# Convolutional Networks

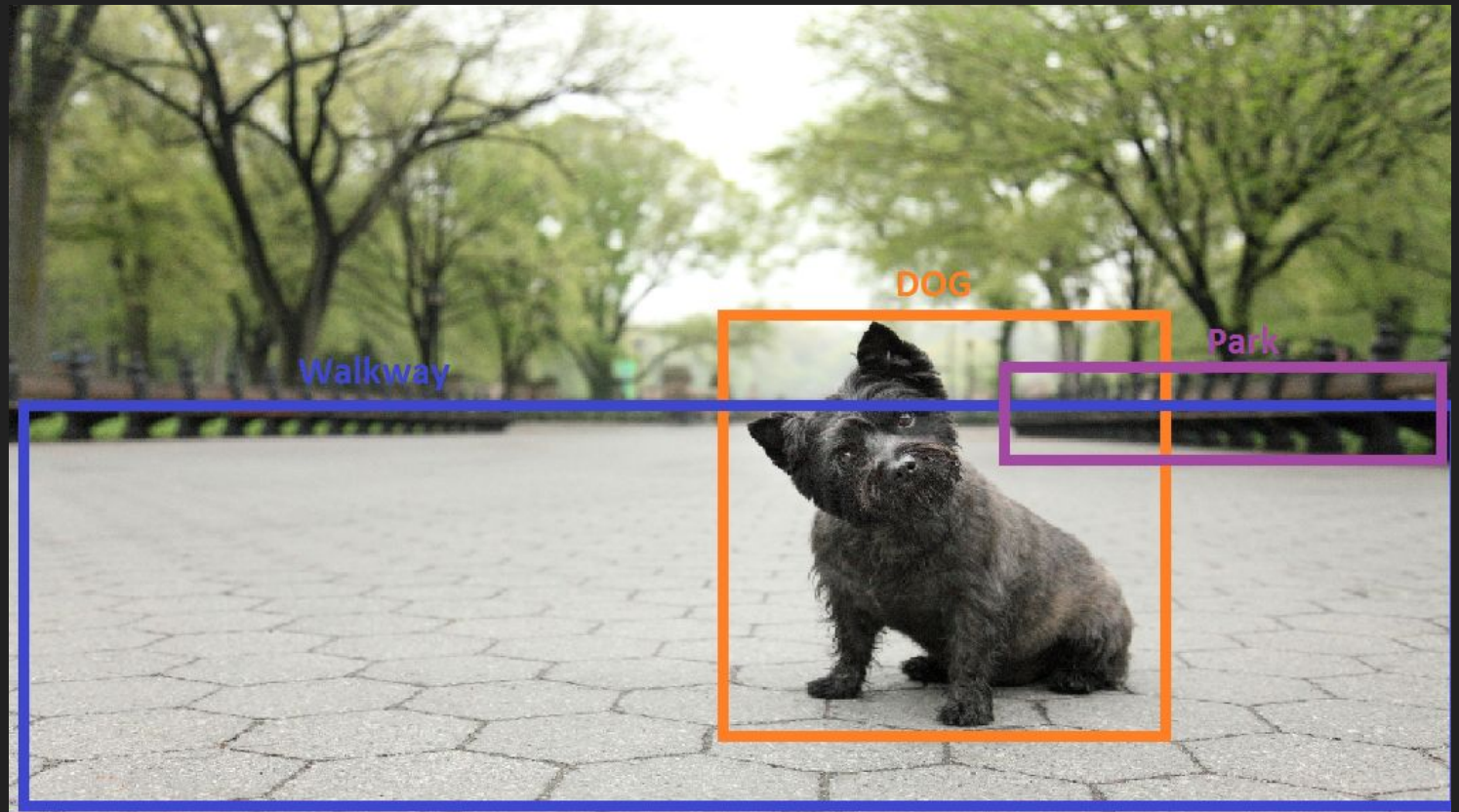
# Why is image recognition hard?

# Why is image recognition hard?

- Just a few examples of what makes it hard:
  - Rotation
  - Lighting
  - Distance
  - How much of the object is covered by another one
- Even harder to distinguish tiny details
  - One person from another?

# How would you describe this image?

We find features through selective *attention* and then look for relationships between those features



# The general idea

- “Having learned features over small (say 8x8) patches sampled randomly from the larger image, we can then apply this learned feature detector anywhere in the image.”
- Learning a smiley face example
  - Learn top-right, top-left, bottom-right and bottom-left of smiley face
  - If we find each of those in the correct positions relative to one another, then we have a smiley face

Can regular feed-forward networks do this?

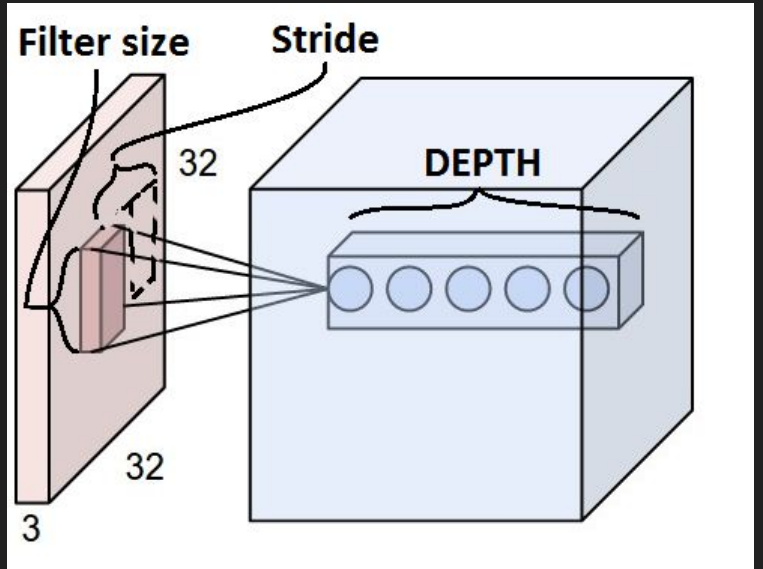
# No! So we switch their architecture

- Let’s say we have 32x32 pixel images (CIFAR-10 database)
  - 32x32x3 inputs to include RGB
- We will stay in 3-dimensions until the output layer
- Vocab
  - **Filter Size**: The size of our *attention* square
  - **Depth**: The 3rd dimension of our hidden layer
  - **Stride**: How much we move our attention square over by
  - **Zero-Padding**: Padding the image with a black perimeter to make the hidden layer sizes clean

# Let’s look at an example

- **Filter Size**: The size of our *attention* square
- **Depth**: The 3rd dimension of our hidden layer
- **Stride**: How much we move our attention square over by

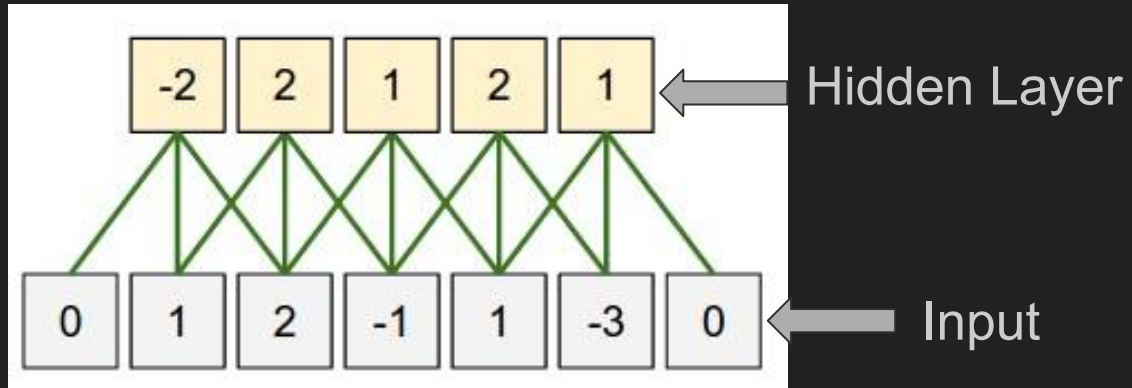
What purpose does this serve?



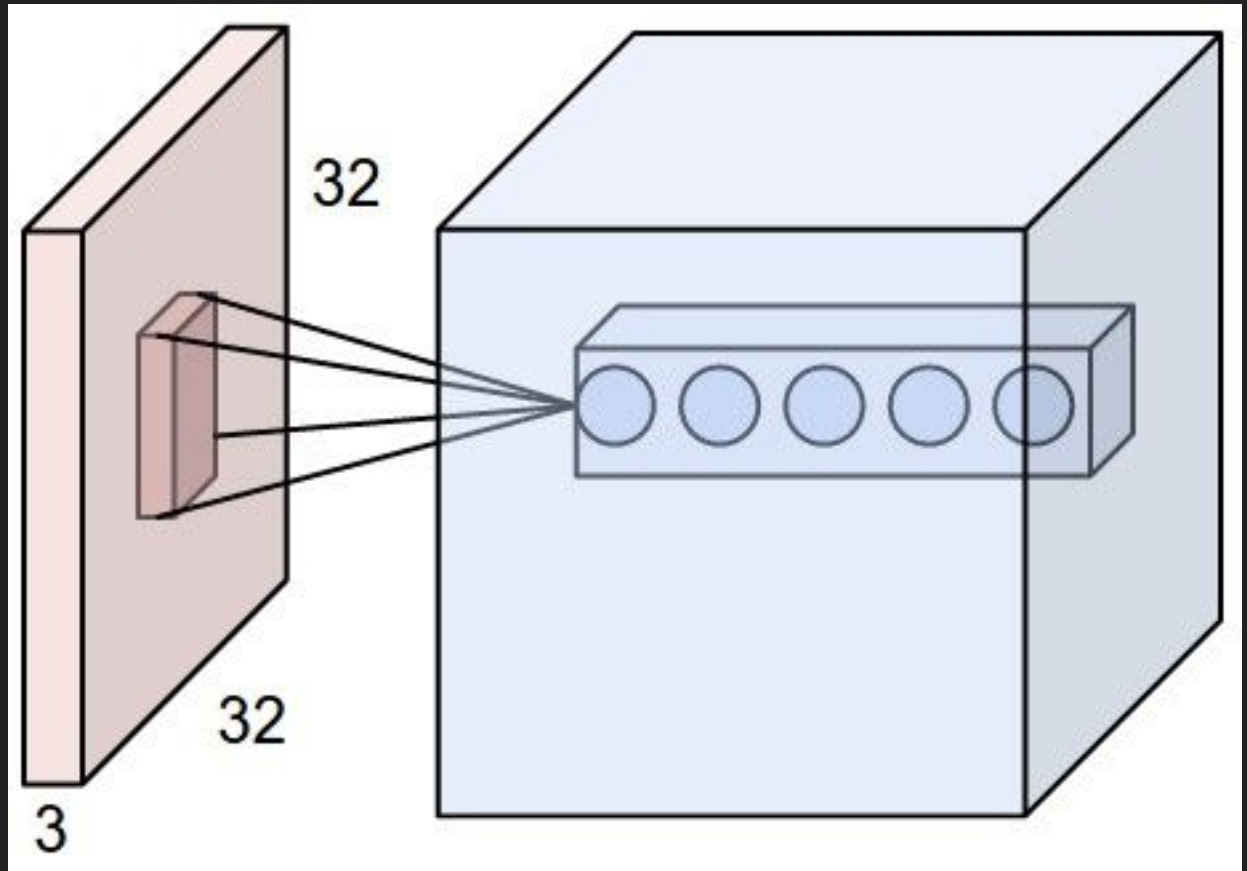
# Zero Padding

- We can pad the input layer with 0’s around the sides in order to maintain layer sizes throughout the network
- Can get a nice formula for output size as a function of input size, filter size, stride and zero-padding
- Set output size to what we want, solve for zero-padding

- This is showing the horizontal row with parameters:
  - **I**nput size = 5
  - **F**ilter size = 3
  - Depth not pictured
  - **S**tride = 1 (Stride of two shown in the reading for this week)
  - **Z**ero Padding = 1 (can solve equation for Z to get padding)
- The size of the hidden layer is H = ((I + 2 x Z) - F) / S + 1
  - Separately in the horizontal and vertical directions



# Why are we doing all of this again?

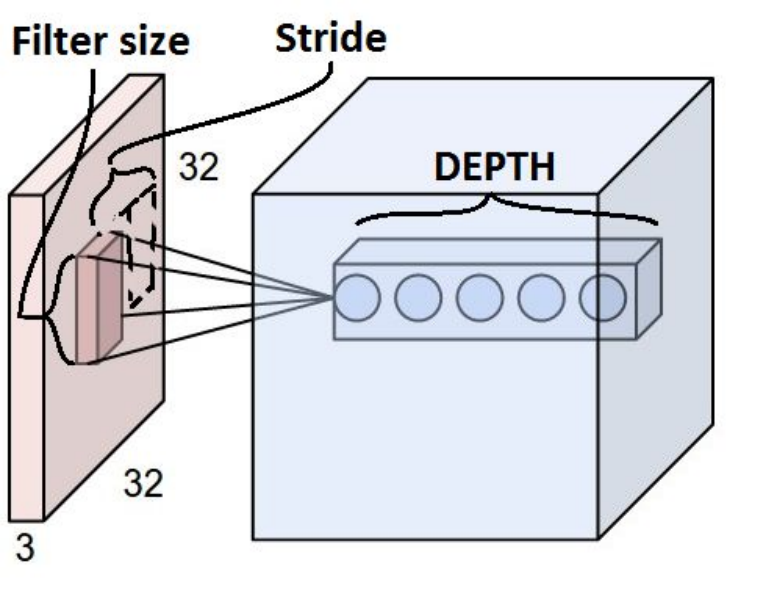


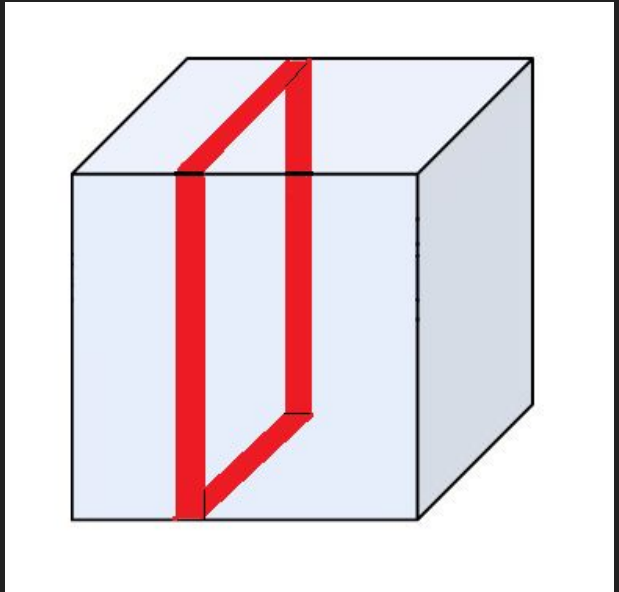
# But we have a problem

- Number of weights explodes!!!
- For example, a famous paper took 227x227x3 images
  - They did Filter Size = 11, Stride = 4, and Zero Padding = 0
  - Plug em into the equation and you get 55x55
  - Then they made a depth of 96, giving a hidden layer of 55x55x96
  - That’s 290,400 neurons in the hidden layer
    - Each has 11x11x3 connections, giving us a total of **105,705,600** weights
- That’s just for one layer!

# Parameter Sharing

- Need to reduce number of weights, so that it's easier to train
- Parameter Sharing: Each *depth slice* shares the same weights
- Depth slices responsible for specific set of features as a result



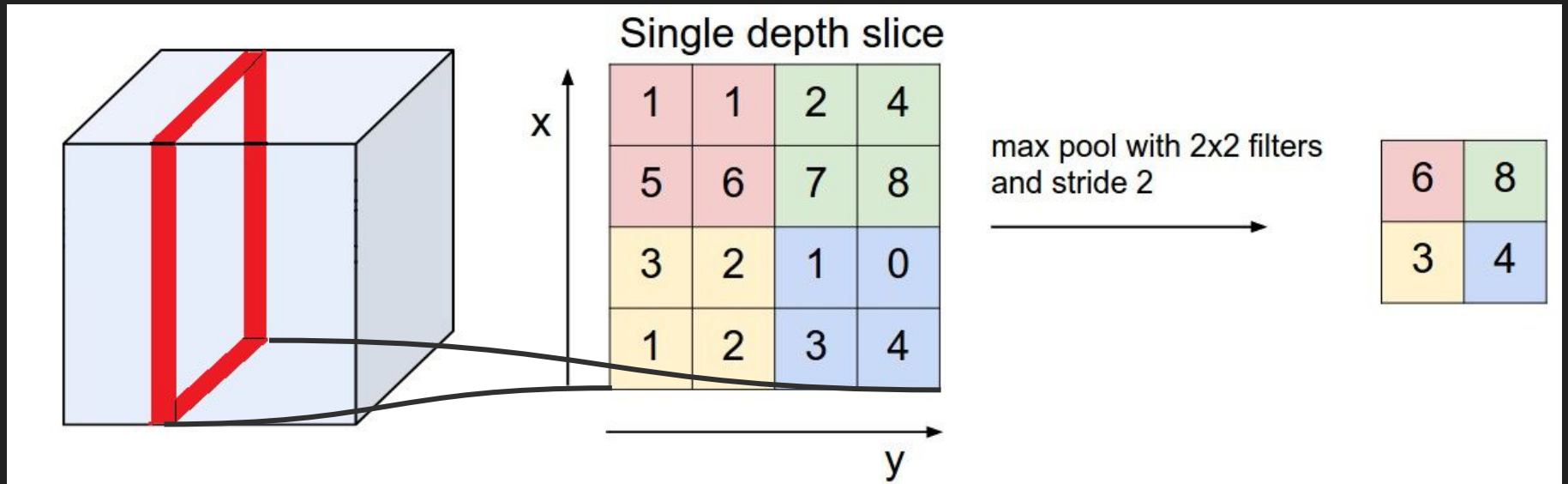



# Sizing it down

- This is still huge, though
- Input layer can be hundreds x hundreds of neurons, while the output layer is usually very small
- Pooling helps architecture reflect this

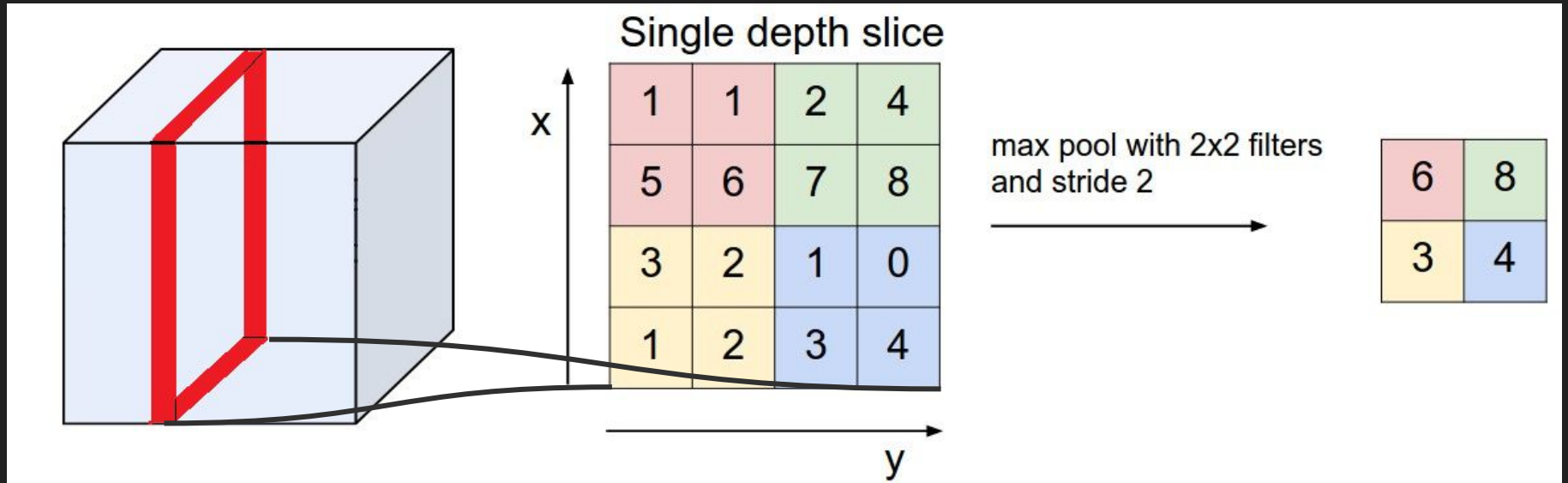
# Max-Pooling

- Remember, these numbers are no longer the image!
- They represent features learned by the image
- Same as when we did k-sparse autoencoding last week
  - Only take the x most important features found

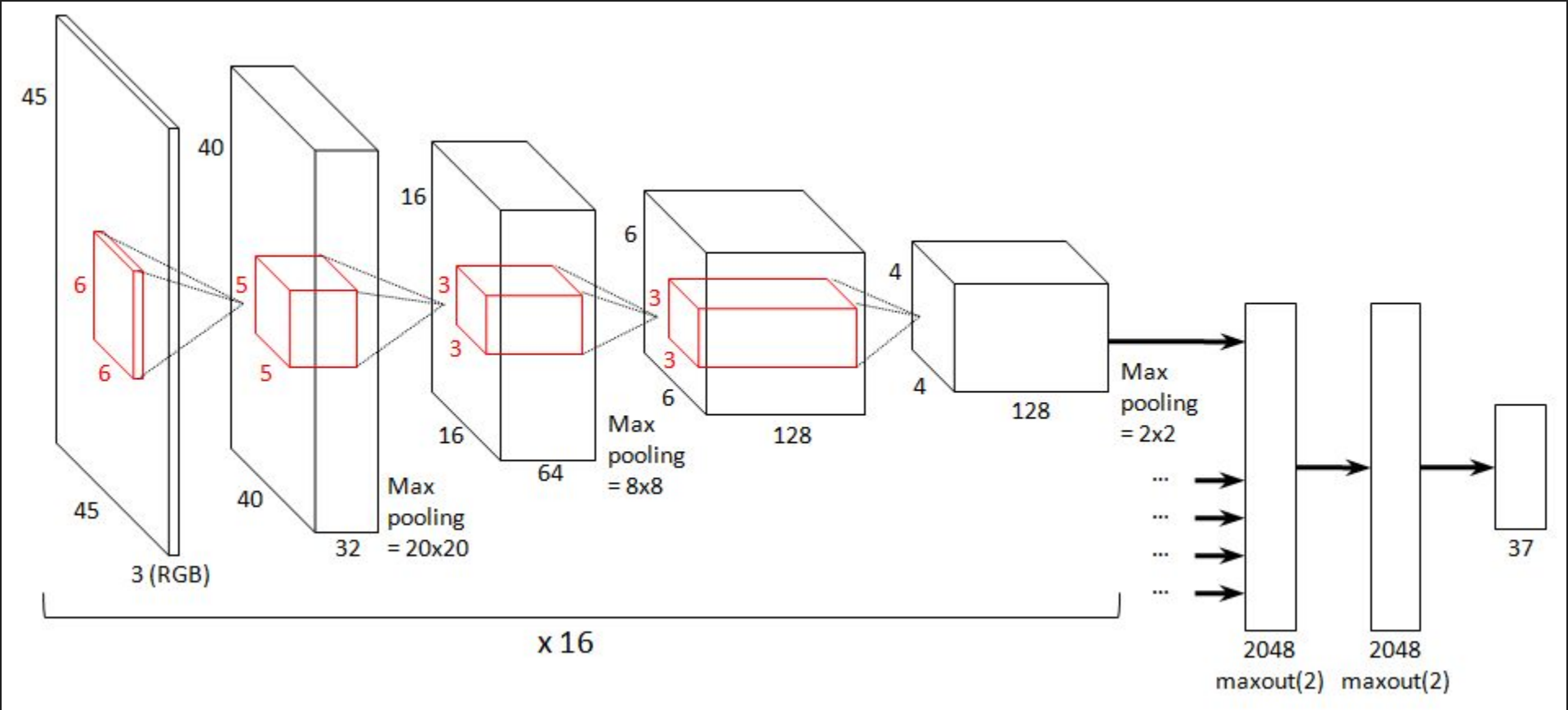


# Max-Pooling Variations

- Shown is most common approach: 2x2 pool filters and stride 2
- Can do overlap pooling: 3x3 pool filters and stride 2
- Don’t need to do max. Can do average, L2-norm, etc.
  - Usually less effective



# End up with architectures like this



# What’s going through the network?

- First hidden layer relatively easy to follow
  - For example, depth slice 2 is looking for blue and red delineations
- The next hidden layers are combinations of those previous ones, and the features become impossible to track

Tensorflow tutorial code

# Reading regarding ConvNets

- What a Deep Neural Network thinks about your #selfie
- Andrej Karpathy is one of the leading researchers on deep learning
  - Teaches the very famous Stanford deep learning class
  - His lectures are available online

# Next week’s reading

- The Unreasonable Effectiveness of Recurrent Neural Networks
  - The coolest article I have ever read about neural nets
  - Also written by Andrej Karpathy