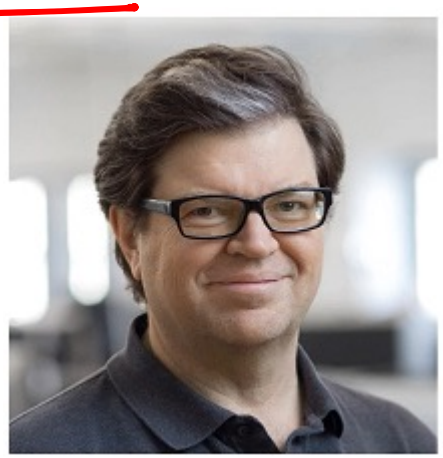
DEEP LEARNING – Convolutional Neural Network

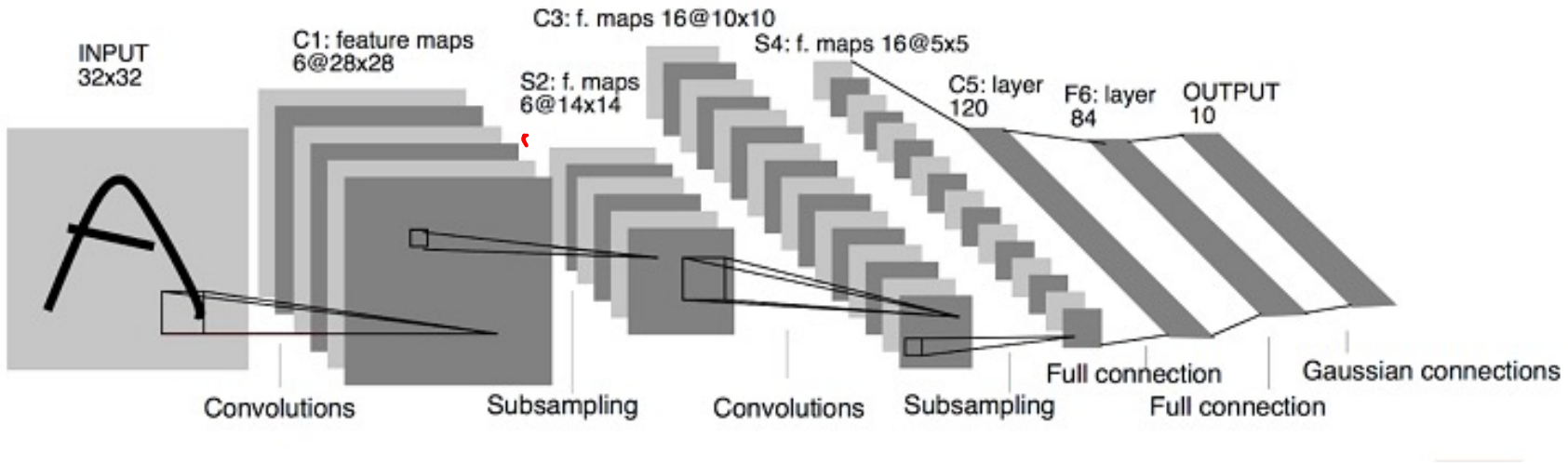
Trainer: Dr. Darshan Ingle.

CNN Architecture

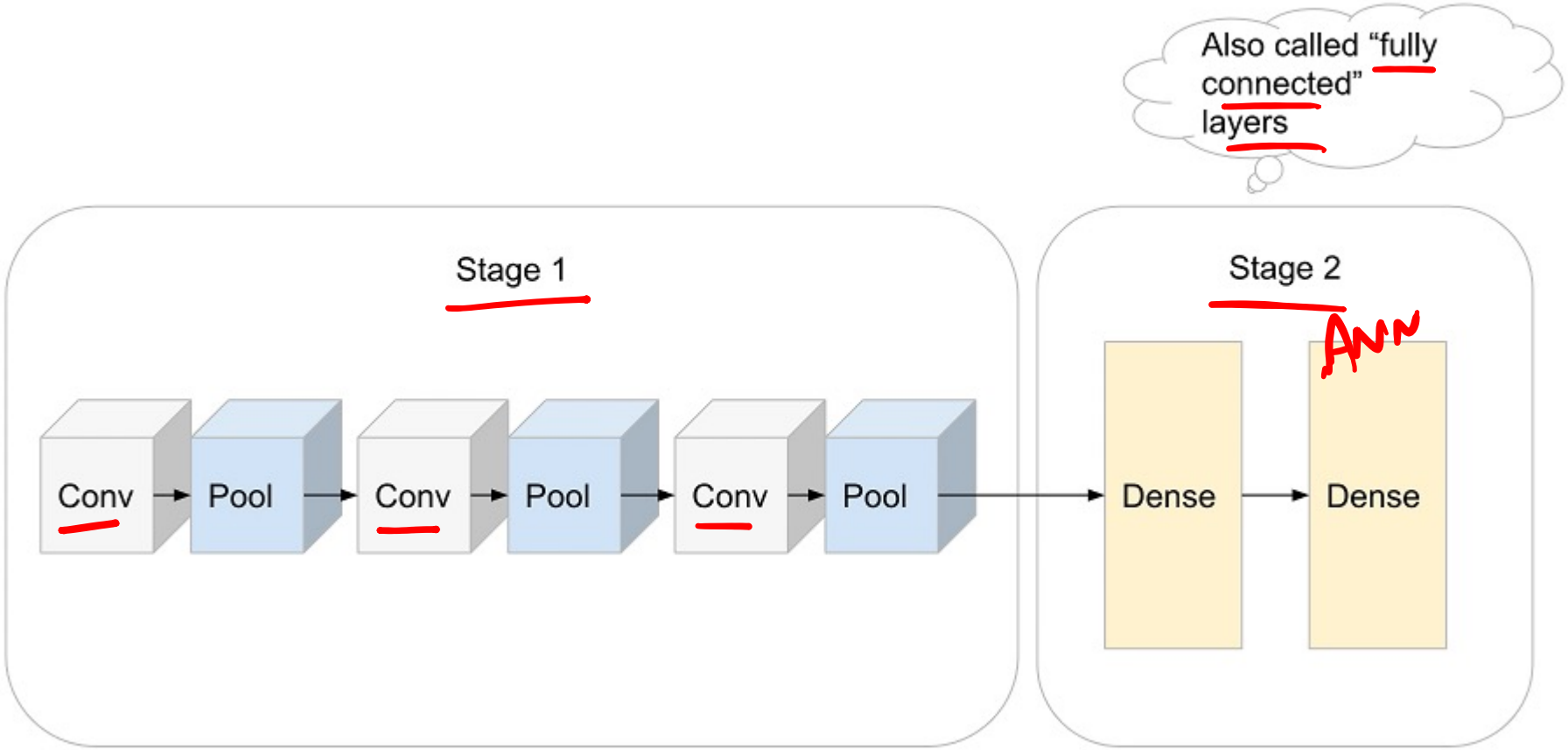
CNN Architecture

- Now that you understand how exactly a convolution layer works including the bias term and activation function we can now consider the architecture of a convolution or neural network and why it's that way.
- So as a little bit of a history lesson, modern CNN is essentially all originated from the same model, the LeNet.
- This is named after Yann LeCun, one of the original Deep Learning pioneers, along with Geoff Hinton and Yoshua Bengio.

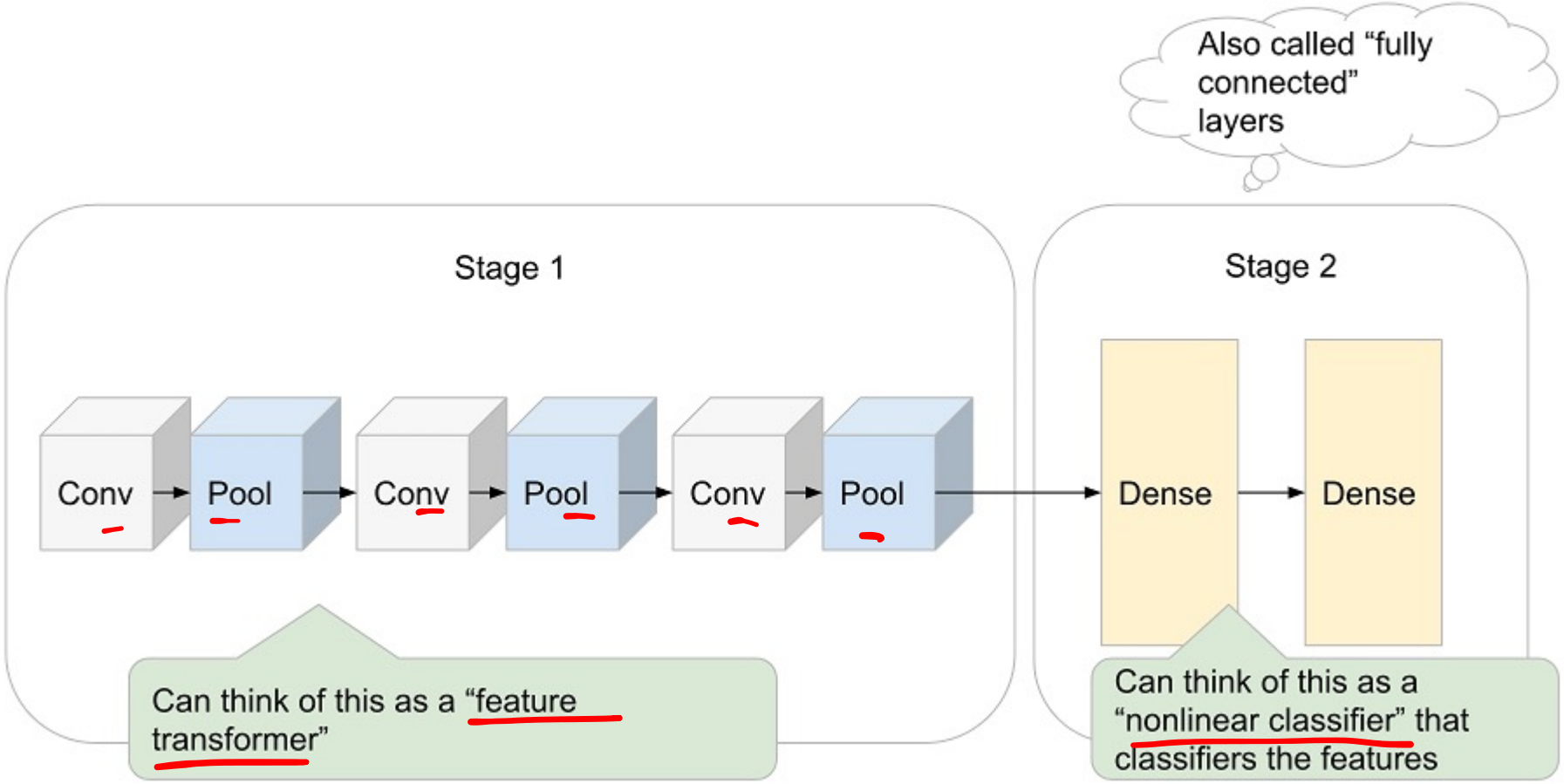




Typical CNN


Also called "fully connected" layers
Stage 1
Stage 2
Conv
Pool
Conv
Pool
Conv
Pool
Dense
Dense

Typical CNN


Also called “fully connected” layers
Stage 1
Stage 2
Conv
Pool
Conv
Pool
Conv
Pool
Dense
Dense
Can think of this as a “feature transformer”
Can think of this as a “nonlinear classifier” that classifiers the features

Pooling

Downsampling

Downsample by 2.

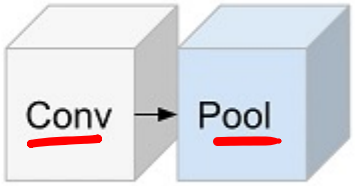
Bigger

PS=2

smaller

50x50

100x100



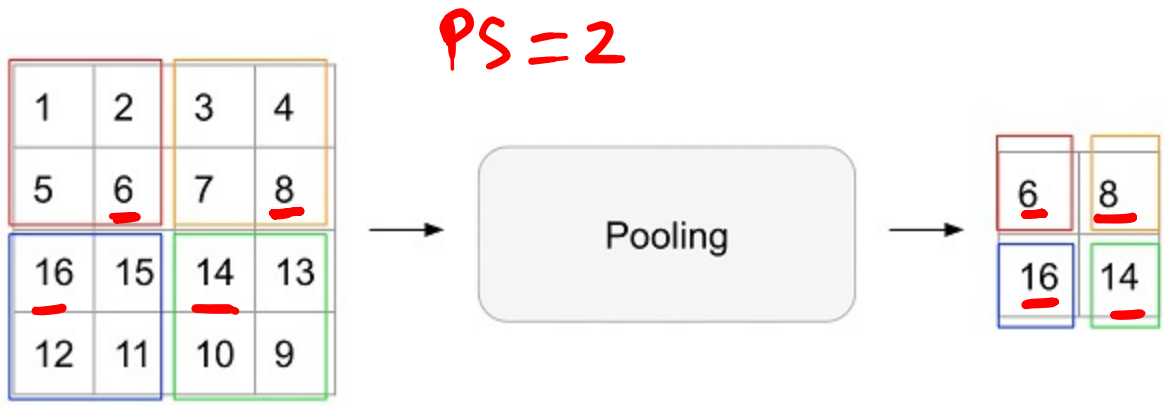
Types of Pooling

- There are two types of Pooling:

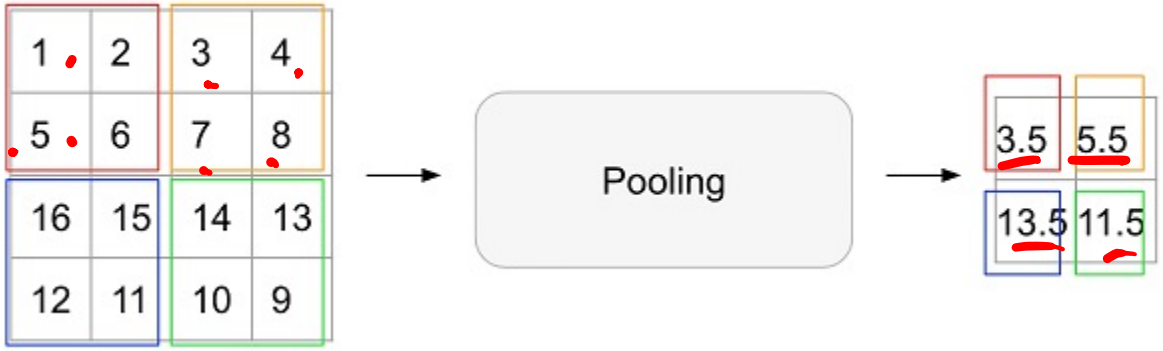
1. Max pooling
2. Average pooling

- Which one to use is a Hyperparameter choice.

Max Pooling



Average Pooling



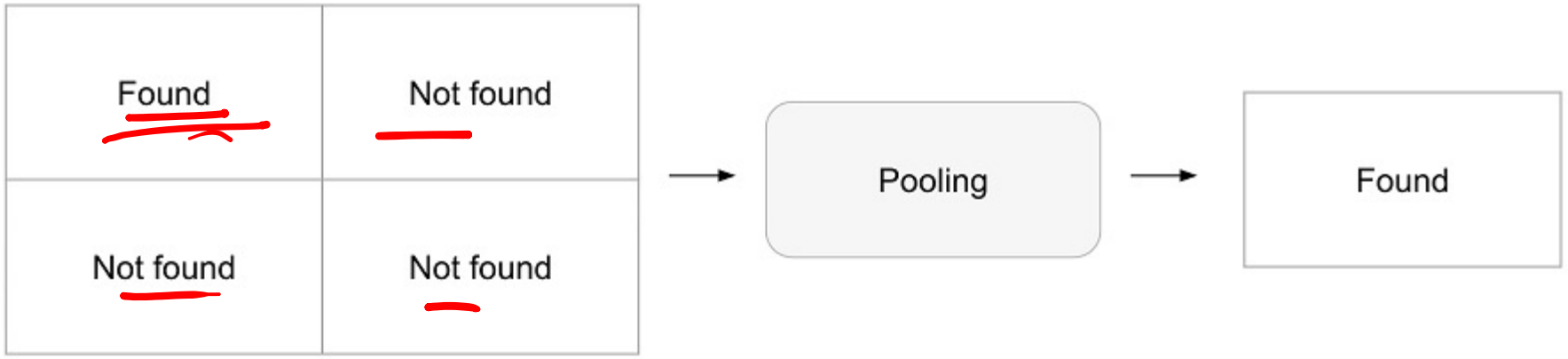
Why to use Pooling?

- Practical: If we shrink the image, we have less data to process.

- Translational Invariance: I don’t care where in the image the feature occurred, I just care that it did.

What is the Max Pooling doing?

- **Recall: Convolution is a "pattern finder" (the highest number is the best matching location)**

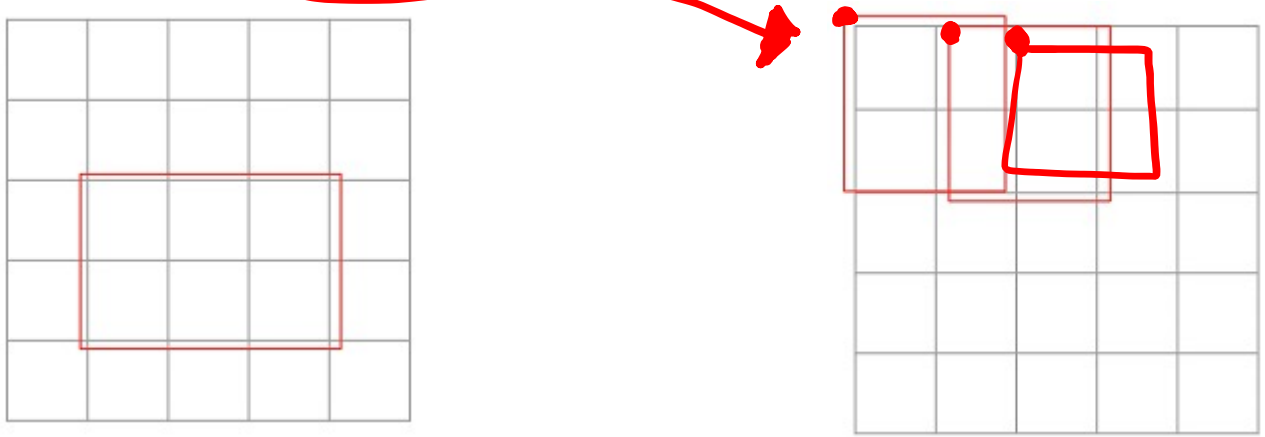


What about Average Pooling?

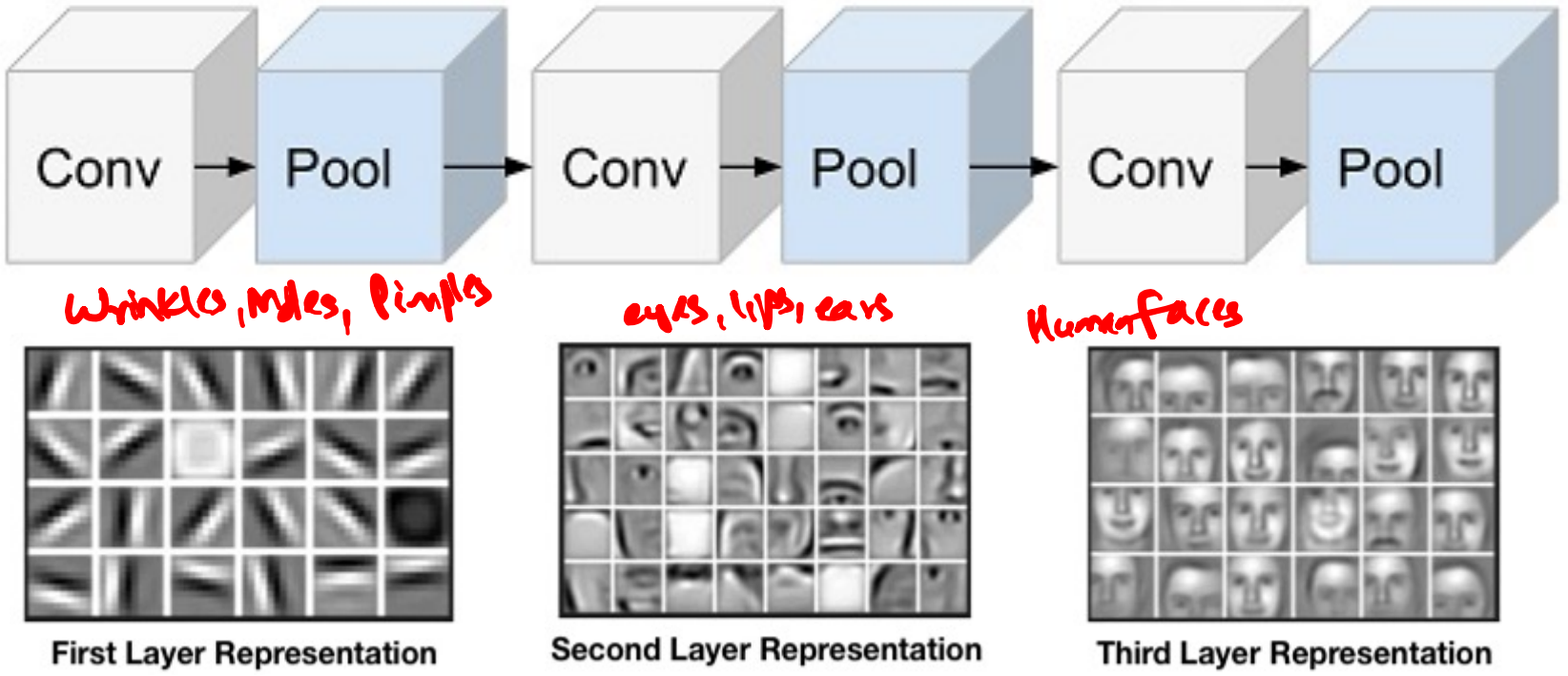
- Average pooling is the same idea but Max pooling is a little more intuitive in this regard.

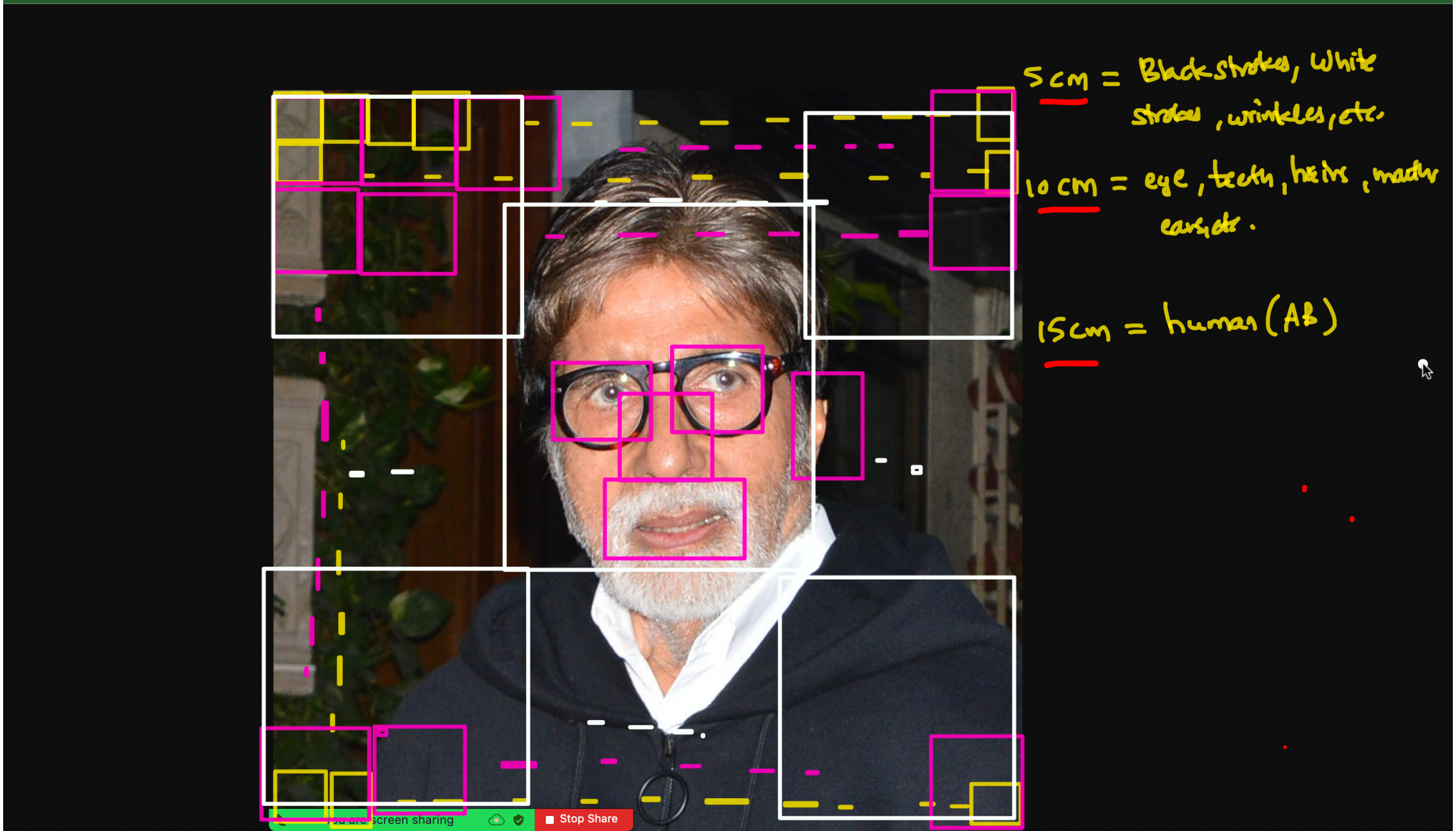
Different Pool Sizes

- It's possible to have a non-square window, e.g. 2x3 or 3x2, but this is unconventional
- It's also possible for boxes to overlap (this is called "stride")
  - Previously, we looked at a pool size of 2 with a stride of 2 (common)
  - If you had a stride of 1, the boxes would overlap (not common)



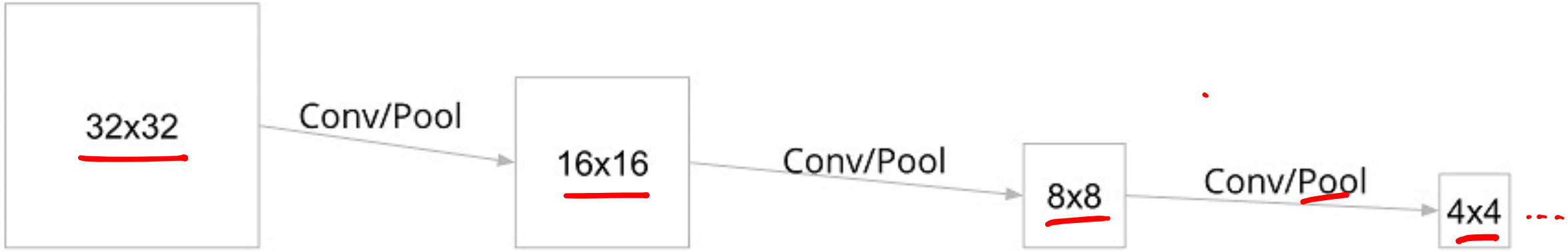
Why Convolution followed by pooling?




5 cm = Black strokes, White strokes, wrinkles, etc.
10 cm = eye, teeth, hairs, mostly
15cm = human (AB)
Stop Share

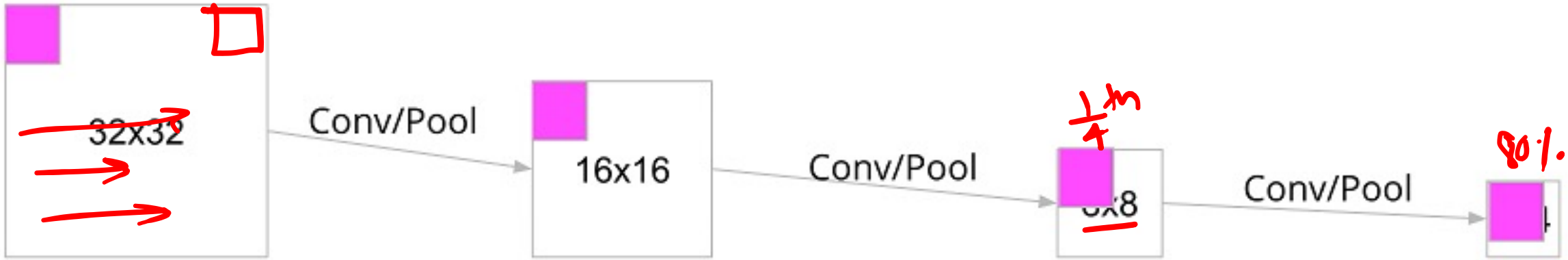
Why Convolution followed by pooling?

- Key point: after each "conv-pool", the image shrinks, but filter sizes generally stay the same
- Common filter sizes are 3x3, 5x5, 7x7
- Assume "same mode" convolution and pool size = 2



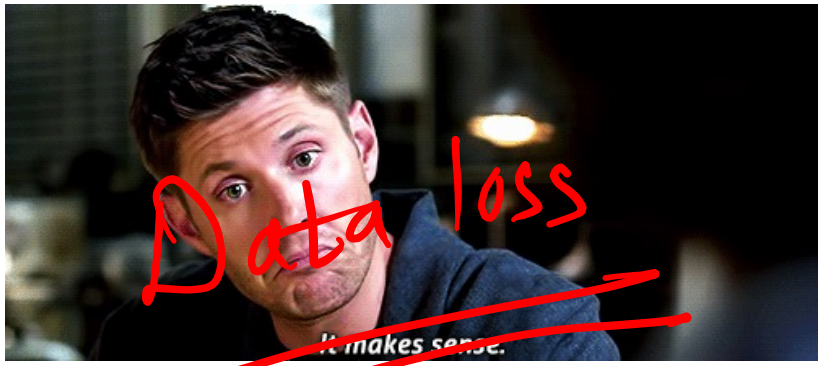
Why Convolution followed by pooling?

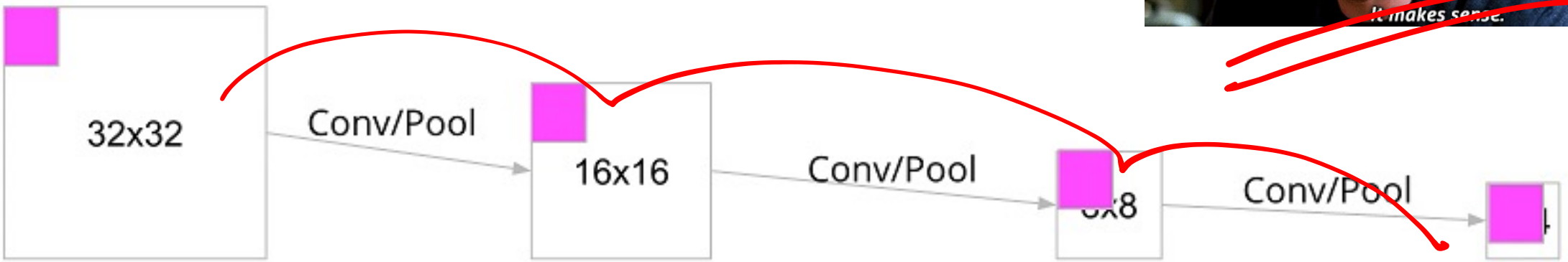
- If the filter size stays the same, but the image shrinks, then the portion of the image that the filter covers increases!



Why Convolution followed by pooling?

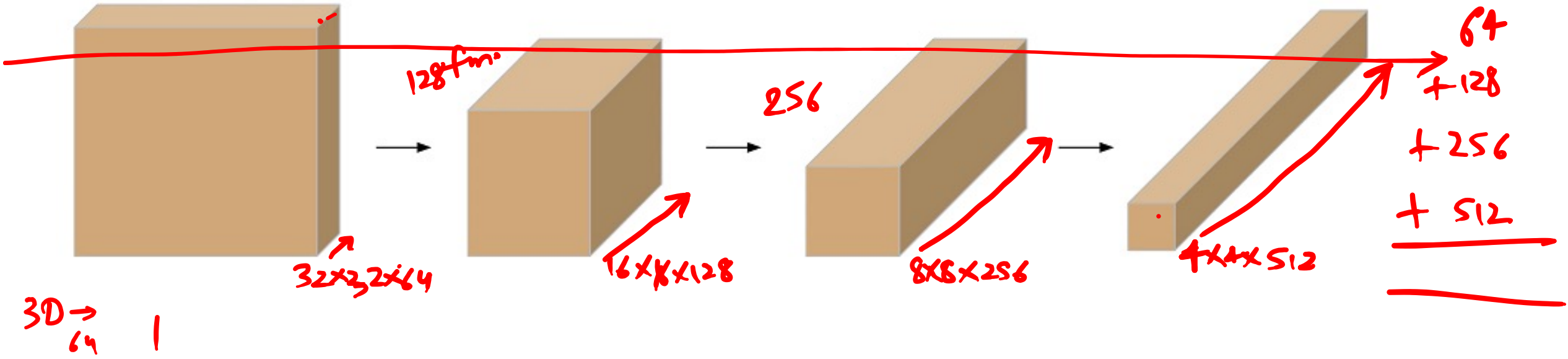
- The input image shrinks
- Since filters stay the same size, they find increasingly large patterns (relative to the image)
- This is why CNNs learn *hierarchical* features





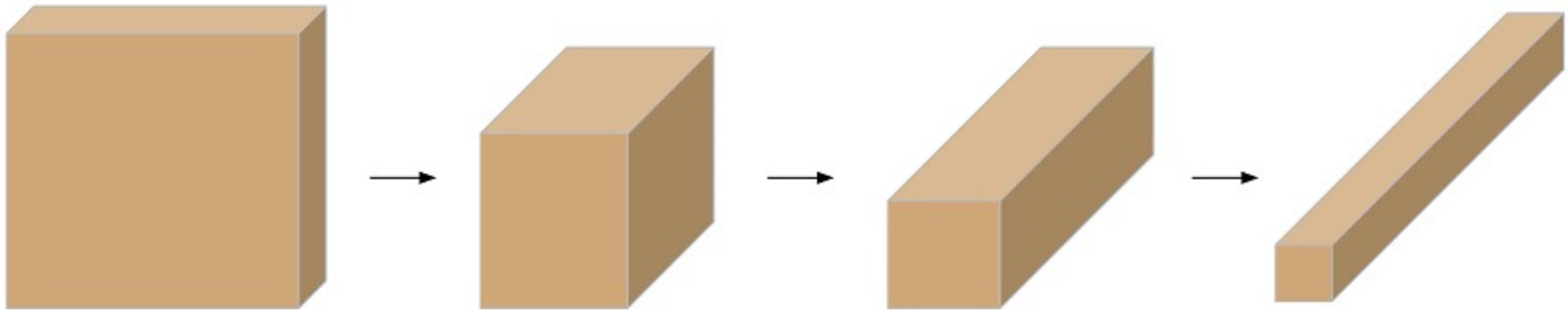
Losing Information

- Do we lose information if we shrink the image? Yes!
- We lose spatial information: we don't care *where* the feature was found
- We haven't yet considered the # of feature maps
- Generally, these increase at each layer
- So we *gain* information in terms of what features were found

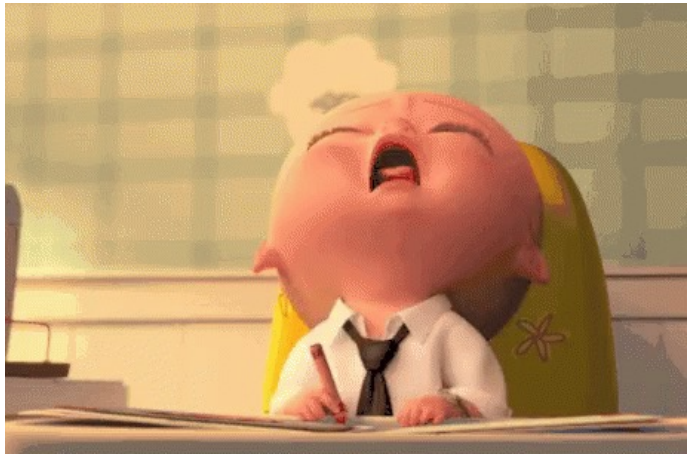


What Hyperparameters to choose?

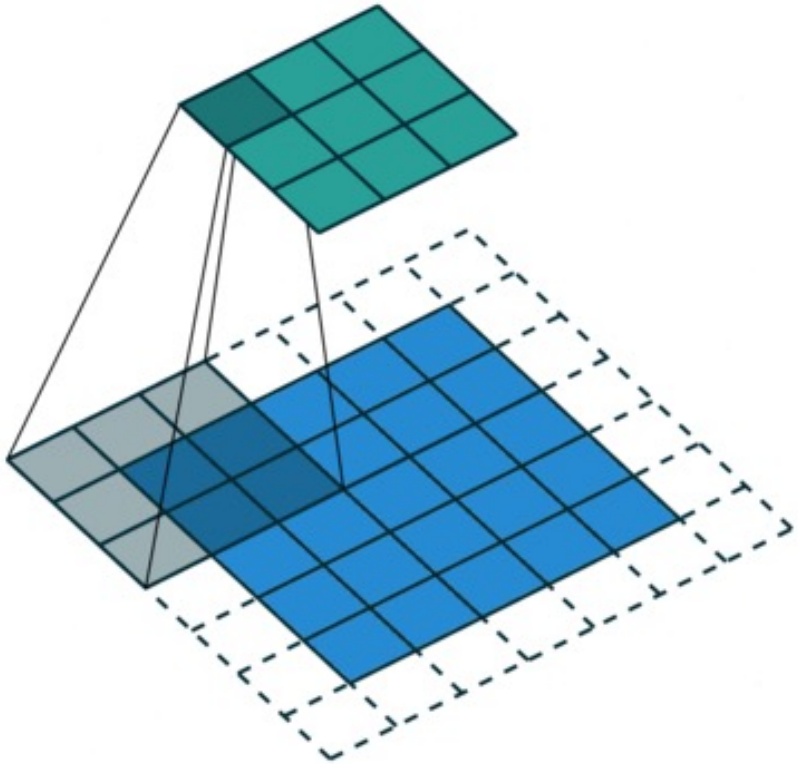
- There are so many choices!
- Previously: learning rate, # hidden layers, # hidden units per layer
- With CNNs, the conventions are pretty standard
  - Small filters relative to image, e.g. 3x3, 5x5, 7x7
  - Repeat: convolution → pooling → convolution → pooling → etc.
  - Increase # feature maps, e.g.: 32 → 64 → 128 → 128
  - Read lots of papers!



Alternative to Pooling: Stride



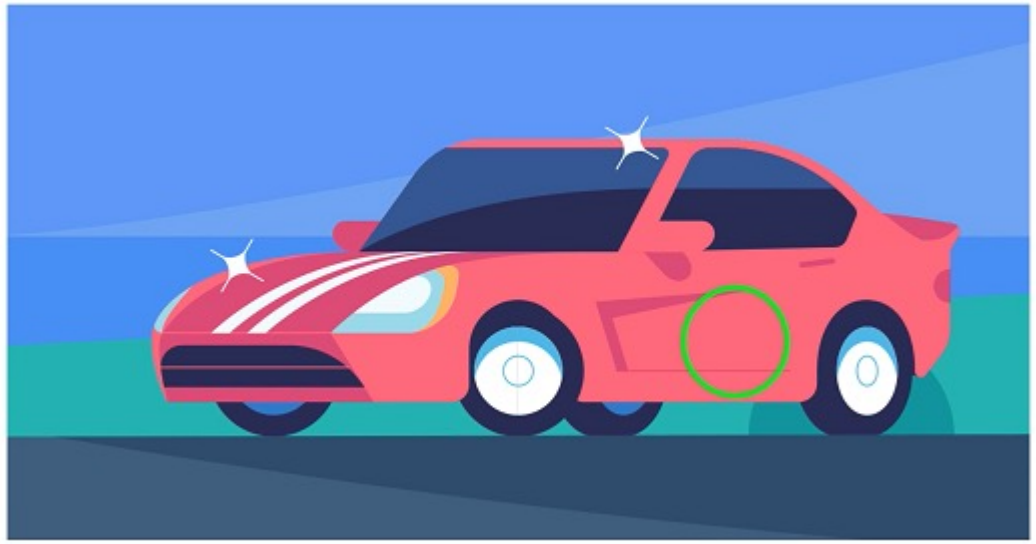
Convolution: It has a stride option.



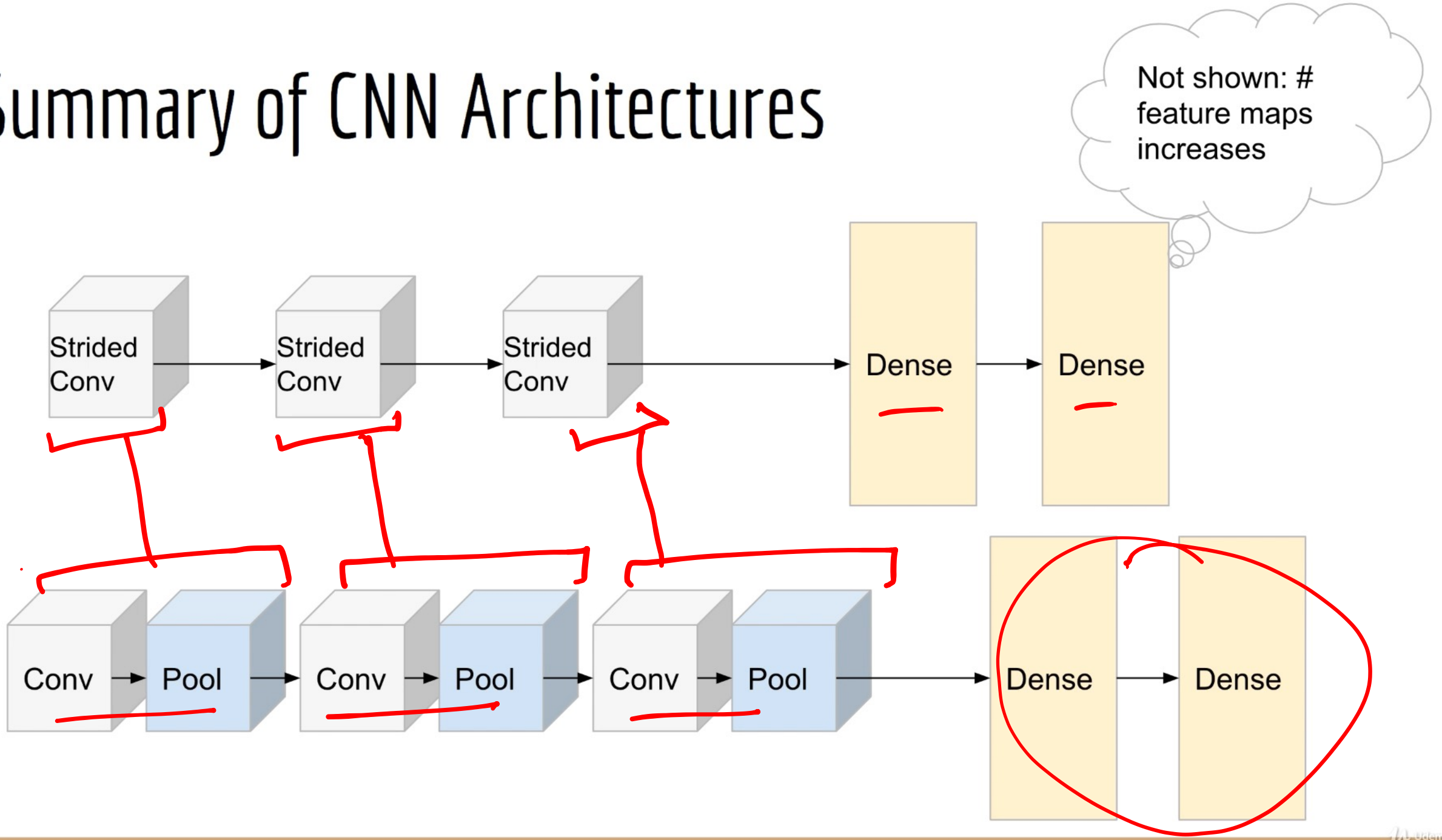
Why does it work?

- An image is just large patches of “stuff”

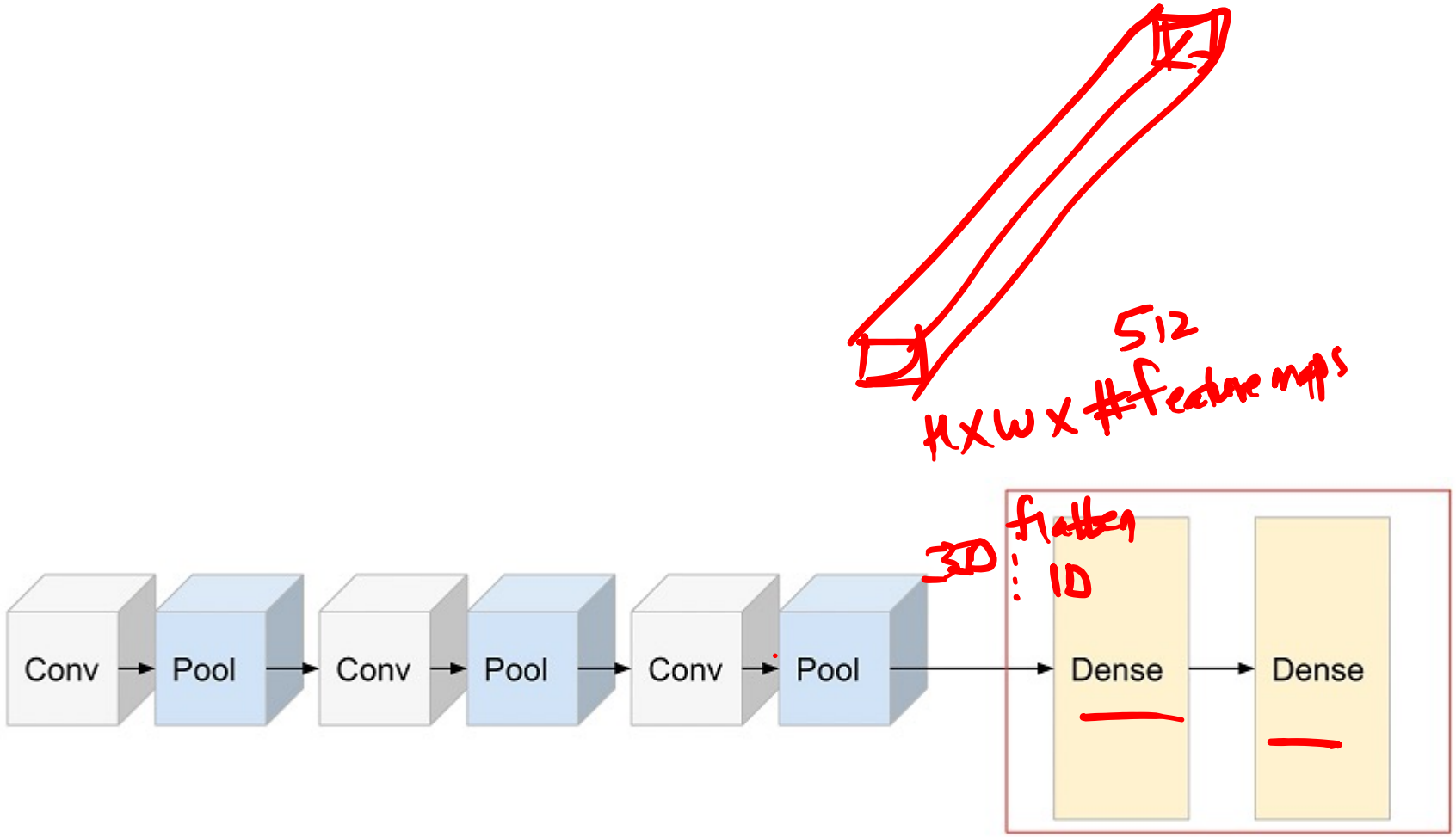
- A red pixel’s neighbors (up / down / left / right) are *probably* also red



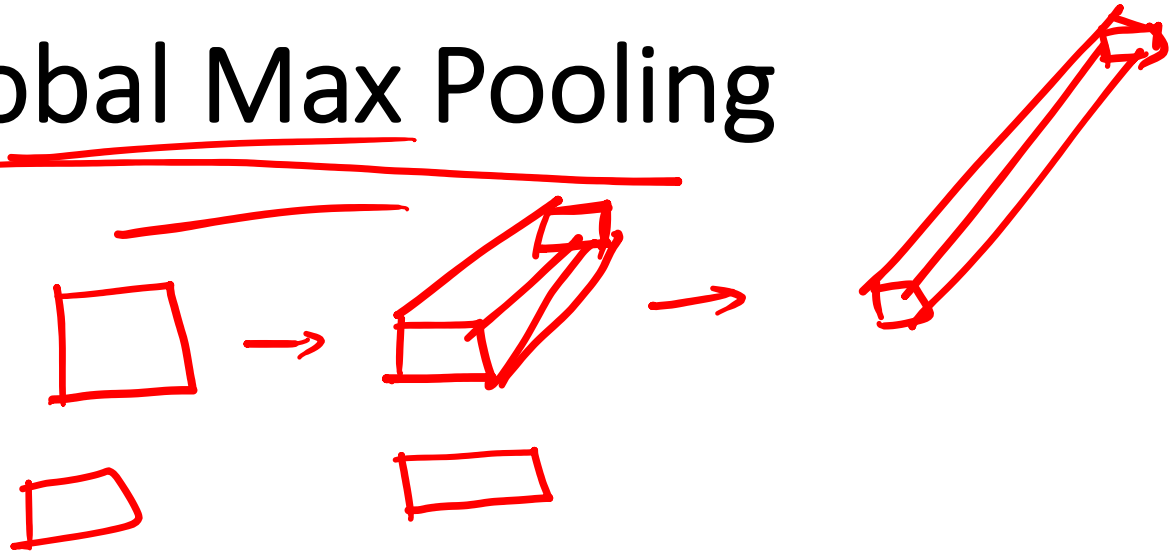
Summary of CNN Architectures

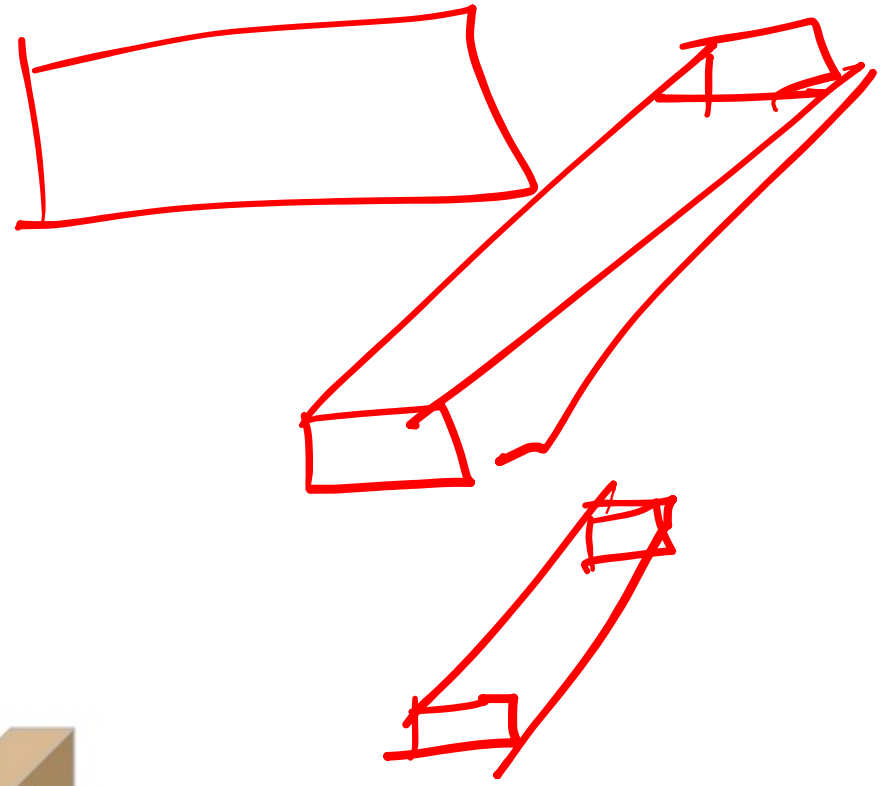


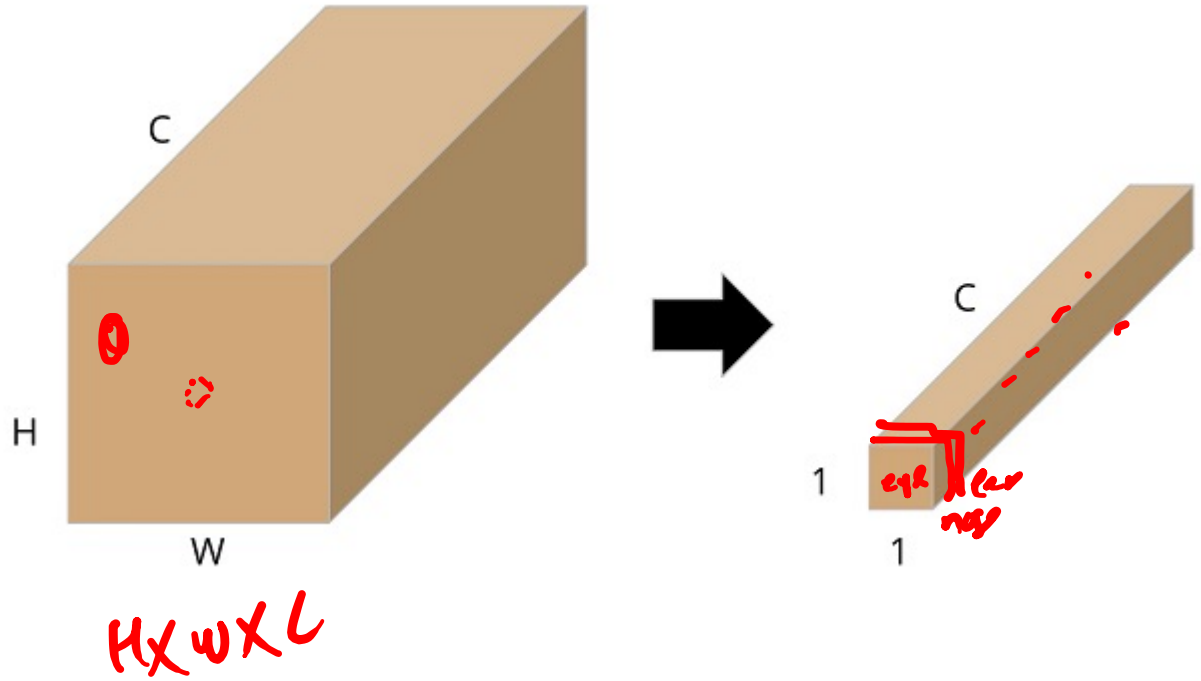
Dense Neural Network



Global Max Pooling







HXWXL

What's wrong with Flatten?

(a) Input image size = 32×32

we do 4 convolutions with stride=2

32×32 > 16×16 > 8×8 > 4×4 > 2×2

(b) Input image size=64×64

64×64 > 32×32 > 16×16 > 8×8 > 4×4

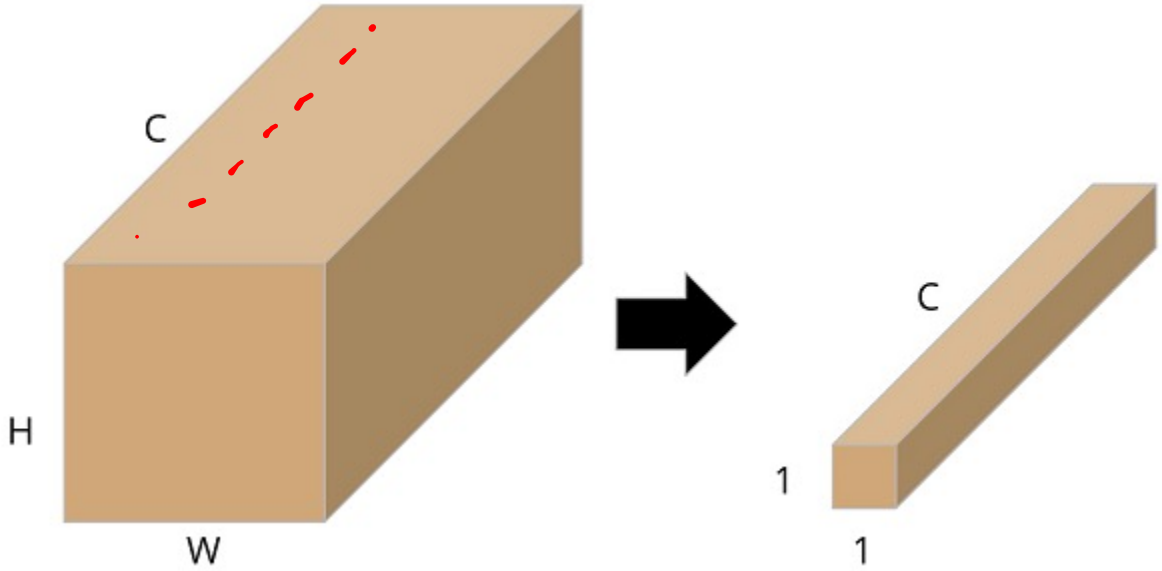
If final #feature maps =100

(a) 2×2×100 = 400D vector for 32×32 image

(b) 4×4×100 = 1600D vector for 64×64 image.

Global Max Pooling

- We *always* get an output of 1x1xC (or just a vector of size C) regardless of H and W
- Takes the max over each feature map
- We don't care *where* the feature was found
- Also: global average pooling



Exception to CNN

- If you pass in images that are too small, you will get an error
- E.g. if you start with a 2x2 image, you cannot halve it 4 times
- You'll encounter this if you try to add too many conv layers to your CNN

..

Summary

3 tasks that determine final activation / # of nodes:

a. Regression : None
b. Binary Classfn: Sigmoid
c. Multiclass classfn: Softmax

① Conv > Pool > conv > Pool > . . . .
or
Strided conv > Strided conv. > . . .

② Flatten() — square
or
GlobalMaxPooling2D() — images of different sizes

③ Dense > Dense > ....