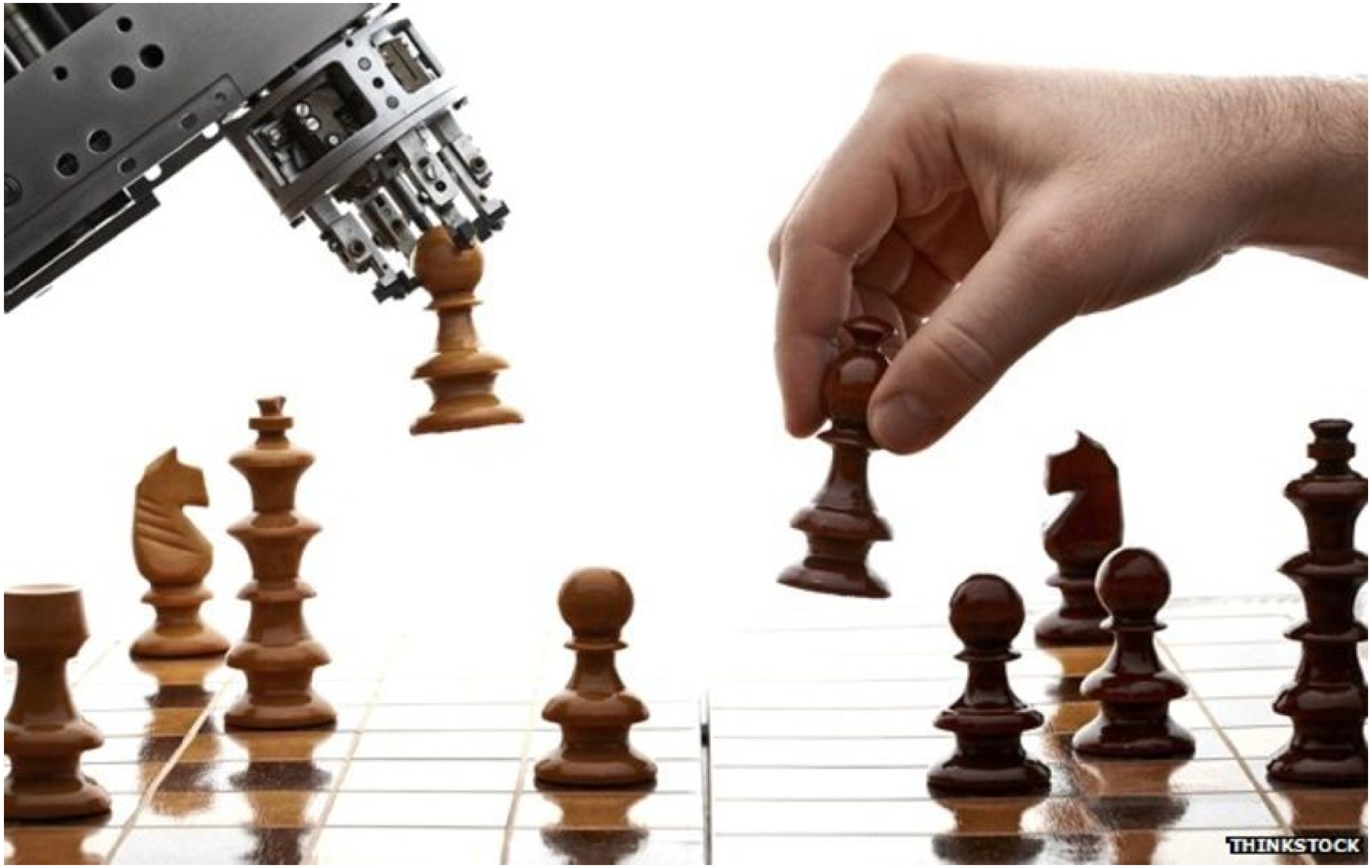


Introduction to Artificial Intelligence

COSC 4550 / COSC 5550

Professor Cheney

10/27/17

deep learning!

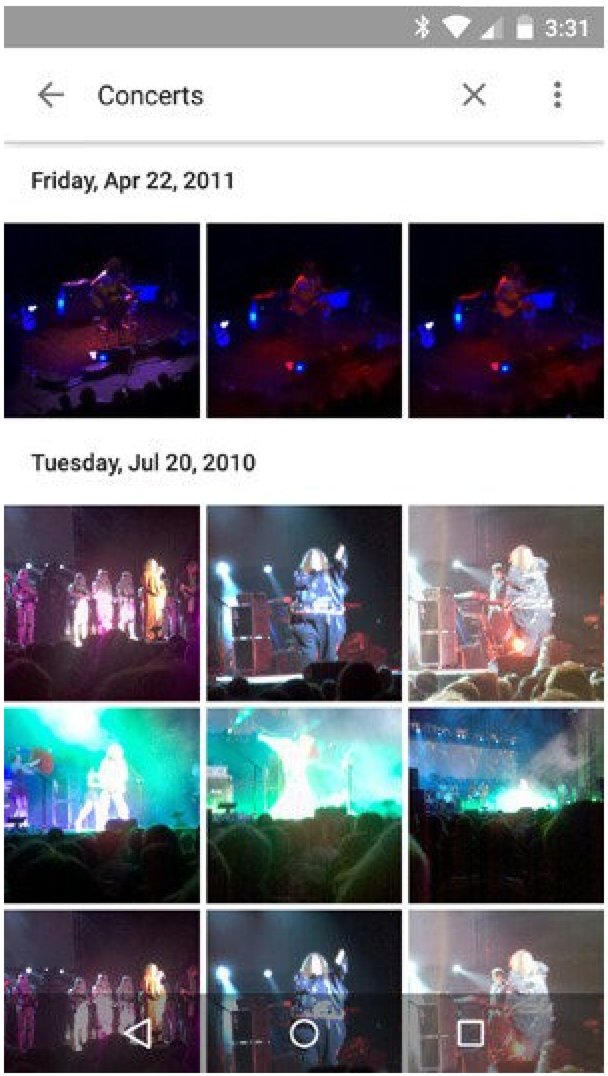
much of our interactions with the world are through vision

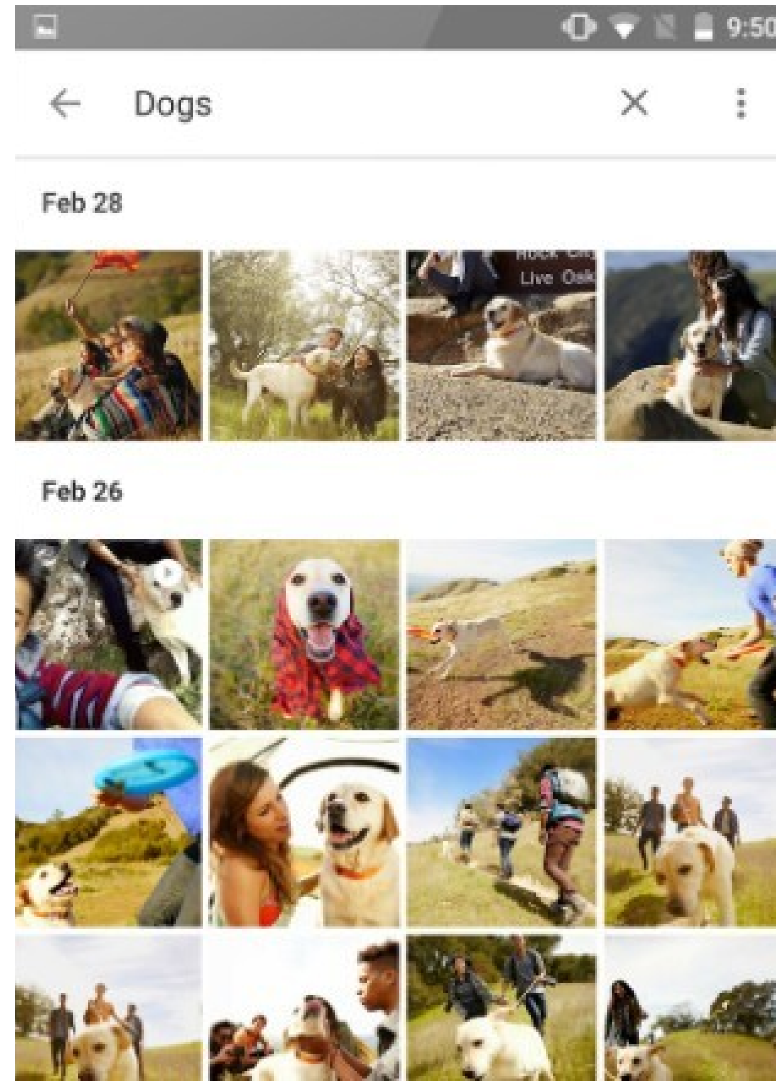
thus, it is very important for general artificial intelligence to be able to make high-level abstract decision about visual information (e.g. from camera sensors)

while we focus here on deep learning for spacial abstraction we will see later how we use similar ideas to make abstractions along other dimensions (e.g. through time) for these reasons, deep learning for image processing has been extremely practical and valuable to industry lately

and deep learning researchers/practitioners have been in high demand

Google Photos

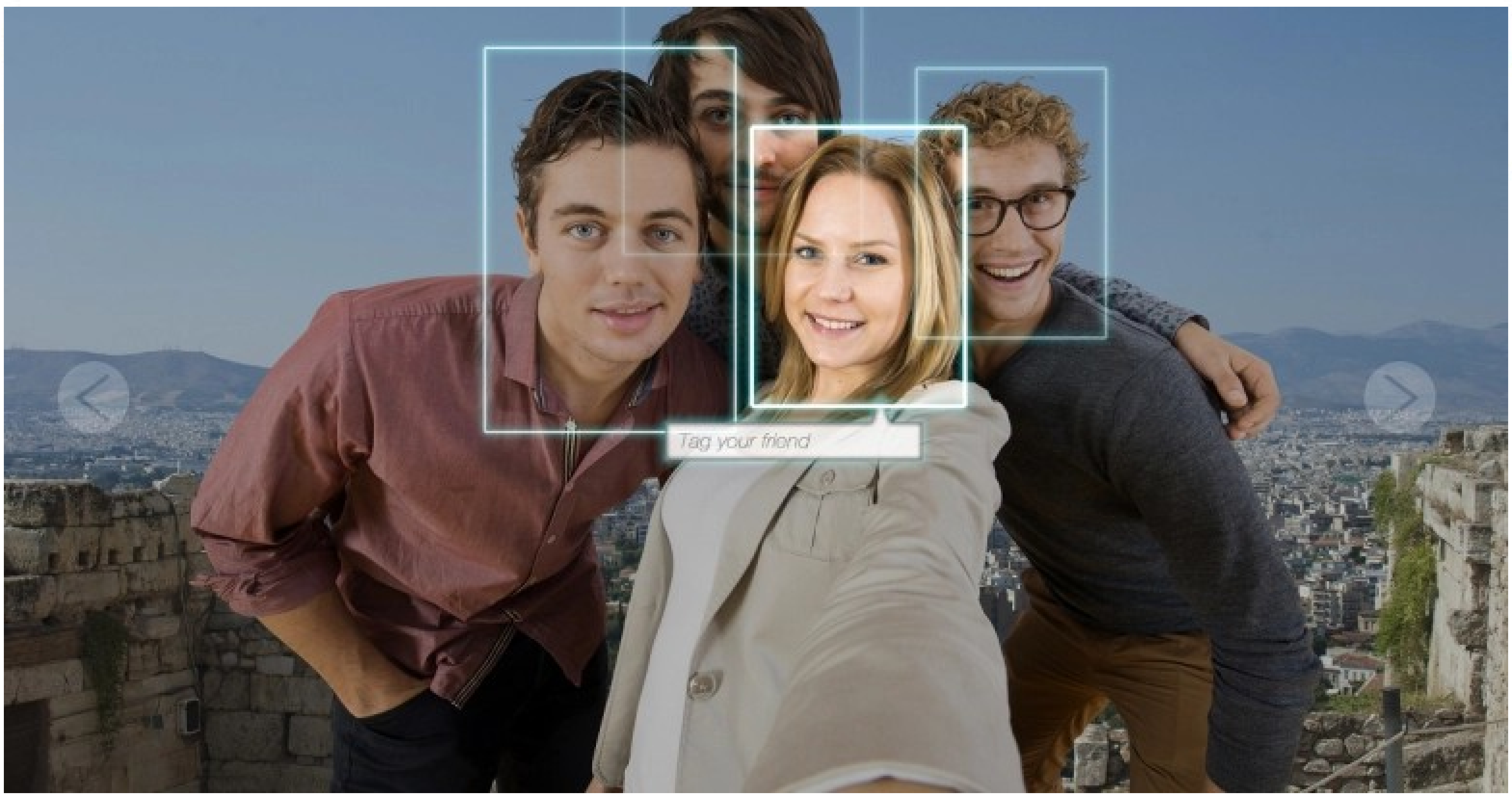




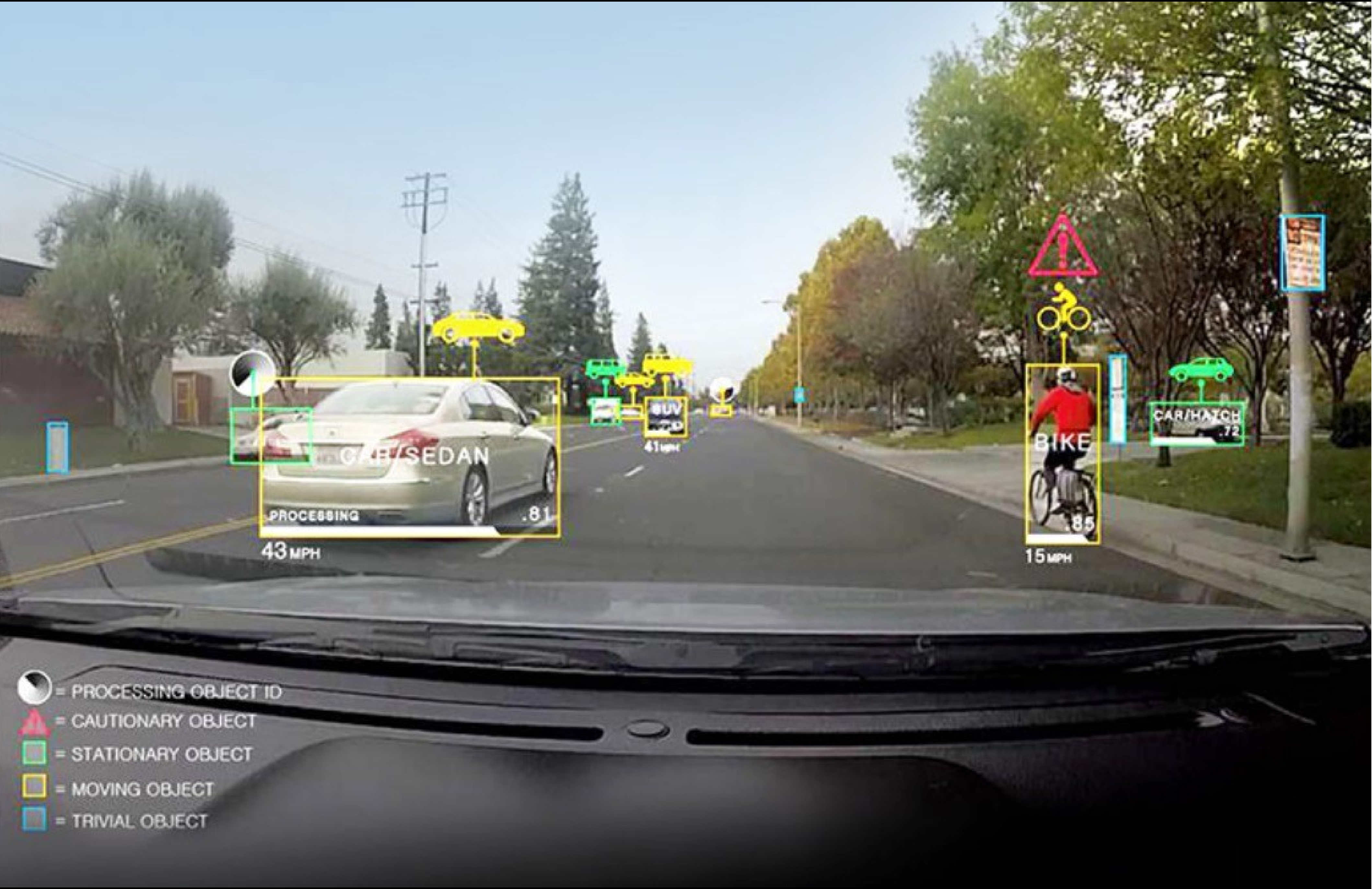
facebook

Photos: Suggest Tags

This helps your friends label and share their photos, and makes it easier to find out when photos of you are posted.



AD Mobilize
Crowd Analytics
SUBWAY SHUTTLE
42ND STREET
WEST BALCONY
TICKET MACHINES
TRACKS 31 TO 42
DINING CONCOURSE
Information
admobilize.com


CAR/SEDAN
PROCESSING
.81
43 MPH
SUV
41 MPH
BIKE
.85
15 MPH
CAR/HATCH
.72
= PROCESSING OBJECT ID
= CAUTIONARY OBJECT
= STATIONARY OBJECT
= MOVING OBJECT
= TRIVIAL OBJECT

Not Hotdog

SeeFood Technologies Inc. **Food & Drink** ★★★★★ 410

E **Everyone**

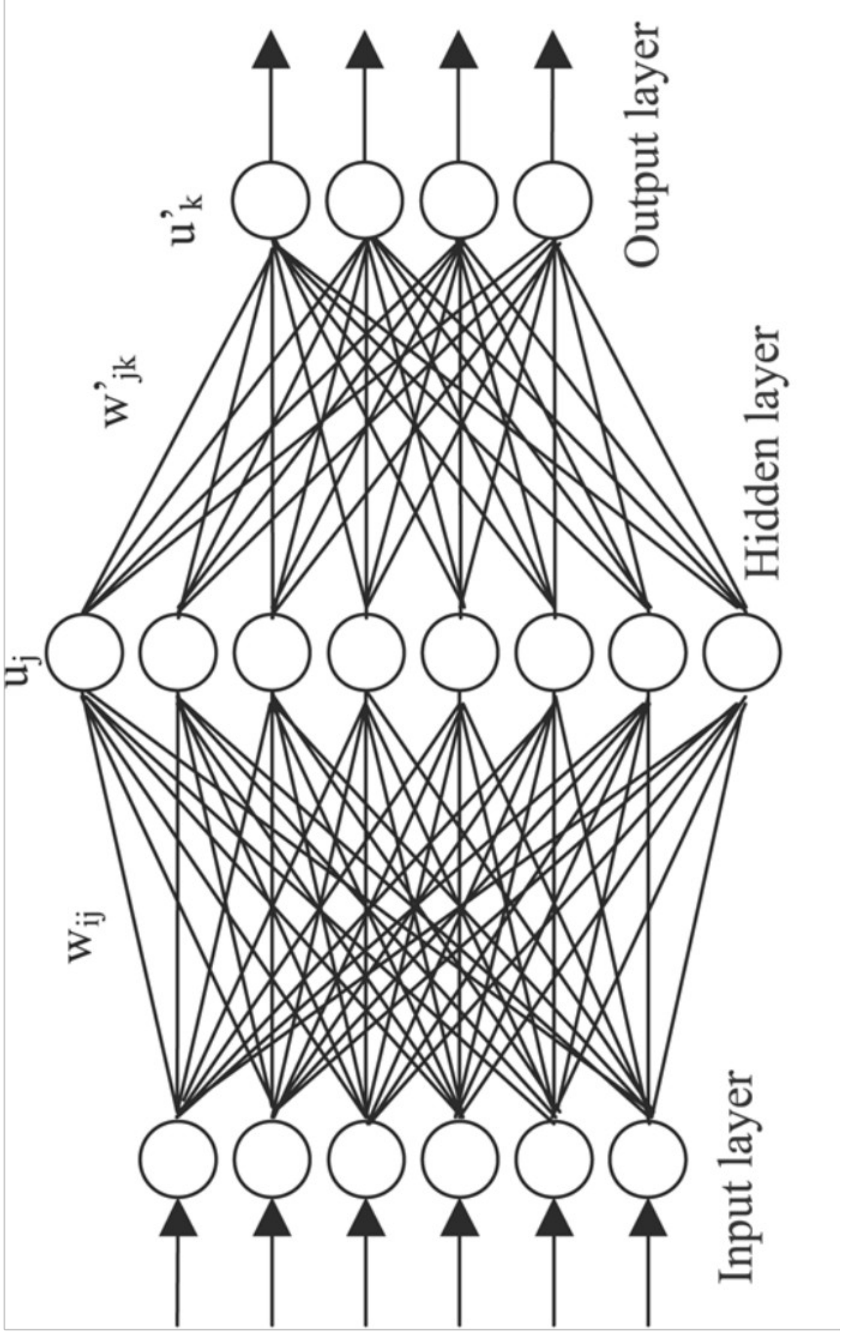
This app is compatible with all of your devices.

Add to Wishlist Install

What would you say if I told you there is a app on the market that tell you if you have a hotdog or not a hotdog. It is very good and I do not want to work on it any more. You can hire someone else.

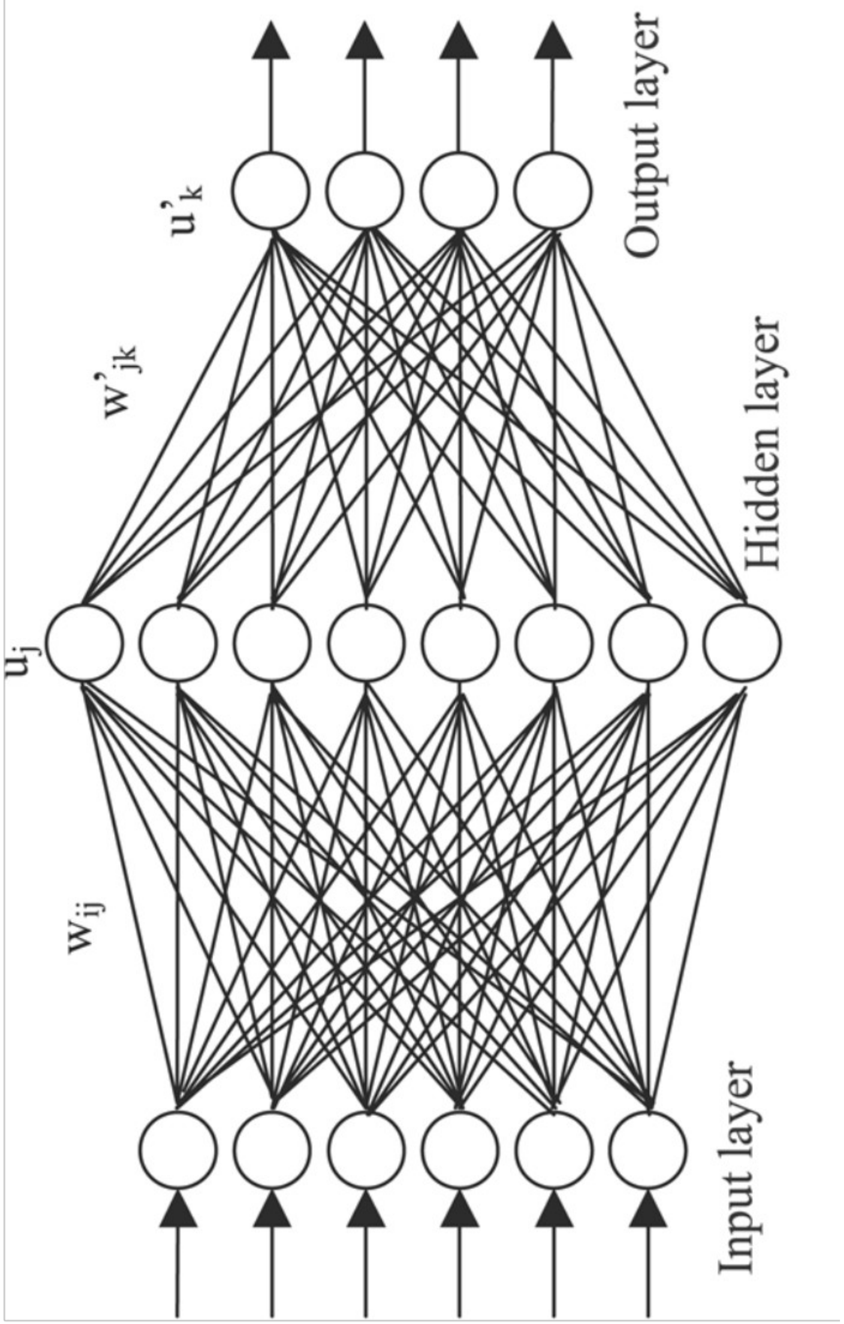
REVIEWS

convolutional neural networks



each layer in a fully connected neural network lets us find more abstract features from lower level attributes in our data

this was great when we were choosing the original attributes by hand, but now that we're using in very high dimensional inputs, it gets quite expensive!

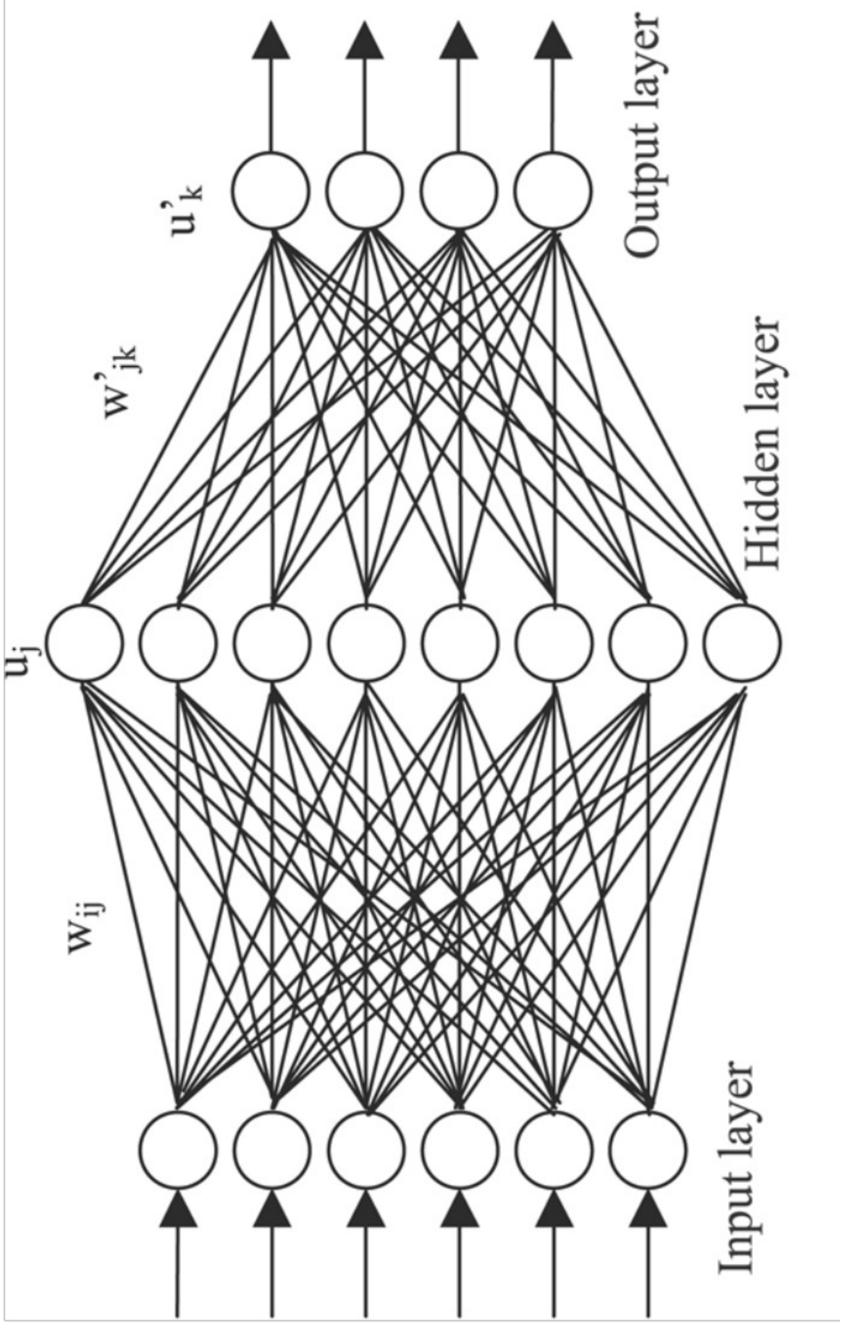


e.g. for a (fairly small) camera image of 256 x 256 pixels, for which we want to find 100 relevant feature at the next layer

256 * 256 = 65,536 input nodes

65,536 * 100 = 6,553,600 weights/parameters to learn!

and that's just for one layer!



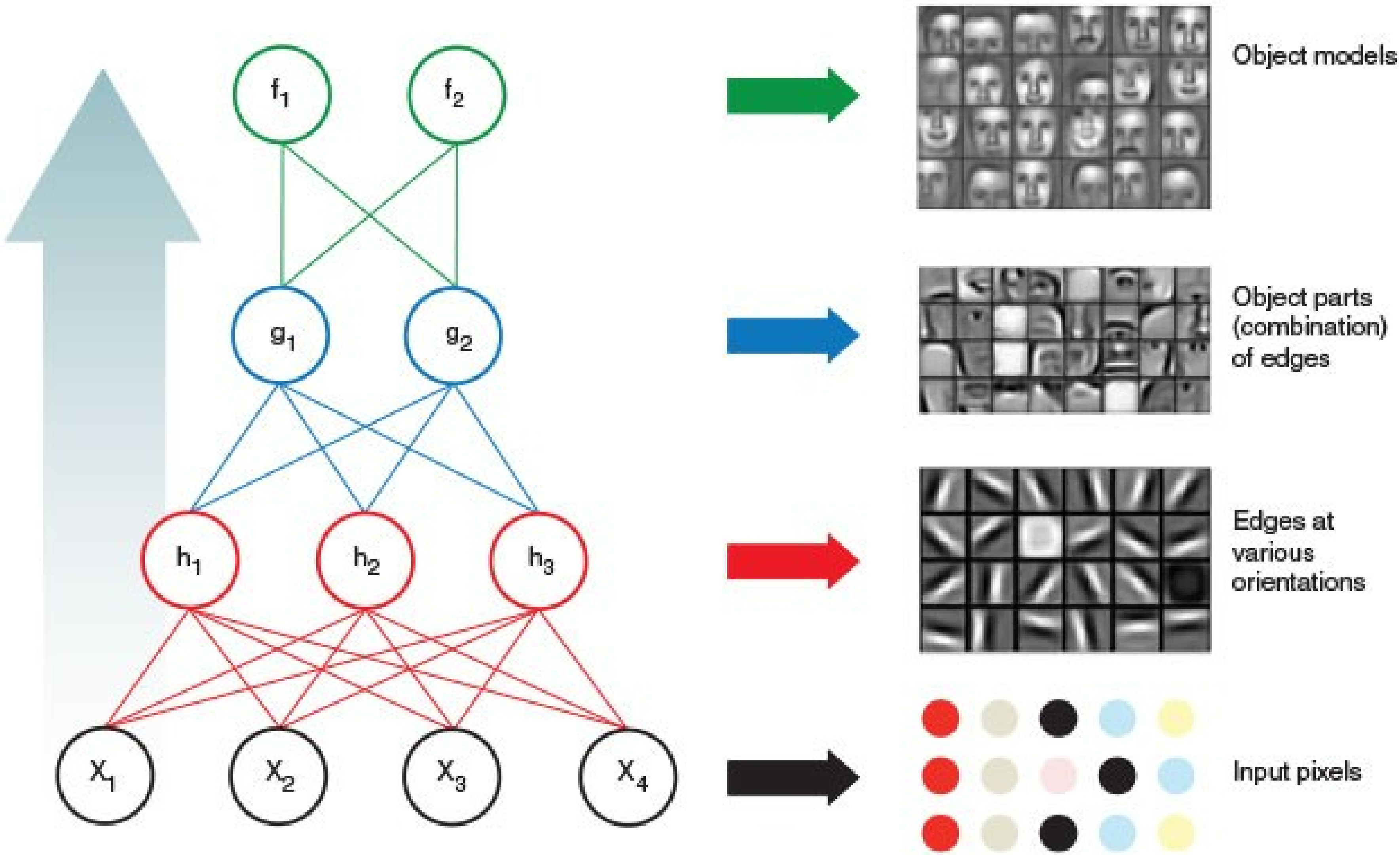
if we wanted just 5 layers, and 100 times more training data than we have parameters to fit

6,553,600 * 5 * 100 =
3,276,800,000 images
to collect for our dataset

luckily for us, it turns out that
fully connected networks are overkill

recall that we were interested in slowly building up
more abstract features over a number of layers

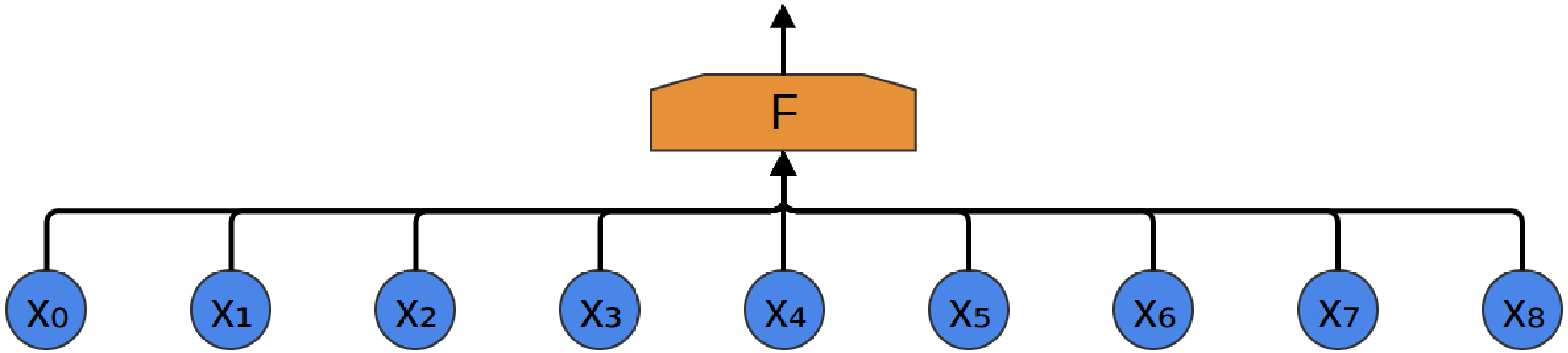
that means that to achieve spatial abstraction,
each layer really only needs to be connected to the
features of the previous layer nearby by it in the image



f1
f2
g1
g2
h1
h2
h3
X1
X2
X3
X4
Object models
Object parts (combination) of edges
Edges at various orientations
Input pixels

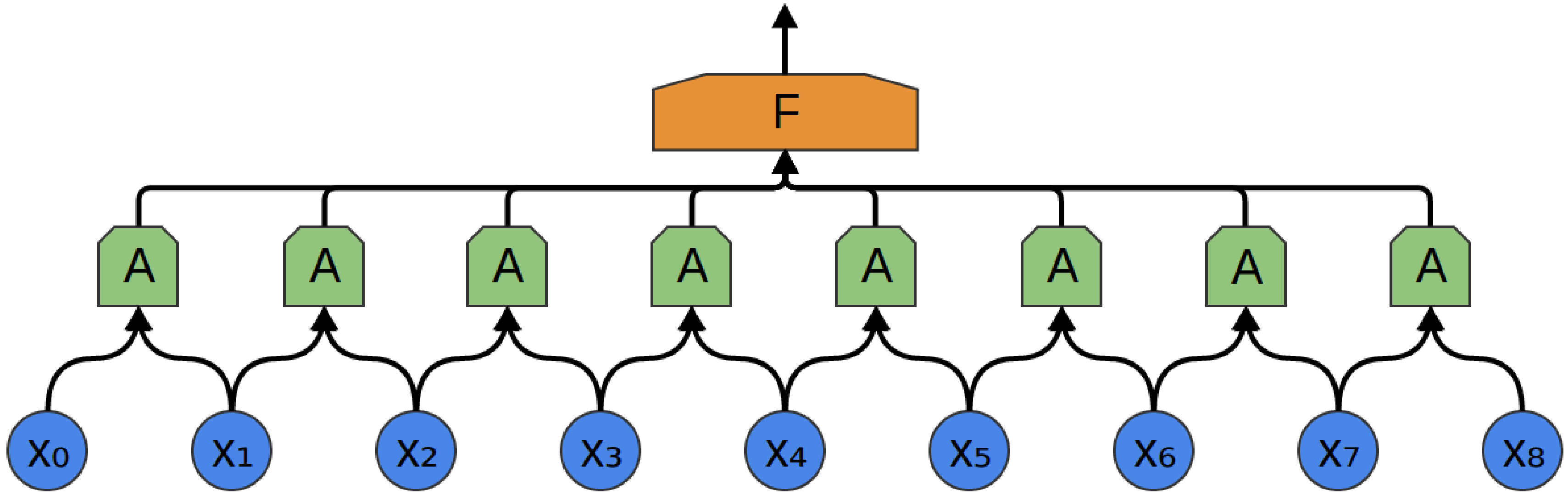
imagine a simple 1-D input vector

a fully connected neural network would use all the weights (x_i) for every attribute in the layer below it to predict what feature was present at that layer of abstraction (F)



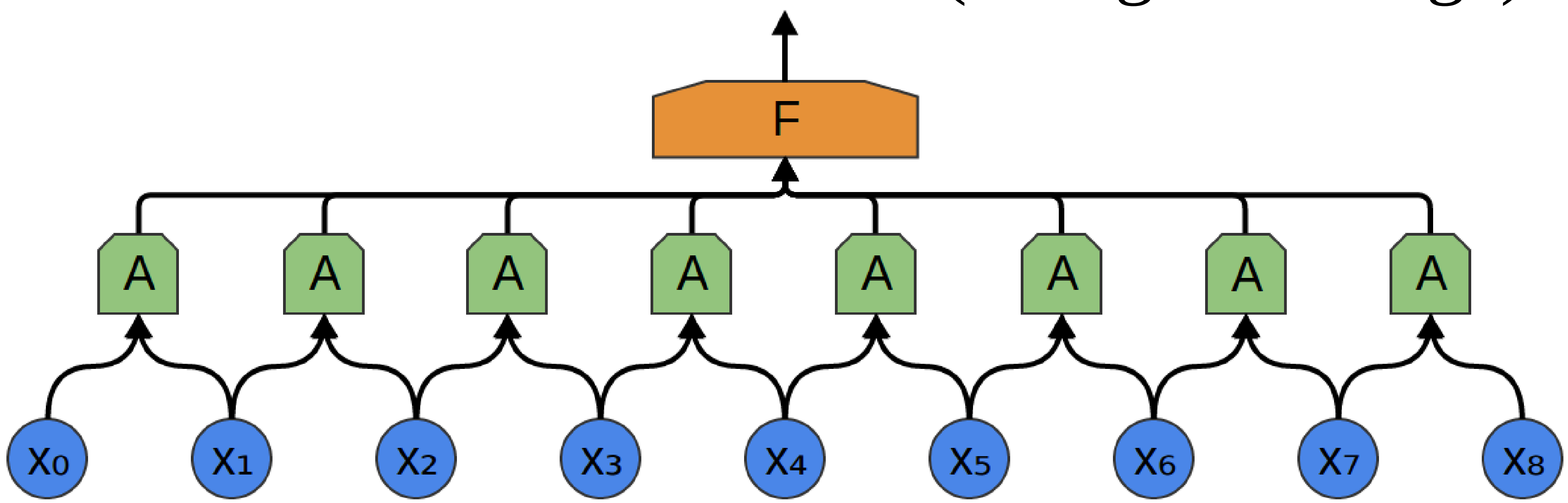
in a convolutional neural network, we'll ask what feature is present at each location, given the previous layer's features at that point in space

so the output of each “A” represents a 2-pixel wide feature



we've reduced the size of our input from 9 1-pixel features to 8 (more abstract) 2-pixel-wide features

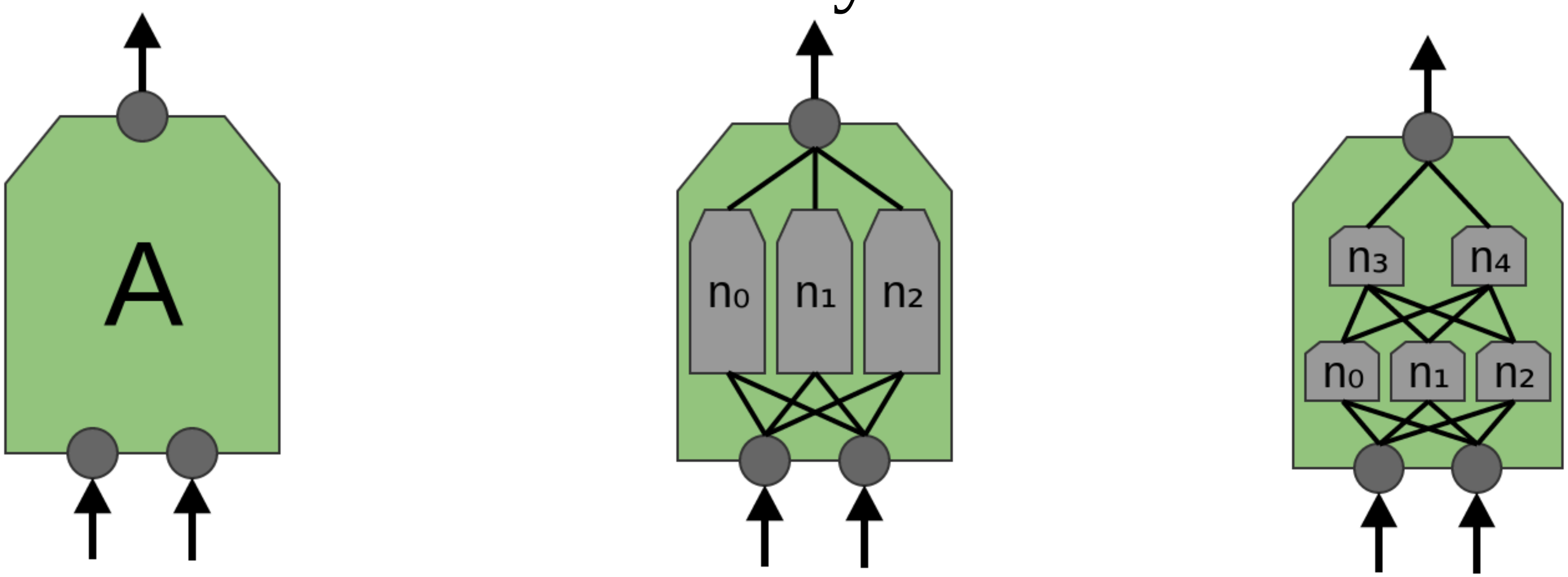
but more importantly, since we've asked if the same “A” feature is present at each localization, we can reuse the same “A” filter for each location! (“weight sharing”)



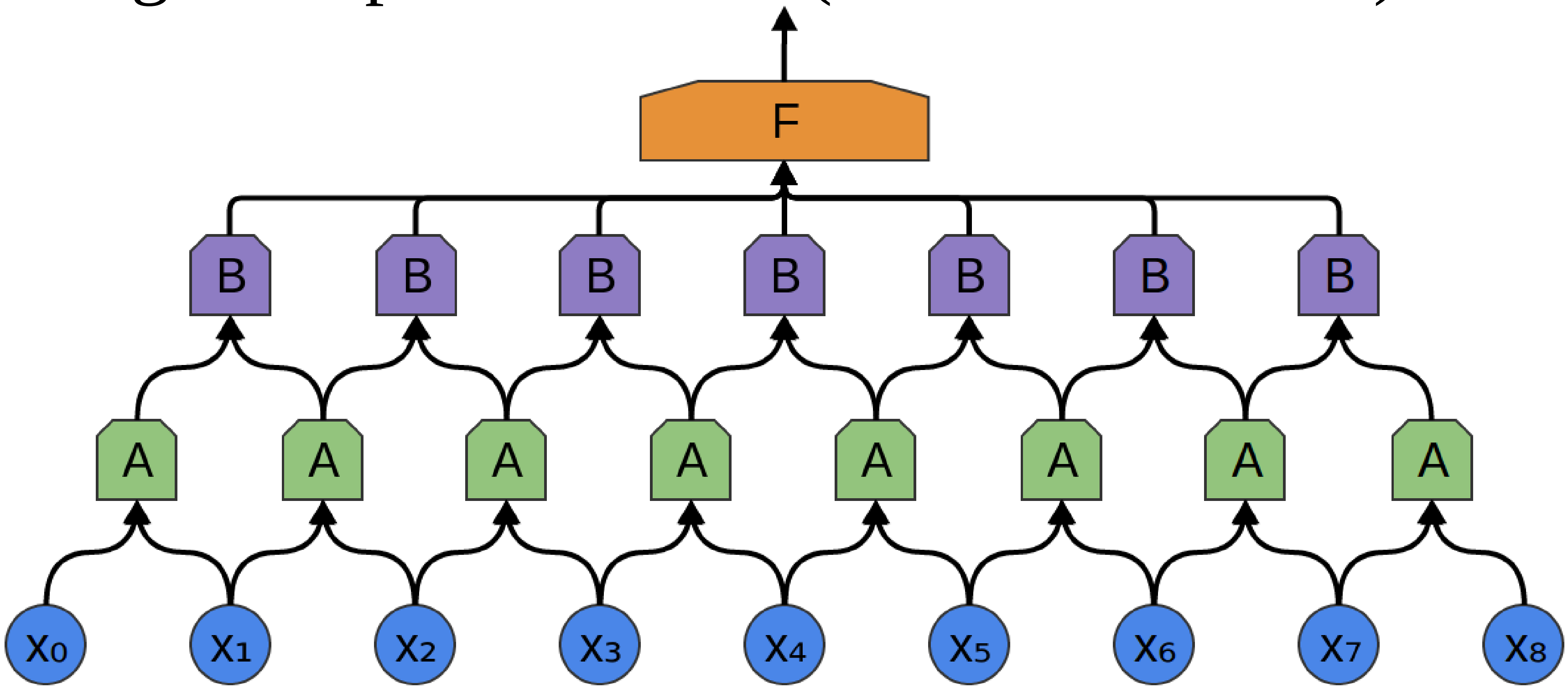
it also means that each of those “A” filters, only needs to take 2 inputs, and spit out 1 output – this is cheap!

so cheap that we could even use a whole neural network to represent a complex mapping in “A” if we wanted to

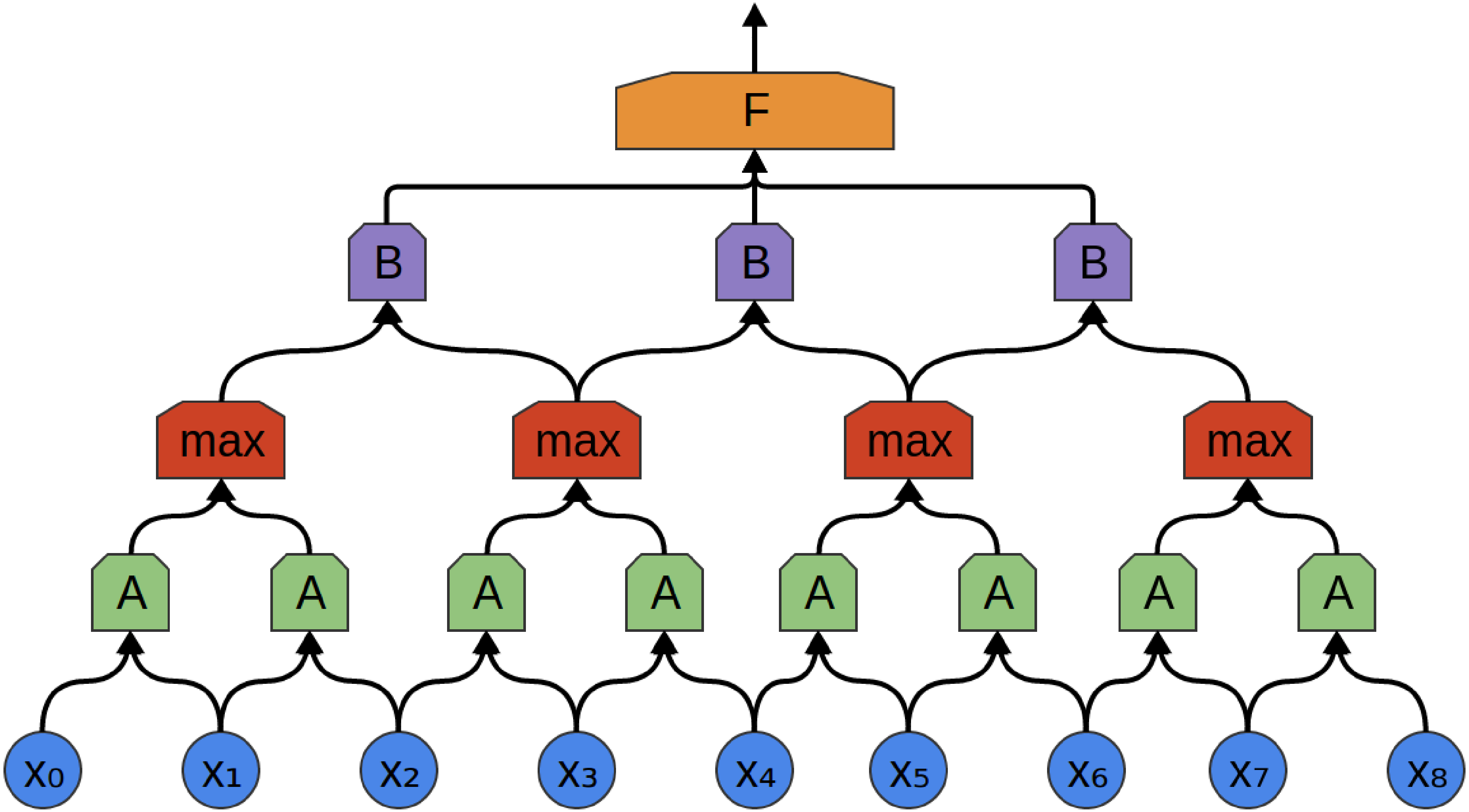
even a multi-layer network...



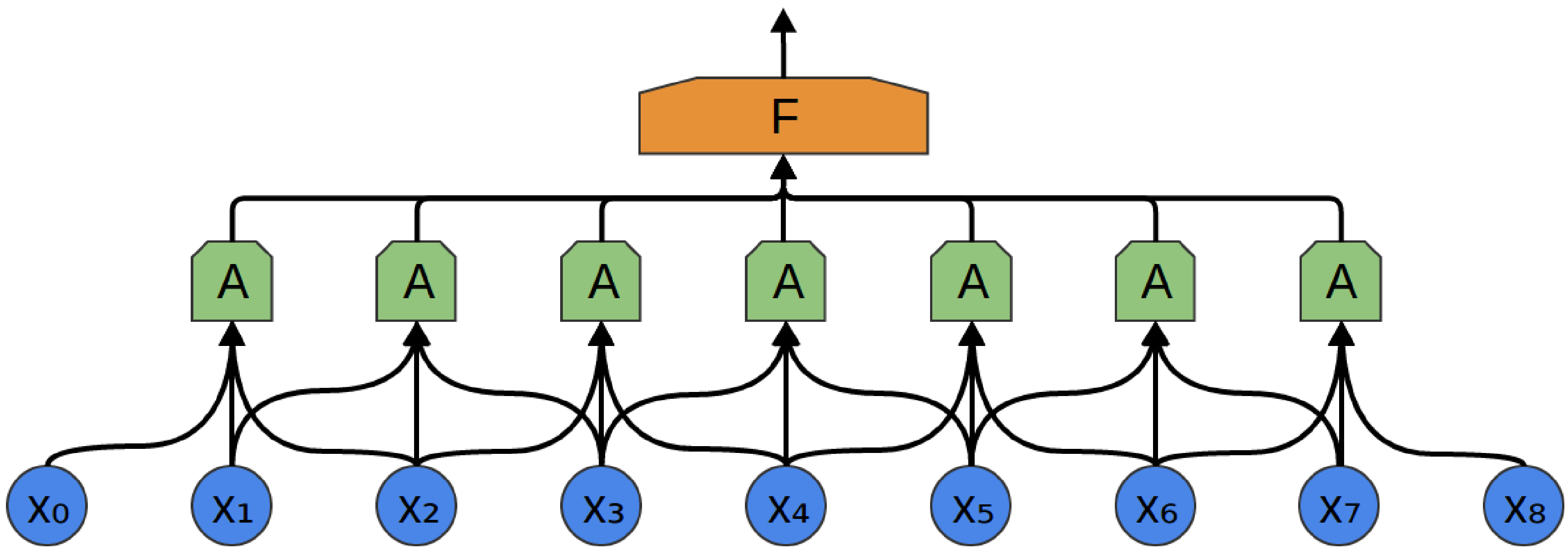
at the next layer, we could do the same thing again, learning a different local feature extractor, compressing the image to represent fewer (but more abstract) features



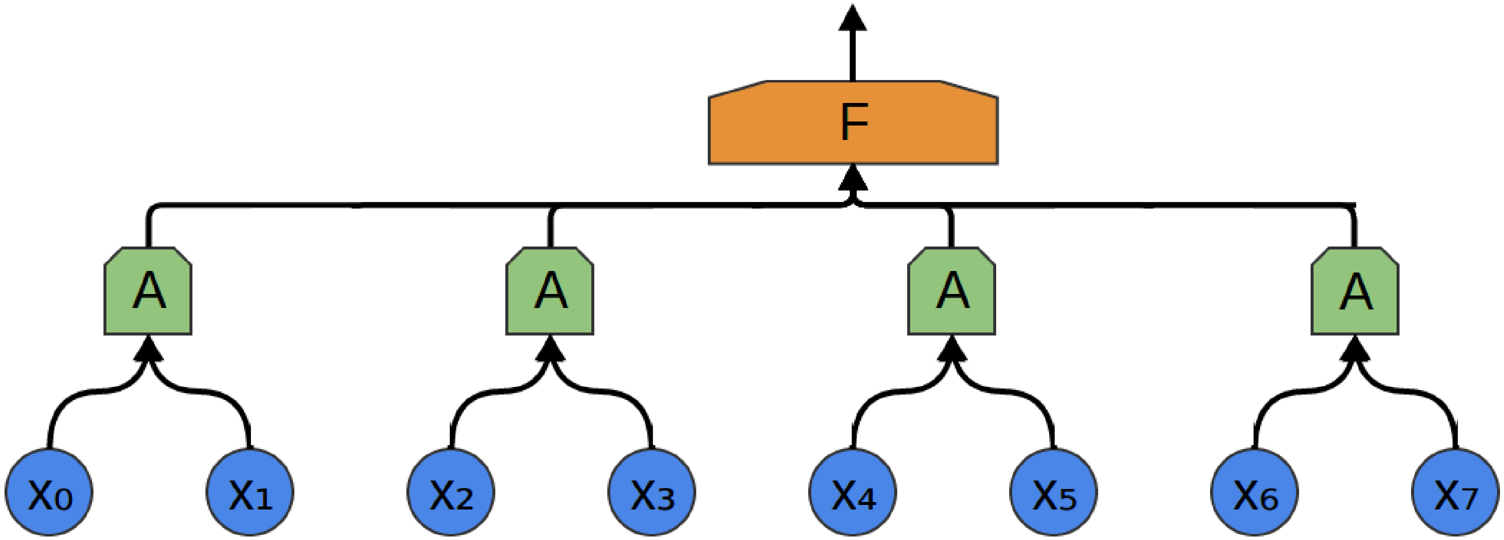
until our final decision is a simple one of few attributes



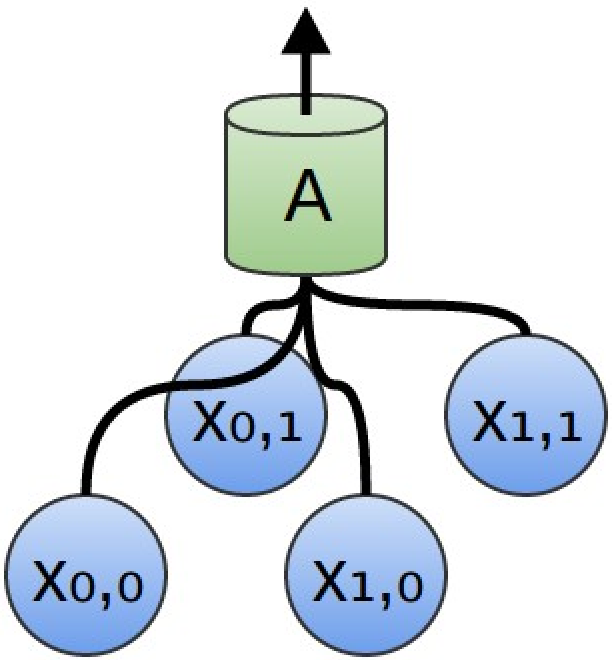
filter's can have arbitrary size, to capture features with larger spacial dependencies (e.g. 3-pixel wide features here)



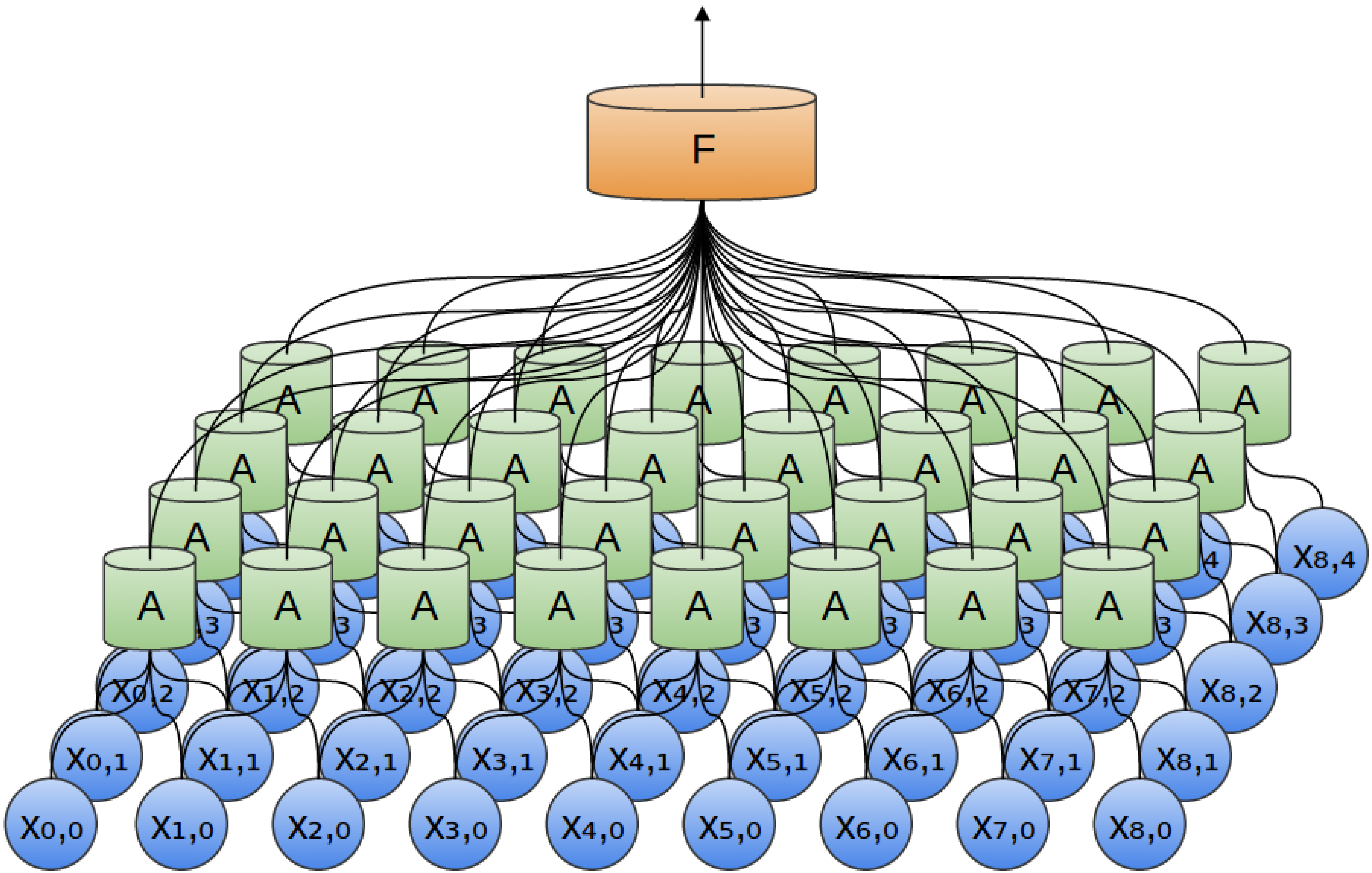
and can be applied with any frequency throughout the image
to compress it more at each layer
(e.g. a “stride” of 2, takes 2 steps between filter queries)



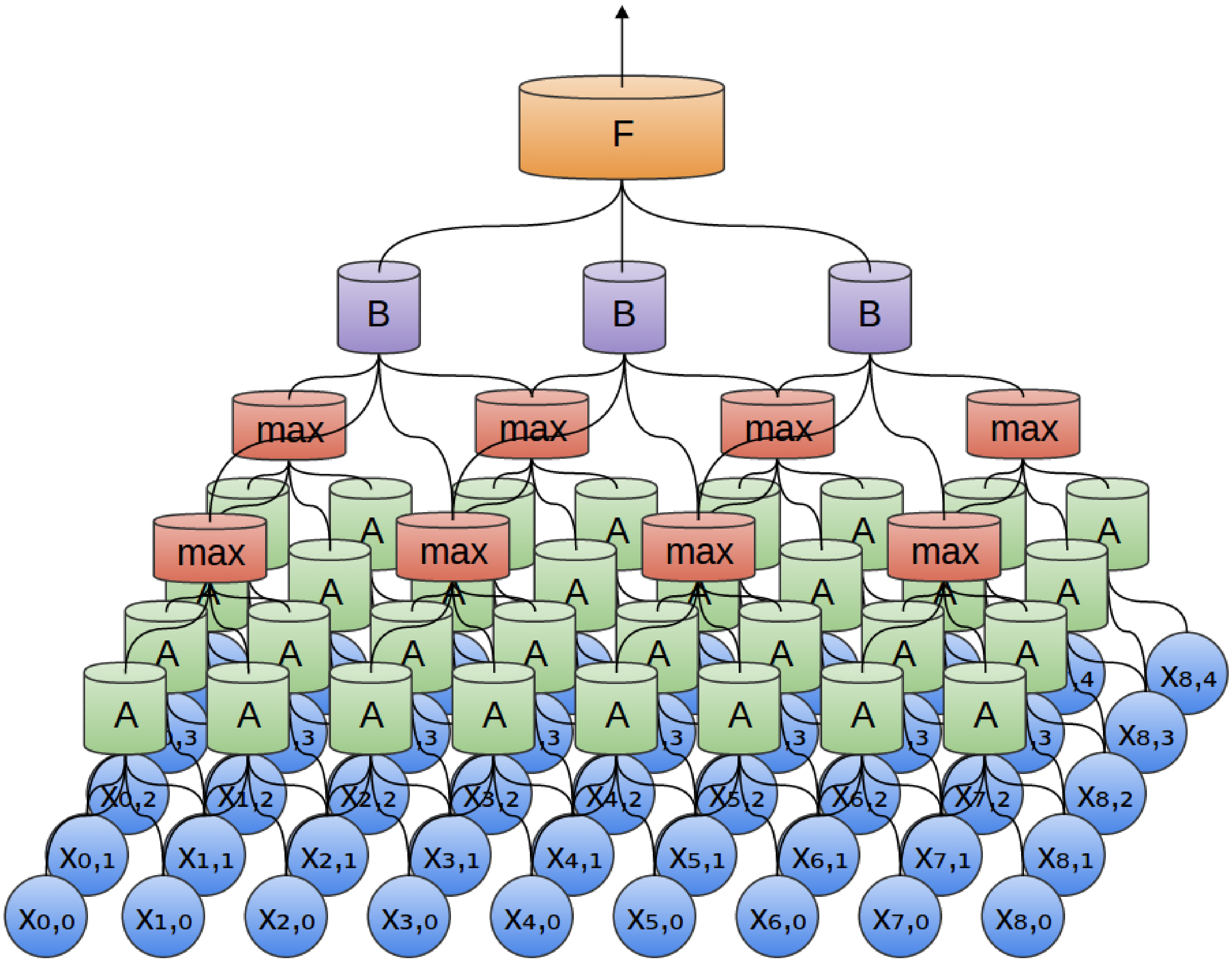
of course for a 2-D image, the filters will be 2-D



but the idea of locally applying repeated filters is the same

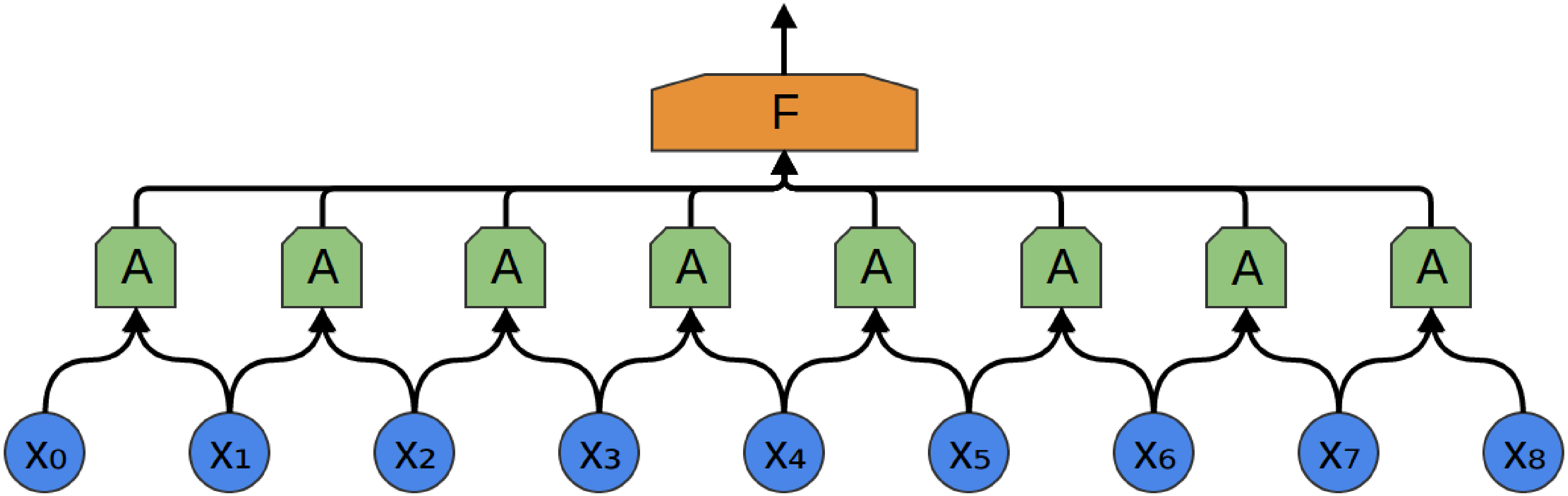


in fact, compression/abstraction are now happening in multiple dimensions at once!



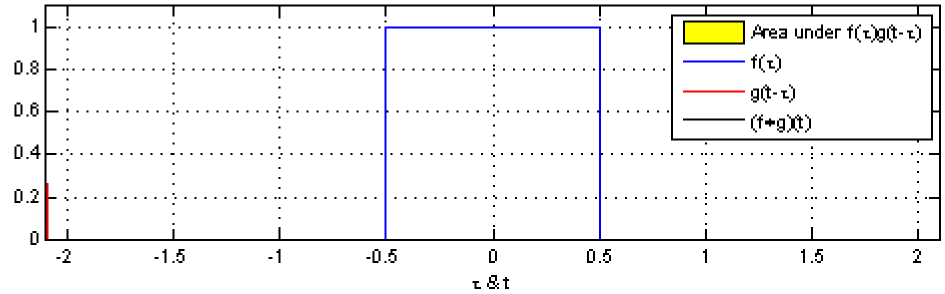
as a side note, this does not have to be done over pixels

if each x_t is an observation in time, then convolution would be finding local features/events at each time step (e.g. did a stock price go up/down compared to yesterday?)

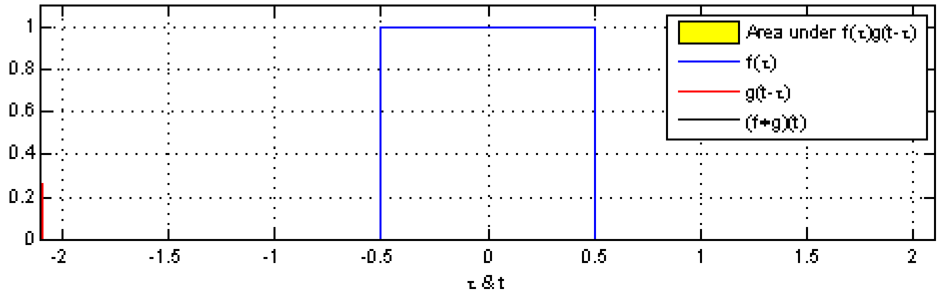


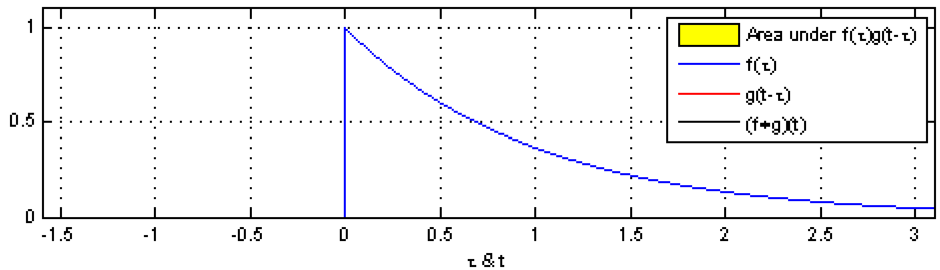
how is this “convolution”… and what is that?

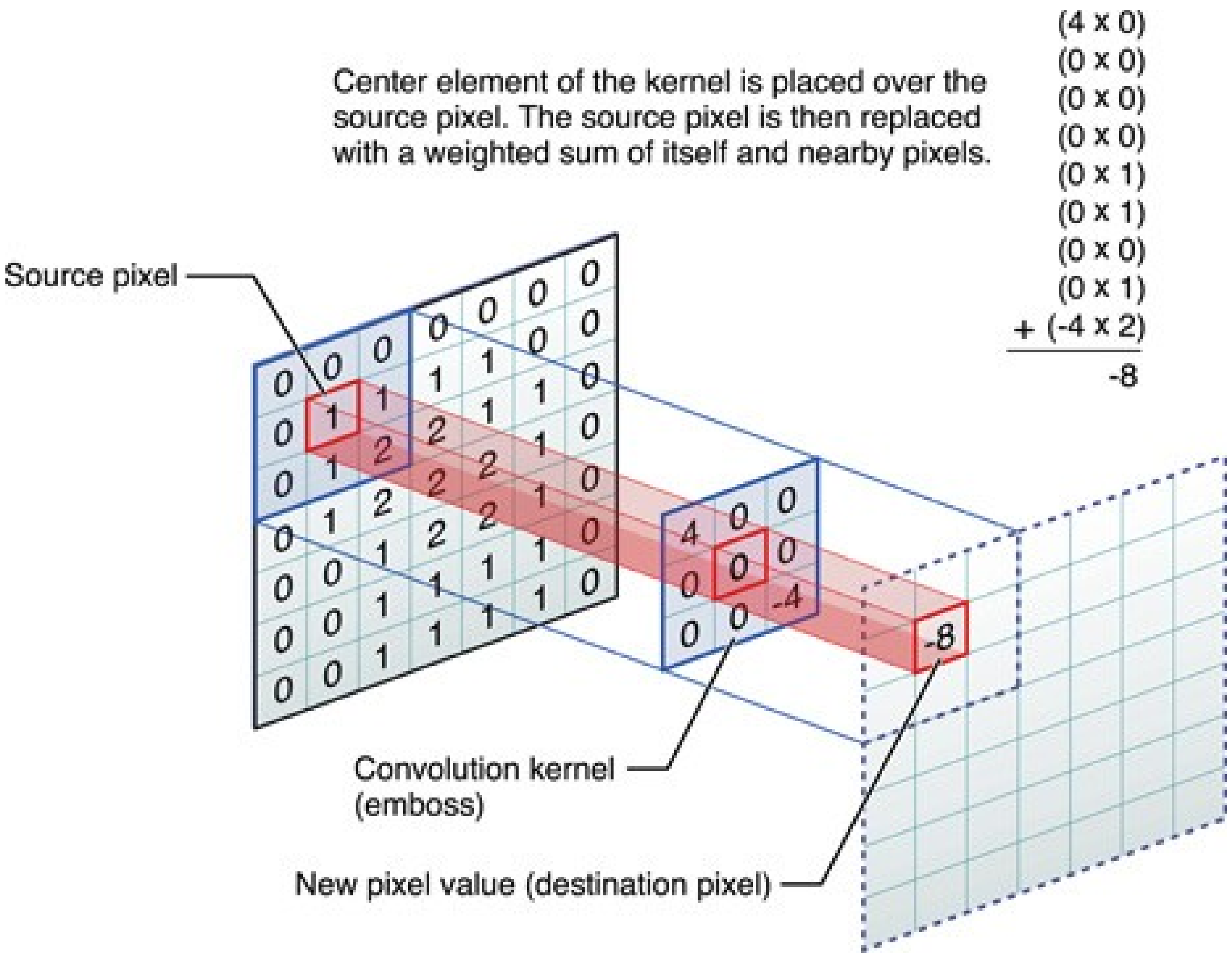
the pointswise overlap (multiplication) of two signals
(e.g. cross-correlation of how strongly
our filter is expressed in our image)



the pointswise overlap (multiplication) of two signals
(e.g. cross-correlation of how strongly
our filter is expressed in our image)







(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Blur

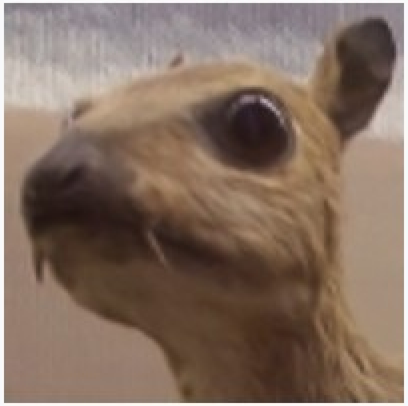
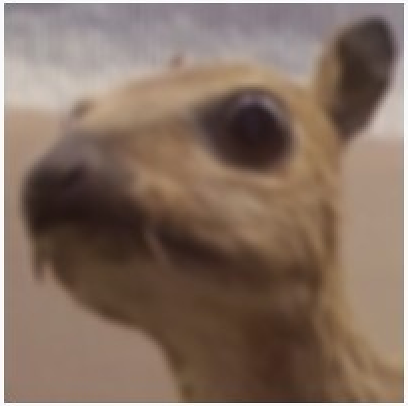
Median

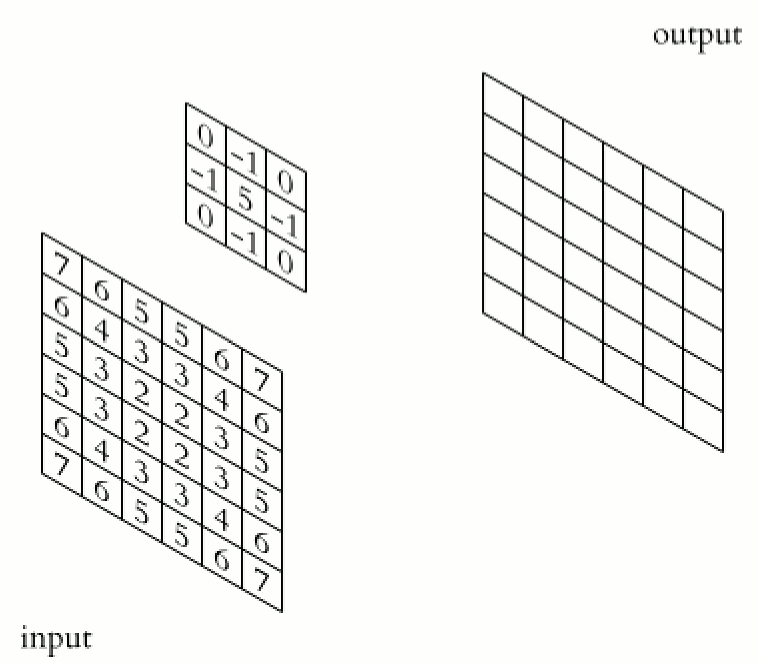
Edge-Detect

High-Pass

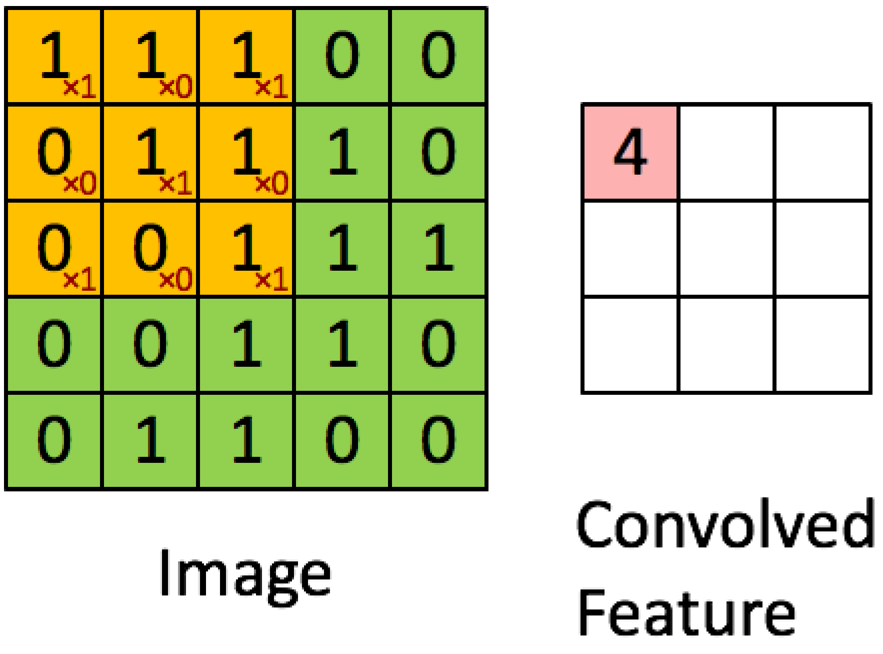
Dilate

Erode

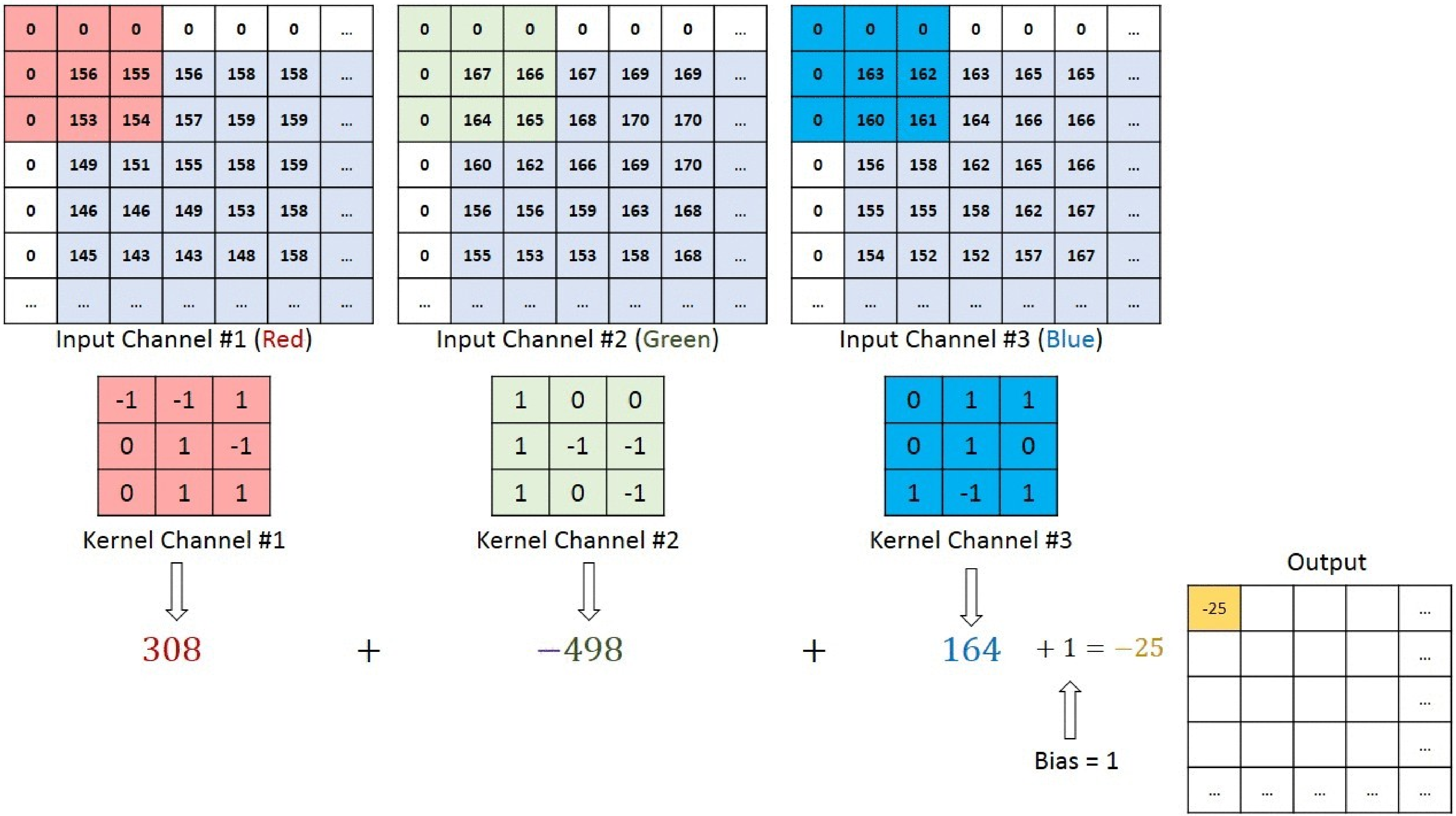
Identity	$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$	
Box blur (normalized)	$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$	
Edge detection	$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$	


output
0 -1 0
-1 5 -1
0 -1 0
7 6 5 5 6 7
6 4 3 3 4 6
5 3 2 2 3 5
5 3 2 2 3 5
6 4 3 3 4 6
7 6 5 5 6 7
input

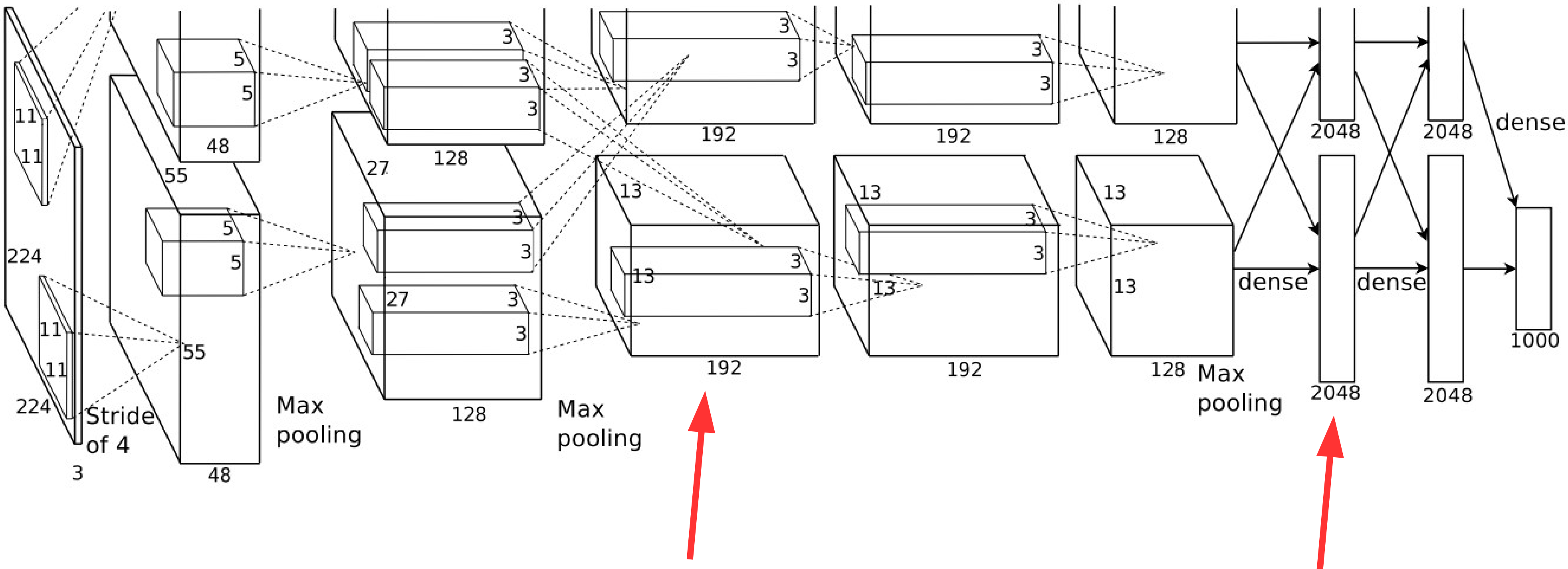
applying this process with a given filter size (e.g. 3x3) and a given stride (e.g. 1) produces a compressed array (next layer activation) of how strongly that filter was present in our previous layer



often we'll apply many filters (e.g. for RGB channels) to spit out one output, or many different features



“Alexnet” (2012)



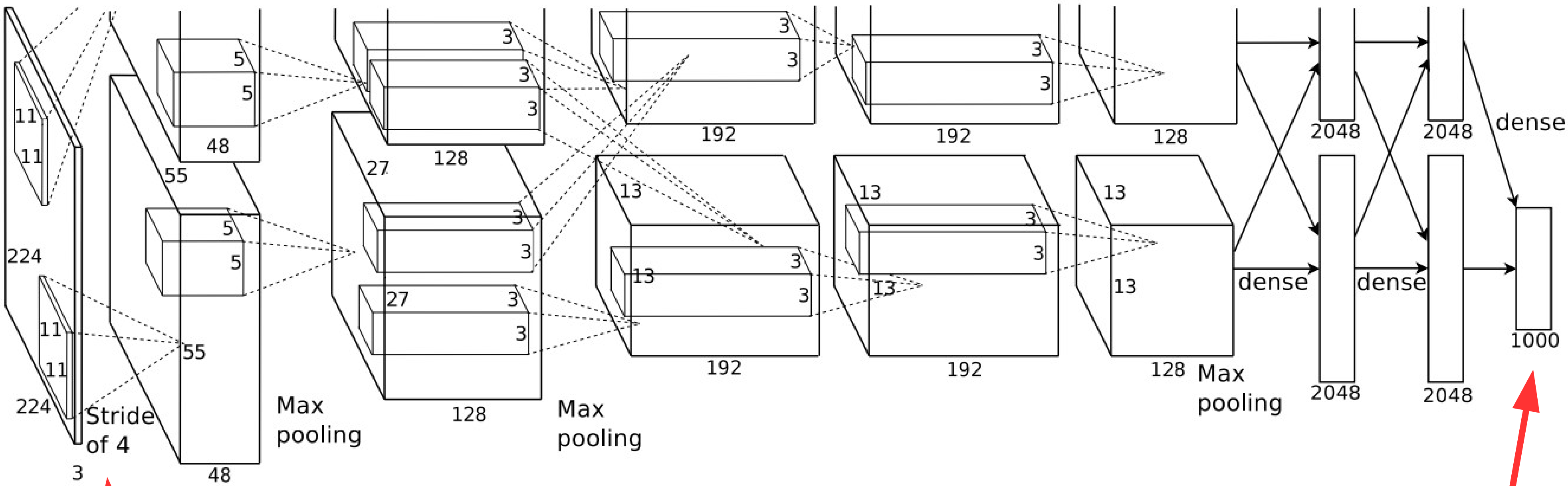
192 * 3 * 3 =
1,728 weights
(applied 13 times)

(convolutional
layer)

2048 * 2048 =
4,194,304 weights
(applied 13 times)

(fully connected
layer)

“Alexnet” (2012)



input an RGB image (scaled down to 224*224 pixels)

output the probability that the images belongs to any of 1000 classes