Advanced Neural Nets

Welcome to the edge of data science

Brief review so far

- We studied a DENSE neural net or a multilayer perceptron
  - Very good for many general problems that can be solved by our usual bag of tricks (Regression, Forests, Bayes, Boosting)
  - Homework is a MLP (Dense) net
  - All outputs connected to all inputs
    - Use Tanh, Sigmoid, Relu etc…

How to ‘Transfer’ a Model

- Export Weights and Bias
- Similar to a Linear regression: Instead of a column of “M” and “B” each layer has a rectangular “M” of size Inputs x Neurons (inputs of internal layers will be neurons of previous layers and a column vector of bias)
- You need activation functions used
- Typically implemented within a library

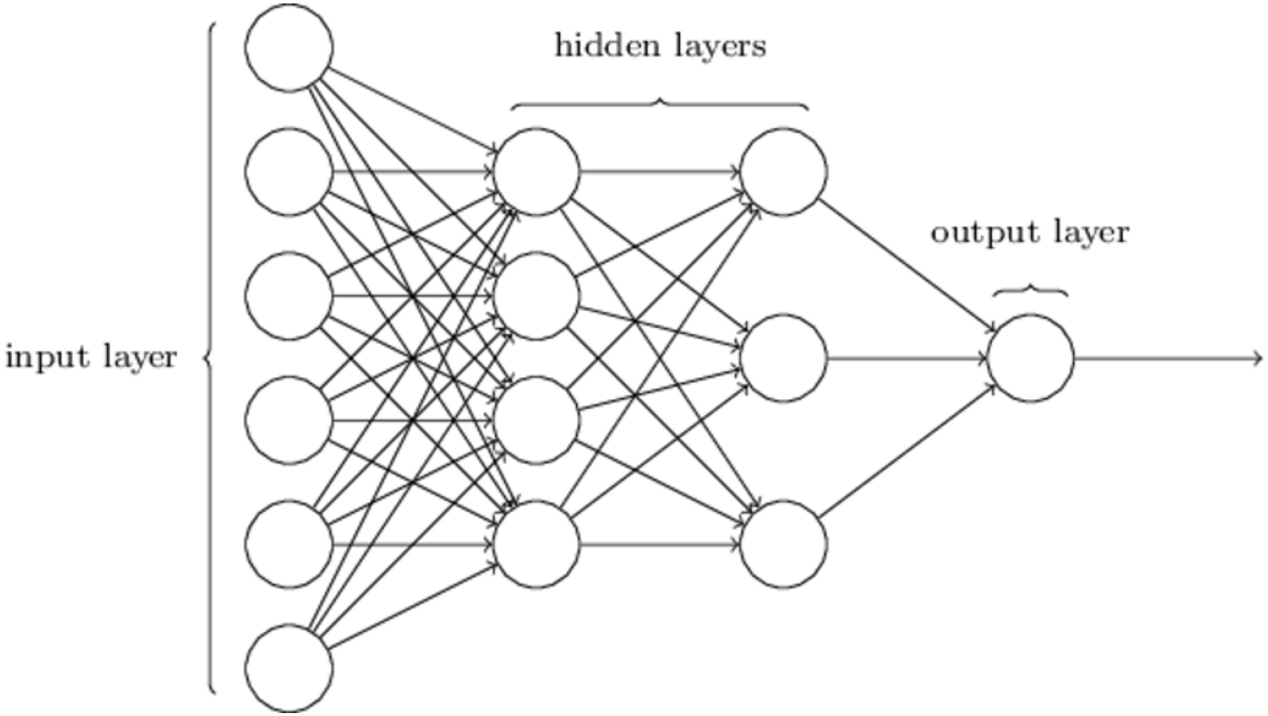
Example

(Indicies/Rows+Cols can be flipped depending on your library)

W is 4x6
B is 4x1

W is 3x4
B is 3x1

W is 1x3
B is 1x1



But wait! There’s more!

- Recurrent Neural Net (RNN)
- The Convolution Neural Net (CNN)

How do I model something temporal?

(THAT MEANS TIME-BASED)

- Language
- Videos (Movie)
- TIME SERIES MODELING

Recurrent Neural Network

- Used in Natural Language Processing. (Alexa, Google, Siri)
- SOLVES THE PROBLEM: MY CURRENT OUTPUT DEPENDS ON THE PAST OUTPUT:

  PREDICT THE NEXT WORD:

  ROBERT WEIGHS ABOUT 200 _____ (POUNDS)

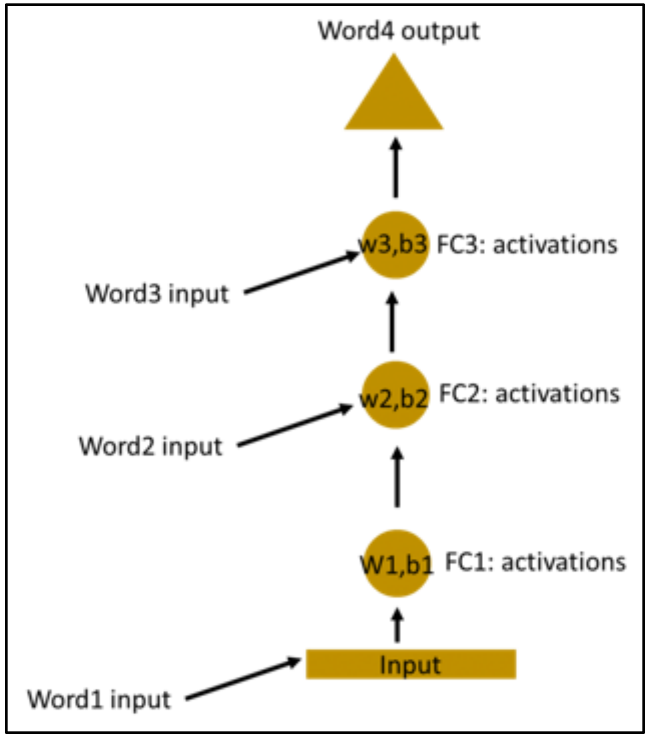
  THE ANSWER DEPENDS ON THE PREVIOUS INPUTS

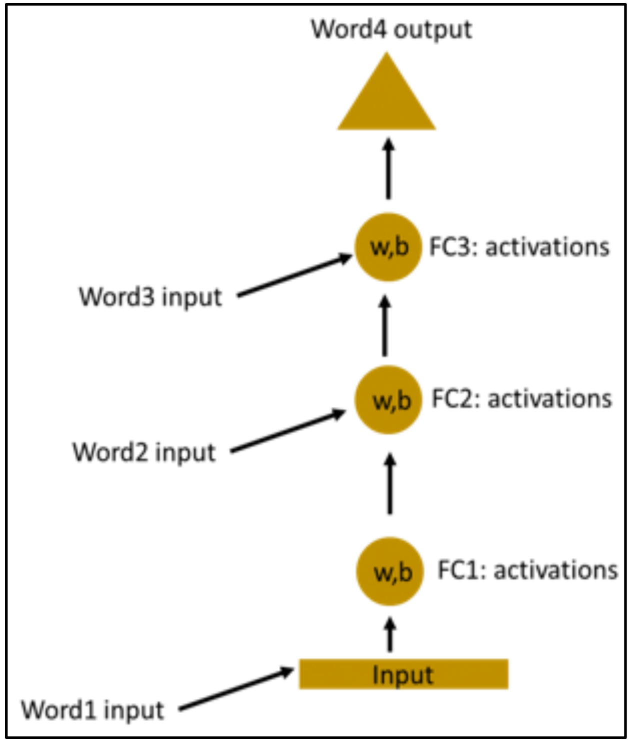
  WHAT PART OF TEXT IS EACH WORD?

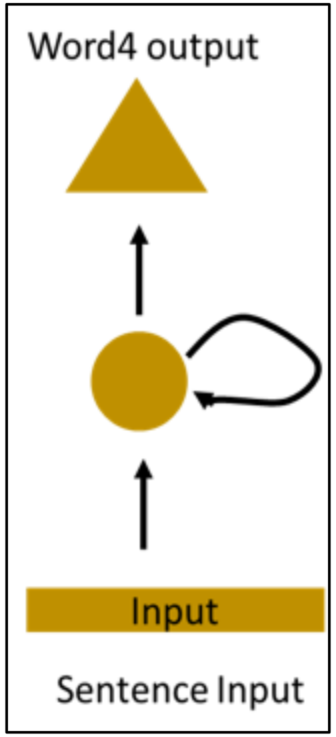
  ROBERT WEIGHS ABOUT 200 POUNDS

  ABOUT 200 POUNDS ROBERT WEIGHS

How is this pulled off?





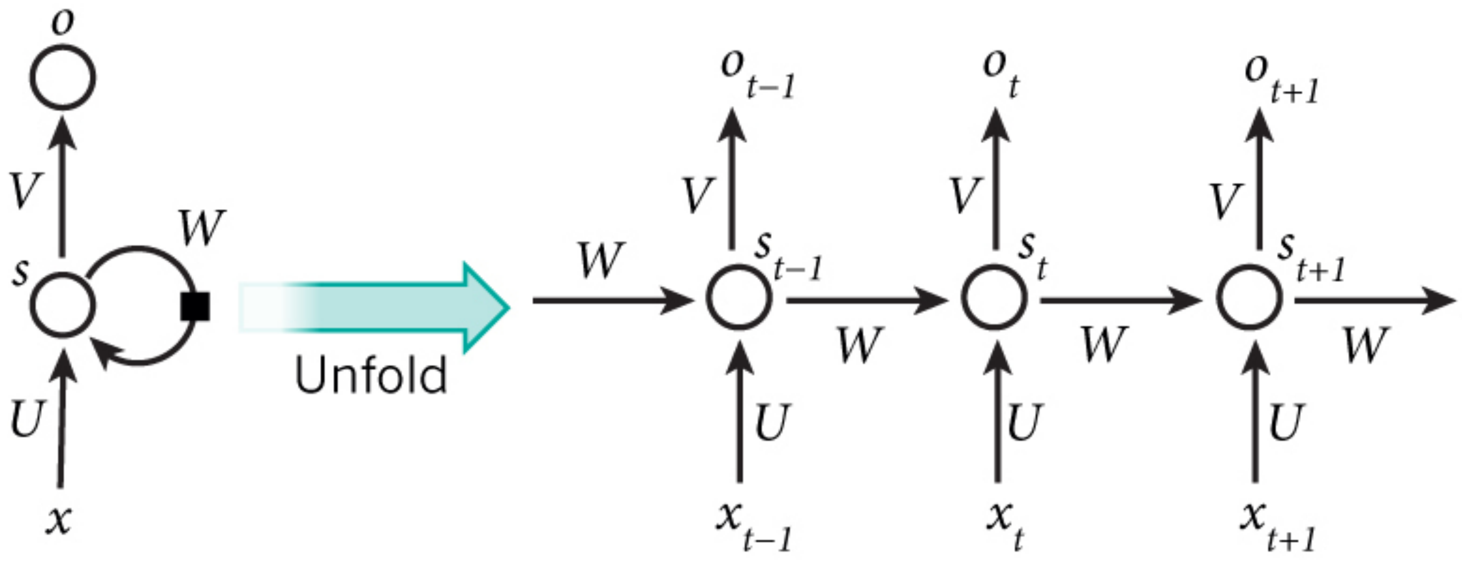


Another view

Think of these diagrams as ‘FLOW’ rather than a diagram

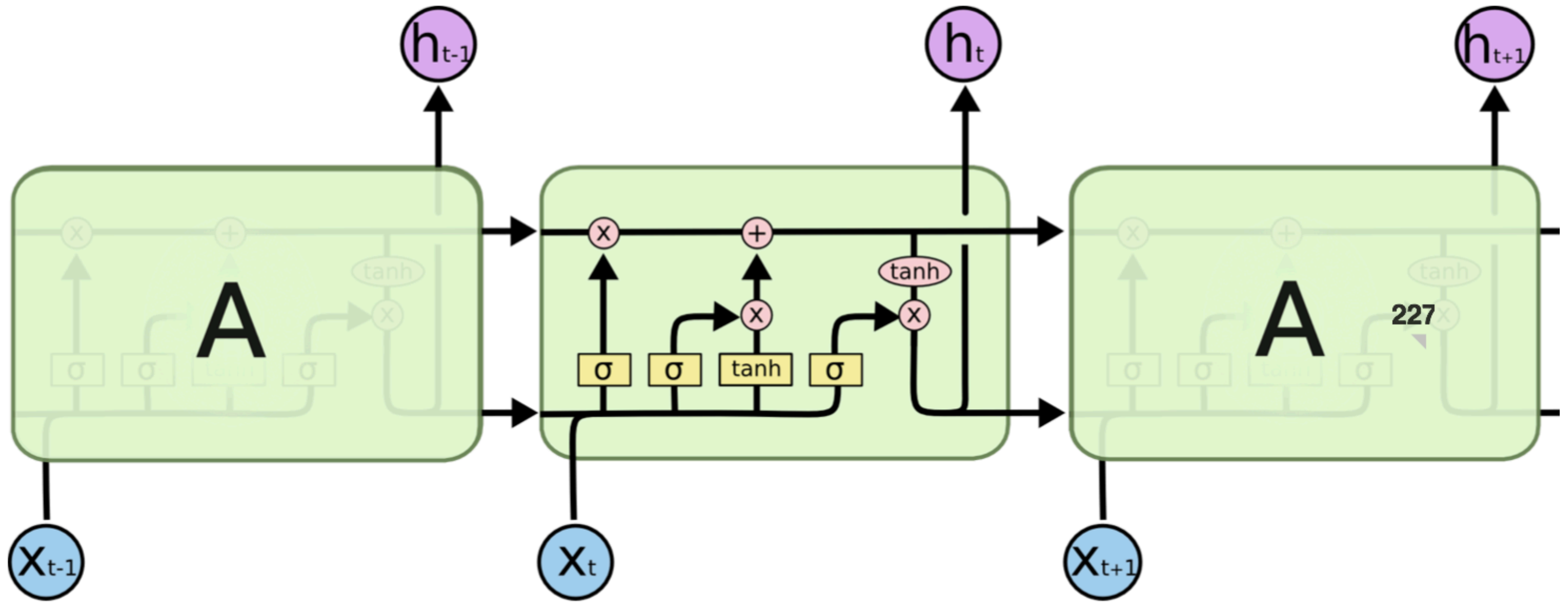
V,U,W: Neuron Weights
S: hidden state
X: input
O: output

Take the OUTPUT of a neural net and feed it back as new inputs for the next ‘state’. The initial hidden state is typically all zeros



What glorious Instructor does:

Long Short Term Memory (LSTM). Change ‘W’ to a more complex algebra



Approve, Yoda does

- Bi directional RNN
  - Run the RNN Forwards AND Backwards to capture both directions of context.

Convolutional Neural Nets (CNN)

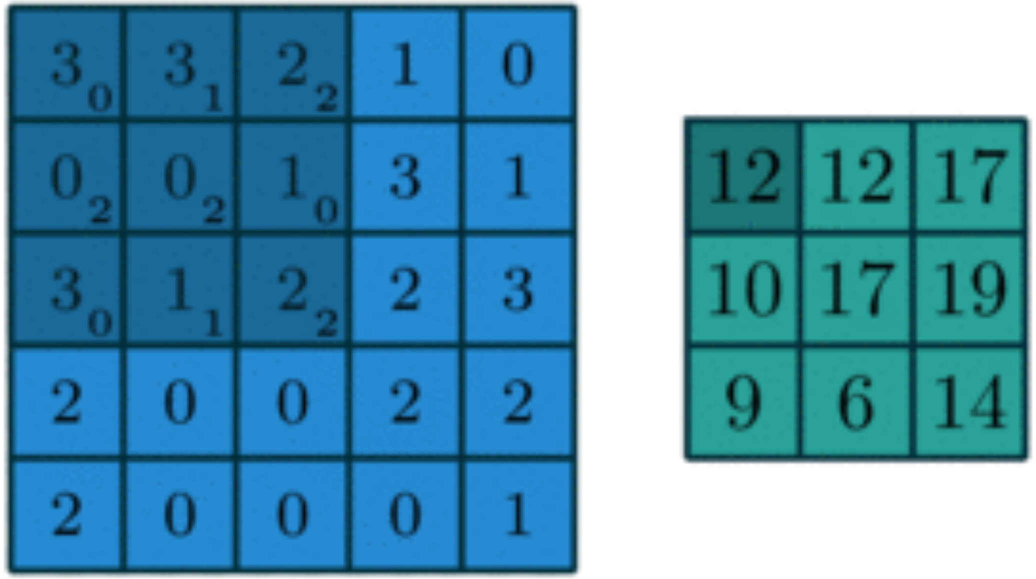
- Image recognition
- Based on ‘compressing’ information of nearby neurons

We need 2 new layers

- Convolution Layer
- Pooling Layer

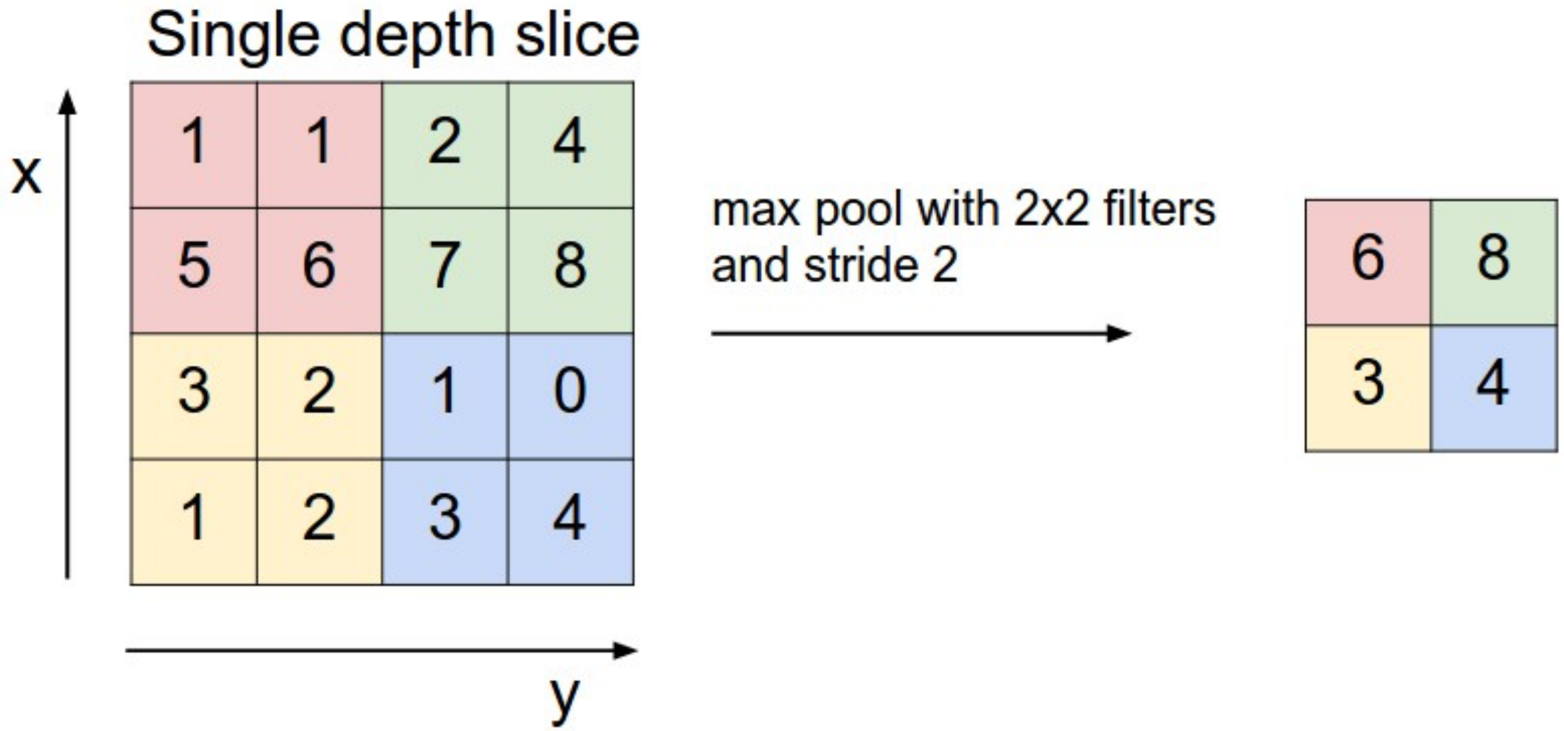
Convolution Layer

- Slide a filter across your input (typically 2-D)

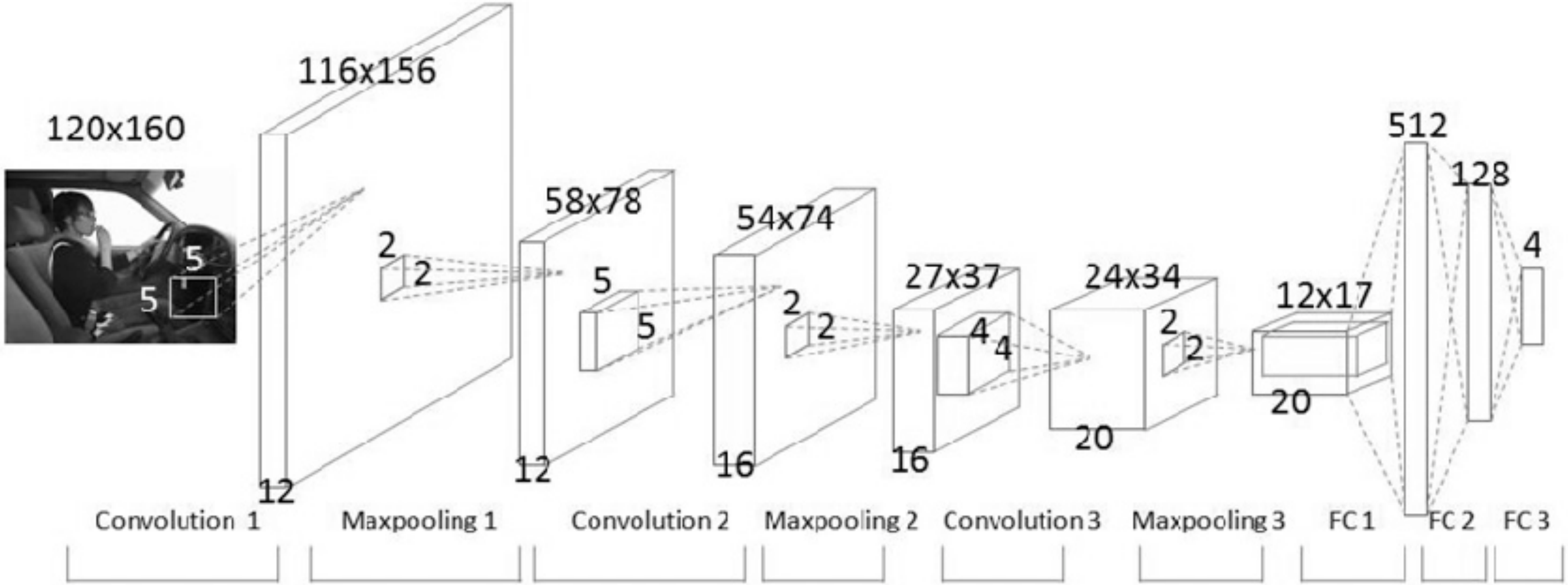


Pooling layer

(Find the max value of a filter)



Putting it together



Transfer Learning

- For deep neural Nets (4+ layers), The ‘Top’ Layers (Closest to input) do not change much.
- After a long training cycle they are essentially fixed.
- SO when google uses 8 GPUs to Train on 10 million images they have a very solid set of ‘TOP’ layers
- Turns out you can ‘attach’ bottom layers for a related problem

Example

- VGG16 (a particular CNN Architecture) is trained on IMAGENET (a public data set)
- Imagenet has 1000 classes
- Take weights, remove the final Dense portion / Create your own to train
- Achieve 99% accuracy in less than 1 epoch!

Transfer Learning

- Works for all Deep nets due to diminishing gradients
- Primarily used in vision (CNN)
- Still works with other architectures

Resources

- http://cs231n.github.io/
- http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- http://scs.ryerson.ca/~aharley/vis/conv/
- http://ruder.io/optimizing-gradient-descent/