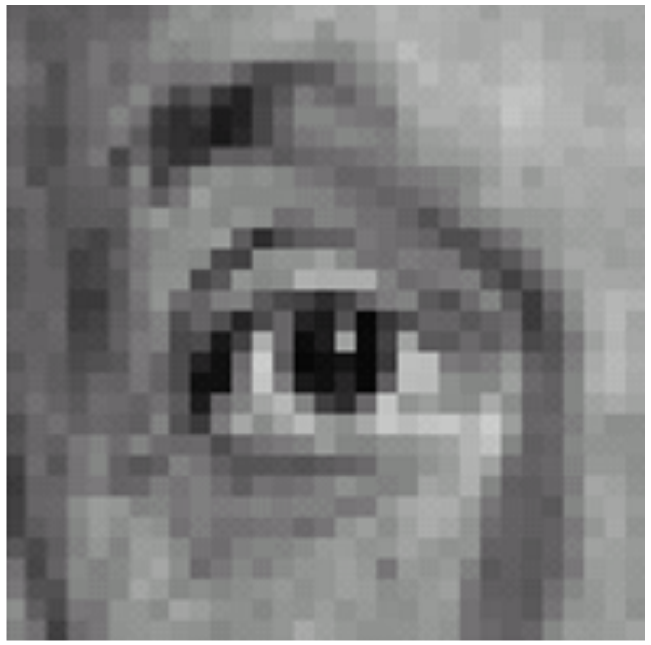


Blur (with a box filter)

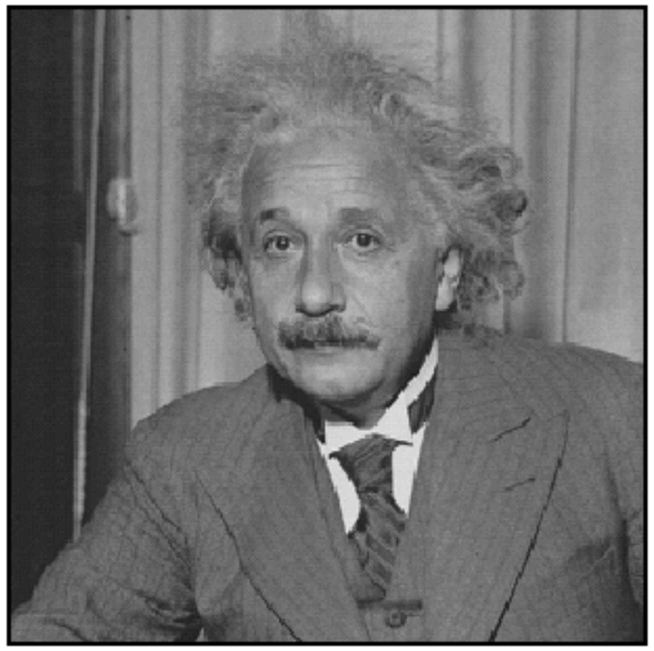
Practice with Linear Filters

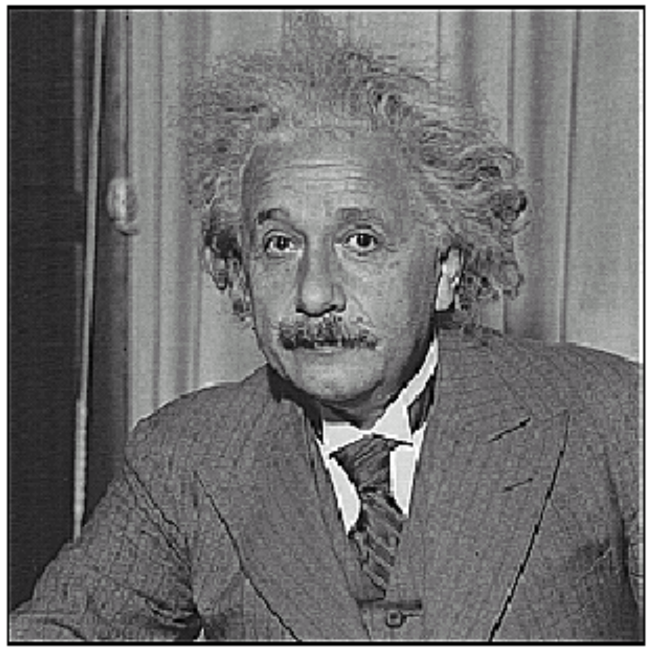

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$


Sharpening filter - increases differences with local average

Sharpening Filter

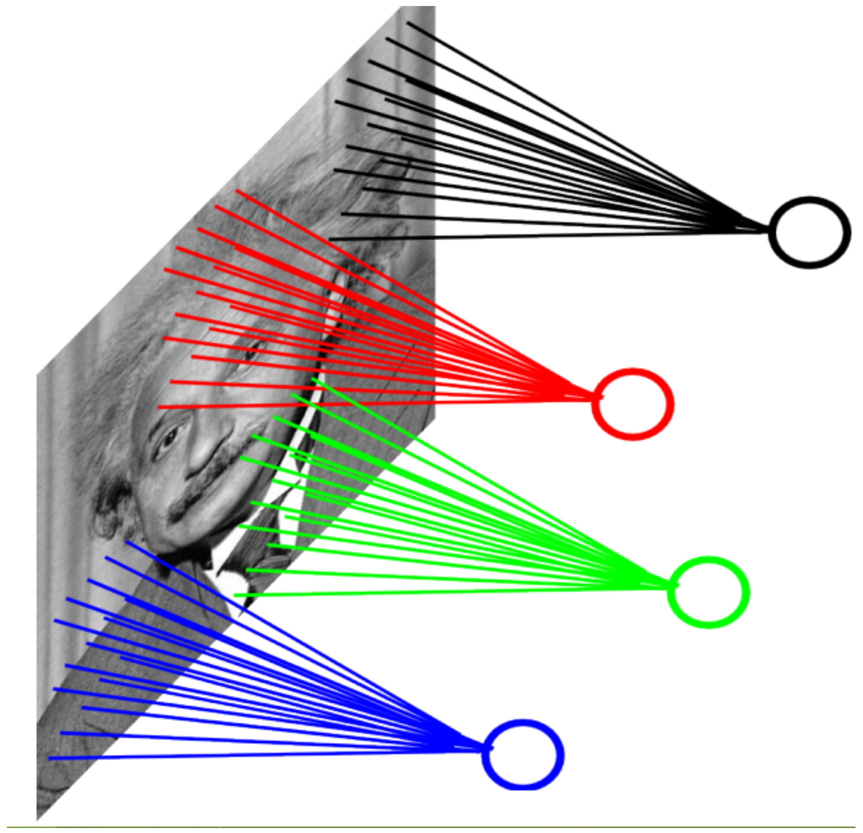

before


after

Filters can be used for sharpening, edge detection and extracting many other properties. See https://en.wikipedia.org/wiki/Kernel_(image_processing) for more examples.

Main idea behind Convolutional Neural Networks: **use learned filters**

Convolution Layer



Each local region of the image is connected to a node in the next layer

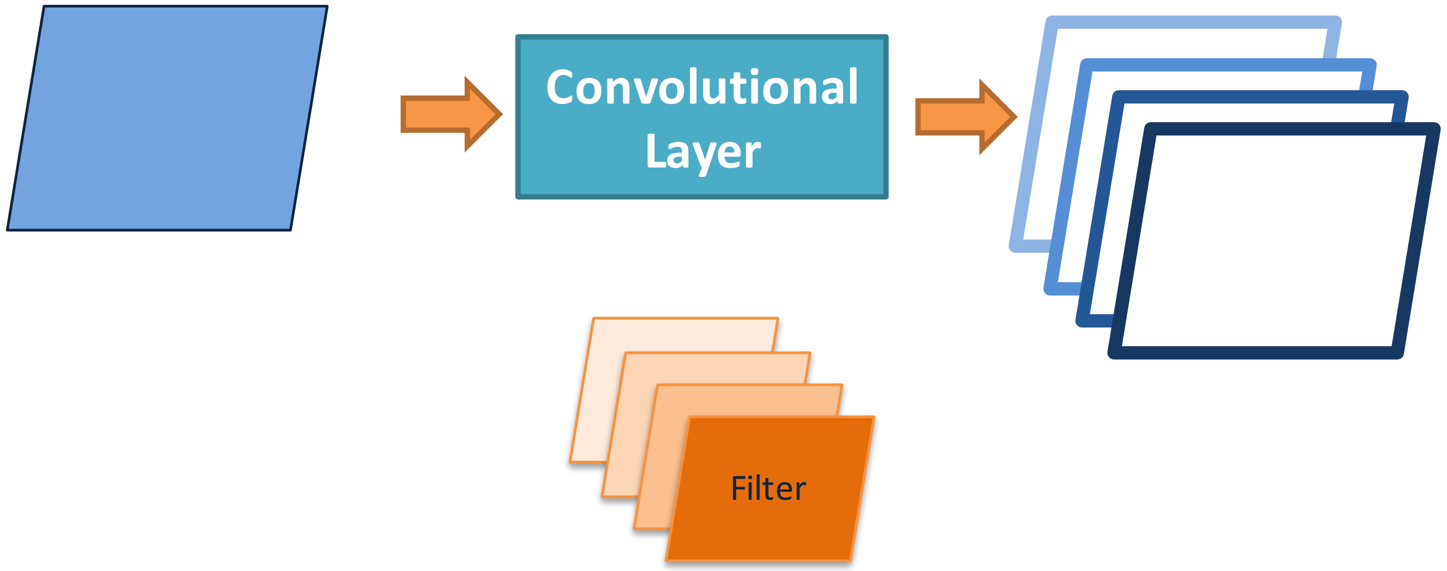
Weights are shared. Distribution of weight in the red connections will be the same as the weight distribution of the green connections, and so on, because they represent the same filter.

Much less number of parameters than fully connected layers we saw in last class

Convolution Layer

You can also use multiple filters on your image, all of them will be learnt

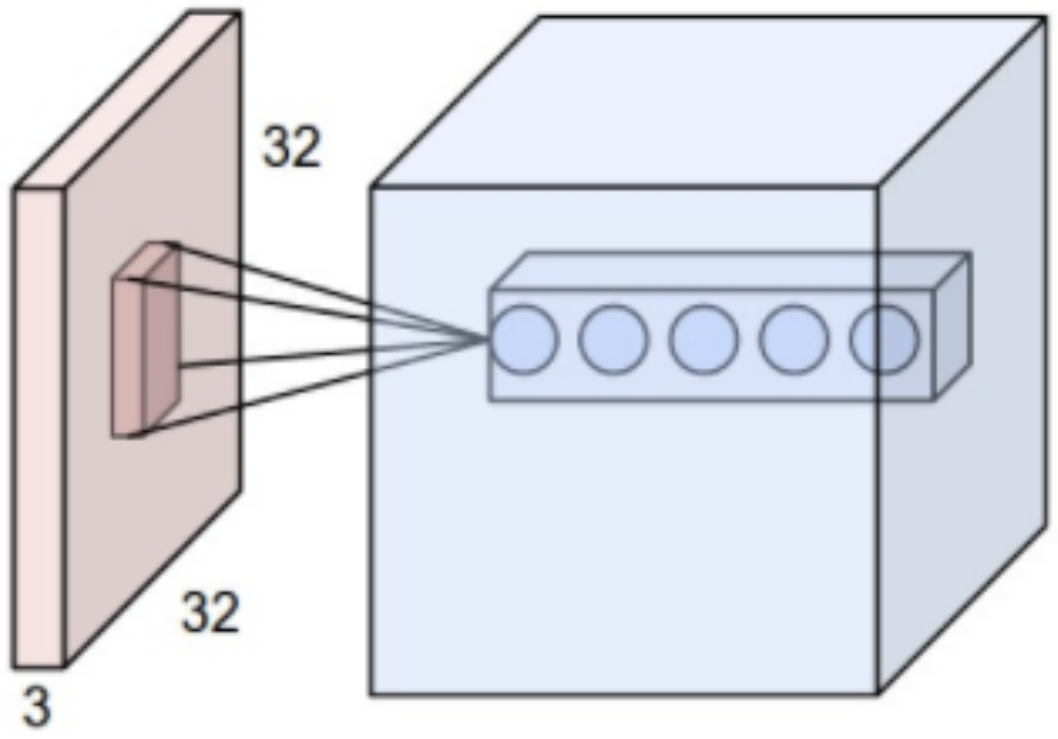
You can add non-linearity at the output of a convolutional layer



Convolution Layer

Colored images have 3 channels for RGB, so the input is a 3D matrix instead of 2D matrix. How to compute convolution there ?

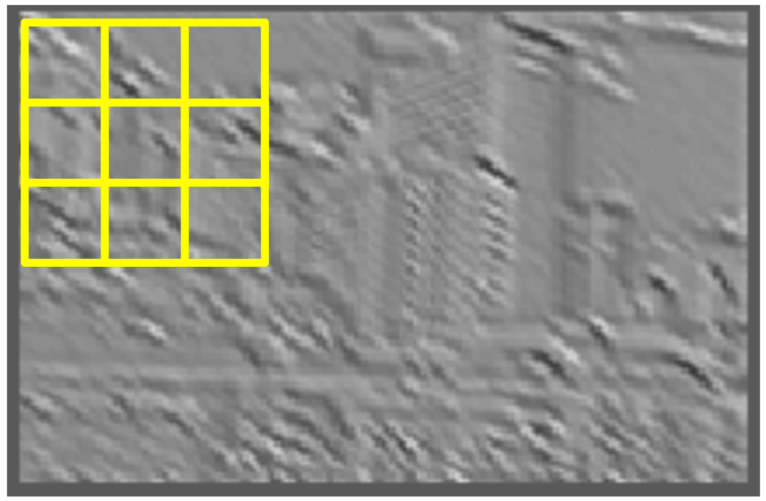
You will now need to perform 2D convolution with a 3D filter / kernel matrix. The third dimension of the filter matrix has to match the number of channels.

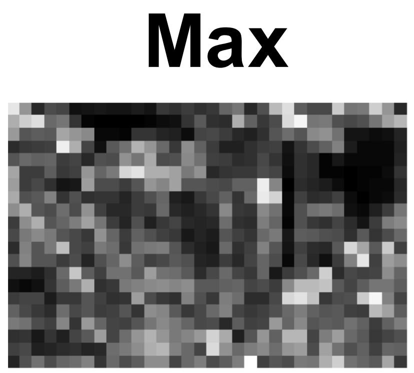


Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels).

Questions ??

Pooling Layer





Look at each window and compute max value

Function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting

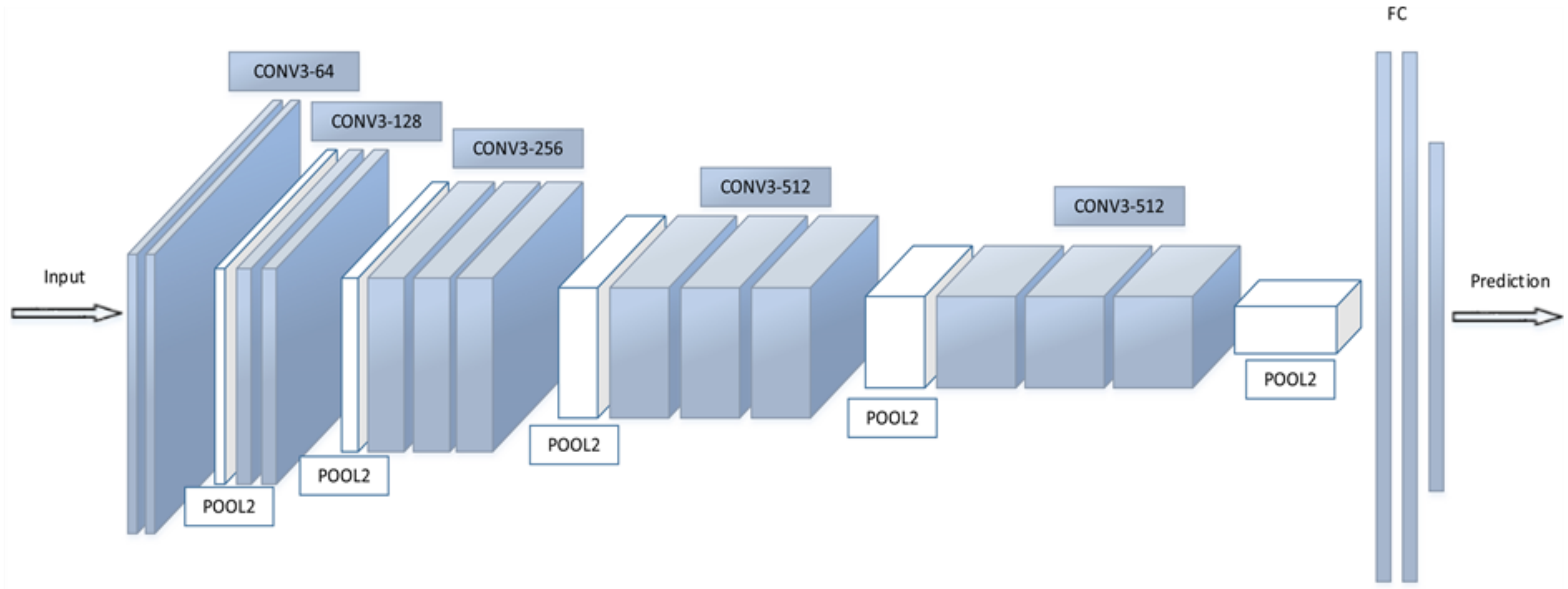
You can use other functions for pooling, like average, L2 norm, etc. but **max** is the most popular now

Some Facts

- Both the convolutional layer and the pooling layer can reduce the size of the input. How ?
- **Stride :** You do not have to apply filter to each consecutive pixel. You can only apply filter once every 2 pixels **(stride = 2)**, etc.
- Usually convolution layers use stride of 1, so does not reduce image response size
- Pooling layers use stride of 2, halves the height and width of image response.

Deep CNNs usually apply multiple convolution and pooling layers to compute rich features as well as reduce the image/ response size

An example: VGG

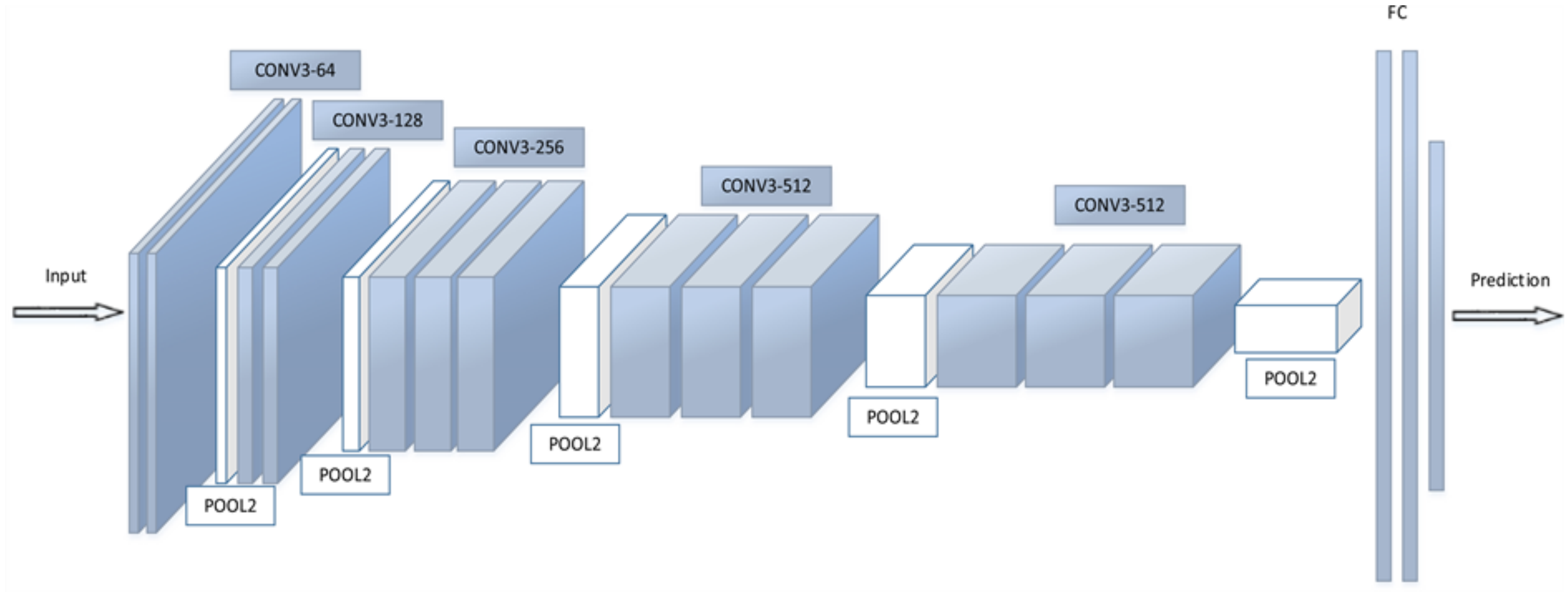


Thickness of each block corresponds to number of channels of the response / output. Height and width correspond to height and width of response / output.

Think of this as stacking several feature extractors, higher stages compute more global, more invariant features

138 million parameters

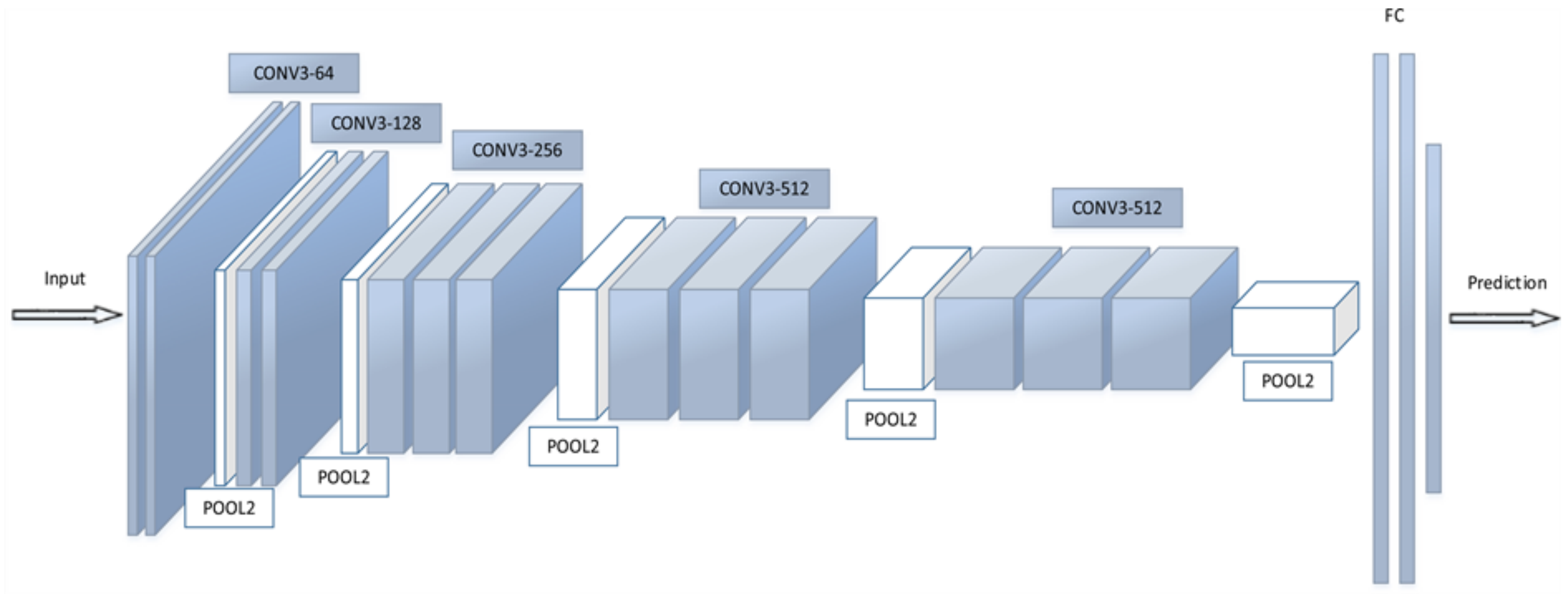
An example: VGG



CONV 3-64 : convolutional layer with filters 3*3*(number of input channels), and 64 such filters are used. Each filter output also passed through **Relu activation**.

Convolutions have stride of 1, so size remains same.

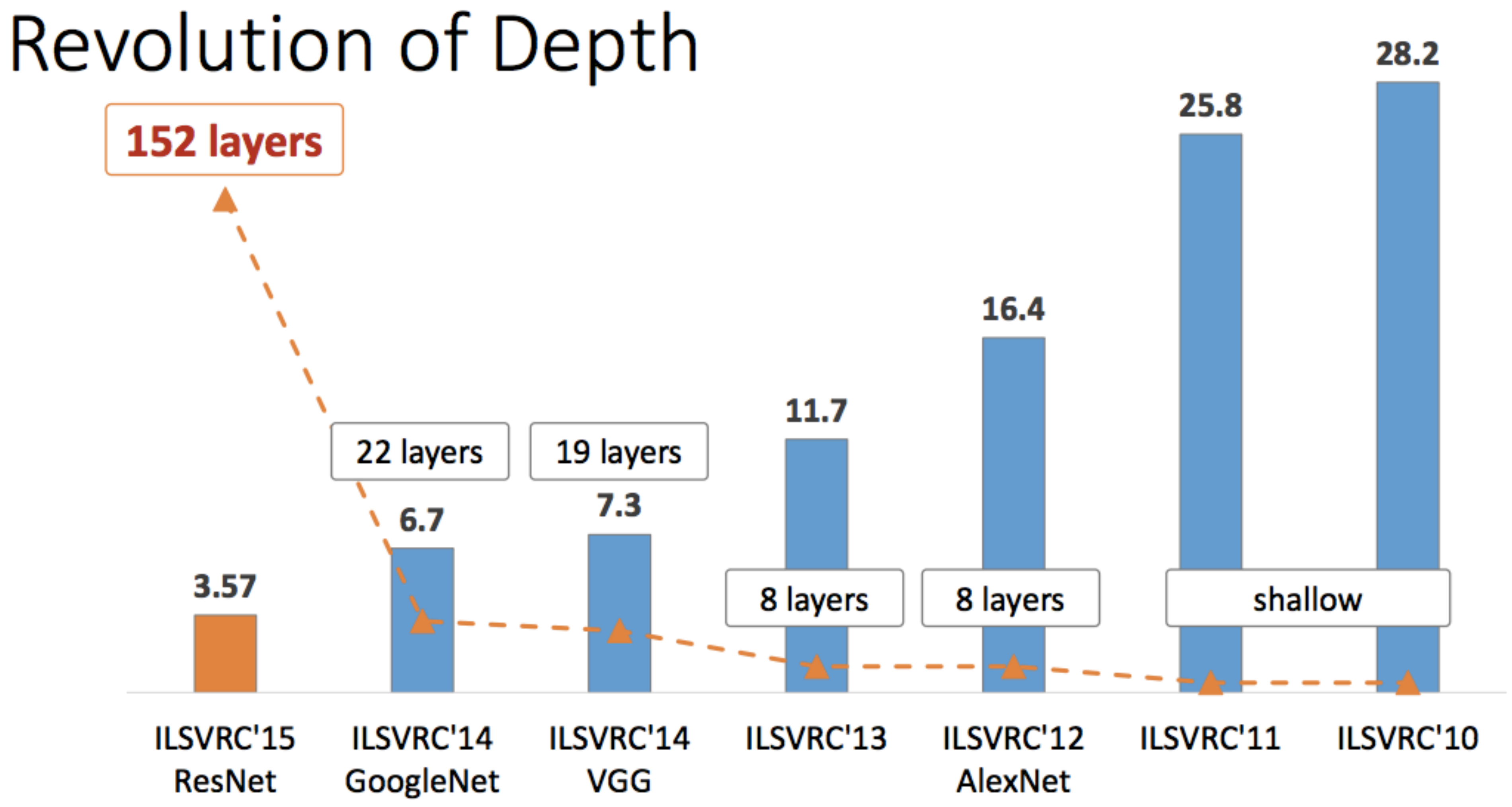
An example: VGG



POOL2 : Max pooling done on windows of 2*2 with **stride of 2**, so the height, width gets halved

FC : indicates fully connected layer, as you saw in standard feed forward network

Object Detection Performance



ImageNet Challenge Top 5 error

Summary

- Filters are used to extract different properties of an image
- Convolution neural networks use multiple layers of learned filters (convolution layers)
- Convolution layers used along side pooling layers (eg.VGG net)
- Each layer may or may not have parameters (e.g. CONV / FC do, RELU / POOL don’t)