Convolutional Neural Networks

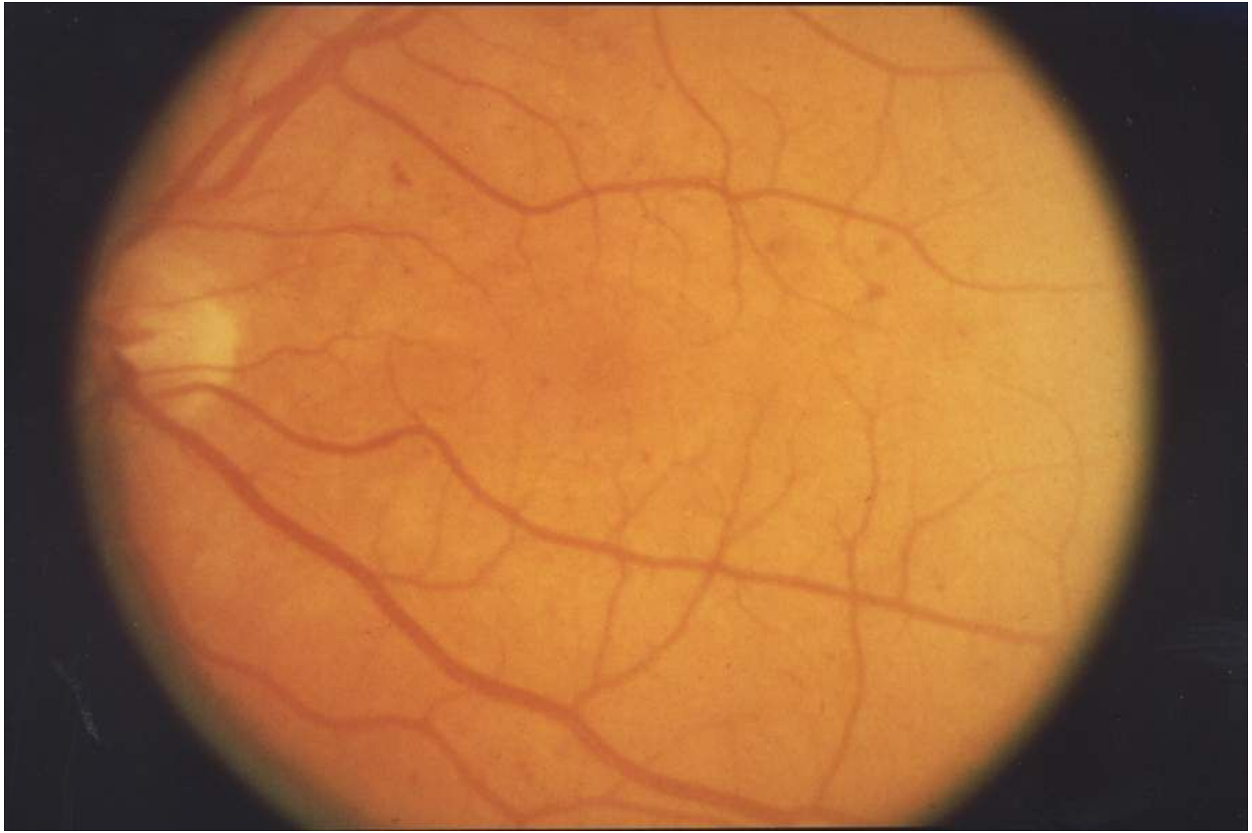
PHYS 555

So far: vectorized inputs

- aka tabular data
- aka catalogues
- aka structured data

age	sex	bmi	children	smoker	region	charges
19	female	27.9	0	yes	southwest	16884.924
18	male	33.77	1	no	southeast	1725.5523
28	male	33	3	no	southeast	4449.462
33	male	22.705	0	no	northwest	21984.47061
32	male	28.88	0	no	northwest	3866.8552
31	female	25.74	0	no	southeast	3756.6216
46	female	33.44	1	no	southeast	8240.5896
37	female	27.74	3	no	northwest	7281.5056
37	male	29.83	2	no	northeast	6406.4107
60	female	25.84	0	no	northwest	28923.13692
25	male	26.22	0	no	northeast	2721.3208
62	female	26.29	0	yes	southeast	27808.7251
23	male	34.4	0	no	southwest	1826.843
56	female	39.82	0	no	southeast	11090.7178
27	male	42.13	0	yes	southeast	39611.7577
19	male	24.6	1	no	southwest	1837.237
52	female	30.78	1	no	northeast	10797.3362
23	male	23.845	0	no	northeast	2395.17155
56	male	40.3	0	no	southwest	10602.385
30	male	35.3	0	yes	southwest	36837.467
60	female	36.005	0	no	northeast	13228.84695
30	female	32.4	1	no	southwest	4149.736
18	male	34.1	0	no	southeast	1137.011
34	female	31.92	1	yes	northeast	37701.8768
37	male	28.025	2	no	northwest	6203.90175
59	female	27.72	3	no	southeast	14001.1338
63	female	23.085	0	no	northeast	14451.83515
55	female	32.775	2	no	northwest	12268.63225
23	male	17.385	1	no	northwest	2775.19215
31	male	36.3	2	yes	southwest	38711
22	male	35.6	0	yes	southwest	35585.576
18	female	26.315	0	no	northeast	2198.18985

What can we do with unstructured data?



Could we apply an MLP for images?

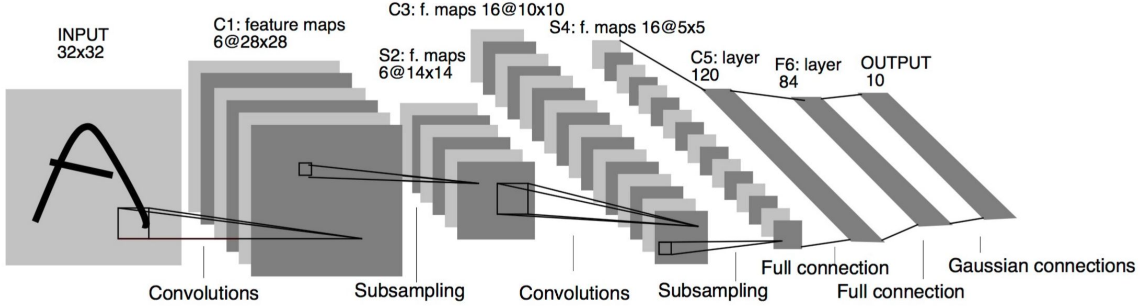
Take an RGB 640 x 480 image.

- First layer: as many as input data
- Second layer 1000 neurons

Number of parameters: $640 \times 480 \times 3 \times 1000 + 1000 = 922M$!

Spatial organization of the input is destroyed!

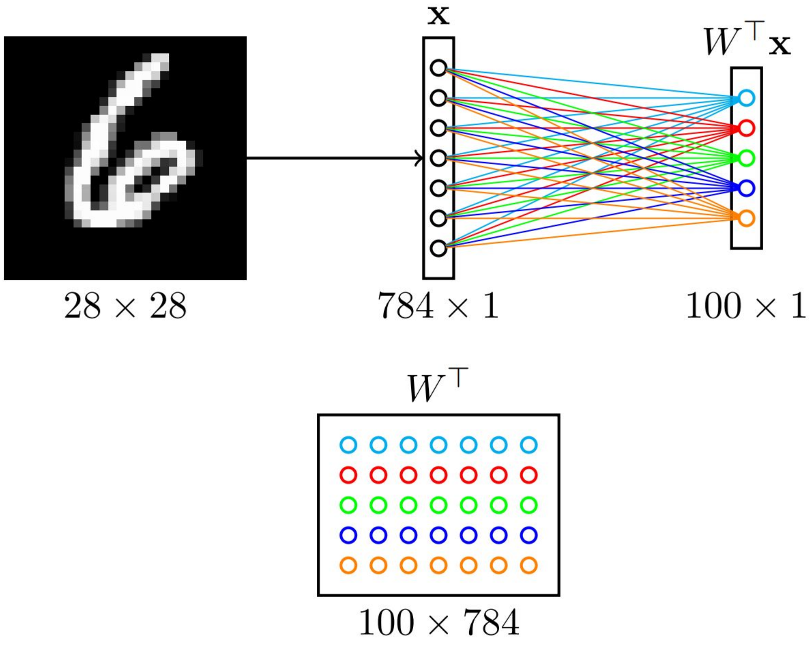
Convolutional Neural Network (CNN or ConvNet)



Motivations for convolutions

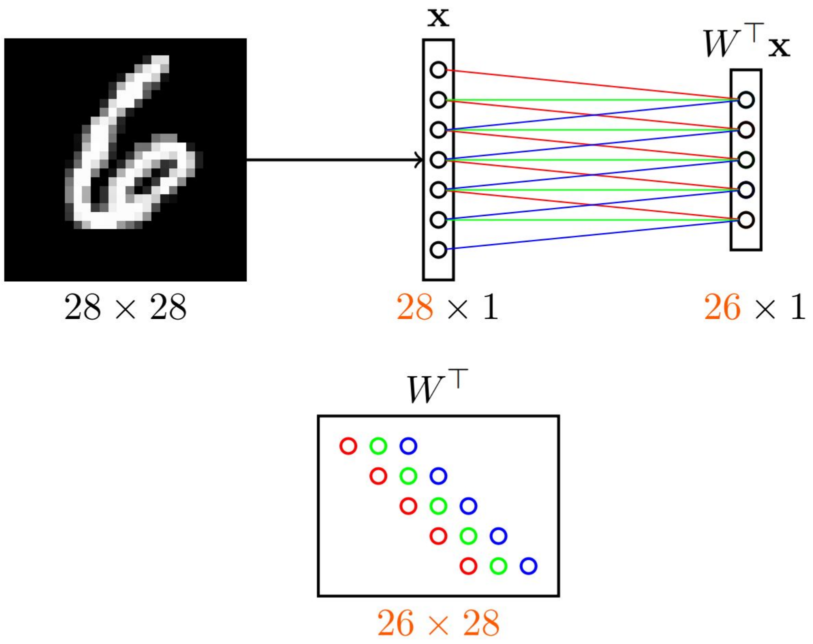
- Local connectivity
  - a neuron depends only on a few local input neurons
  - translation invariance
- Comparison to a fully connected
  - Parameter sharing: reduces overfitting
  - Make use of spatial structure, strong prior for vision

CNN are simpler than ANN



- each row of $W^\top$ yields one activation element (cell)
- each cell is fully connected to all input elements

CNN are simpler than ANN



- this is an 1d convolution and generalizes to 2d
- this new mapping is a convolutional layer

Convolutions recap

Discrete convolution (actually cross-correlation) between two functions f and g:

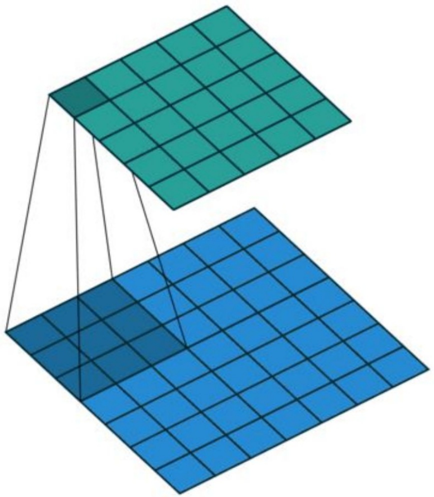
$$(f \star g)(x) = \sum_{a+b=x} f(a).\, g(b) = \sum_{a} f(a).\, g(x+a)$$

2D-convolutions (actually 2D cross-correlation):

$$(f \star g)(x, y) = \sum_{n} \sum_{m} f(n, m).\, g(x+n, y+m)$$

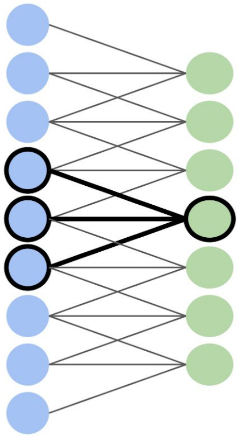
f is a convolution **kernel** or **filter** applied to the 2-d map g (our image)

Convolutions

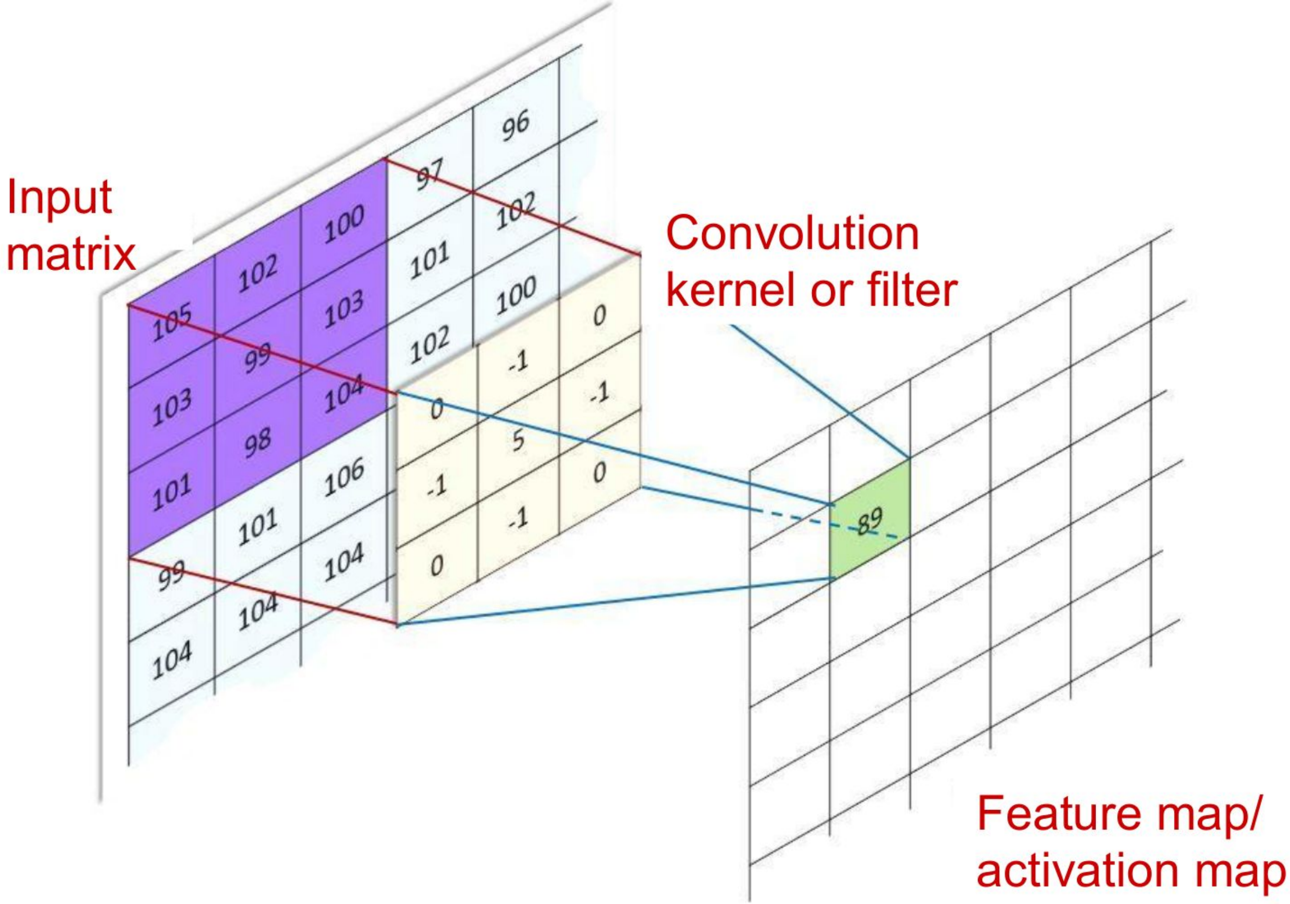


CNNs slide the same kernel of weights across their input, thus have local sparse connectivity and tied weights

Sparse connectivity
+ parameter sharing

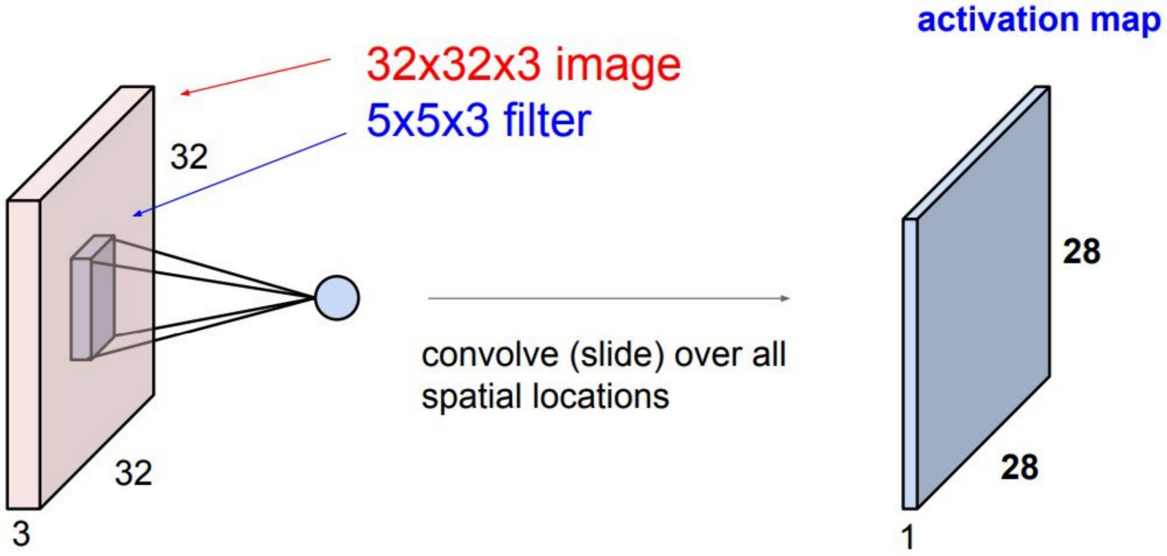


Parameters are shared (tied weights) across all neurons



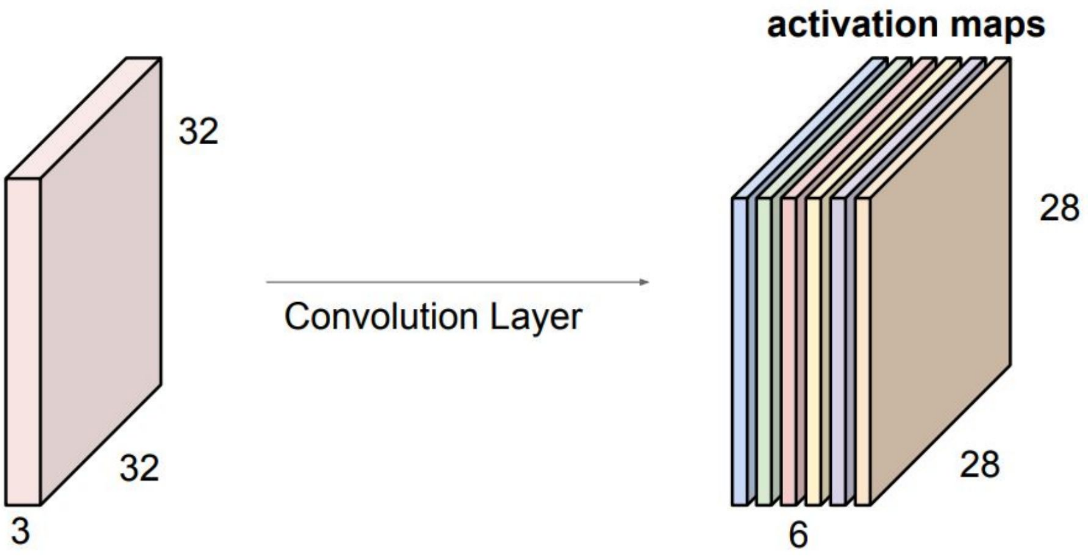
Input matrix
Convolution kernel or filter
89
Feature map/ activation map

Channels and output dimensions



CNNs output dimensions

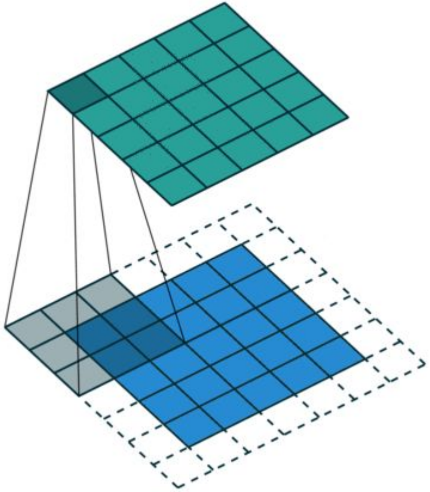
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

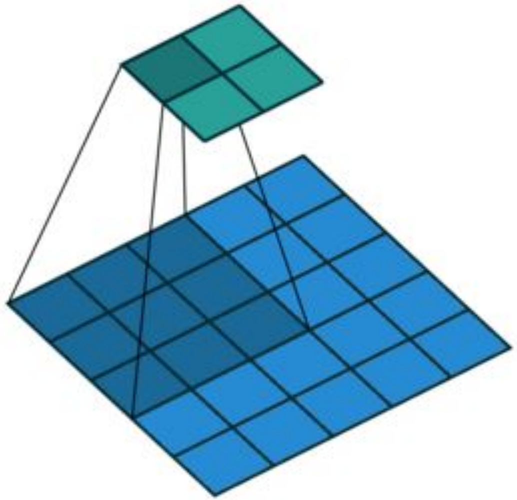
Padding

- Padding: artificially fill borders of image
- Useful to keep spatial dimension constant across filters
- Useful with strides and large receptive fields
- Usually: fill with 0s



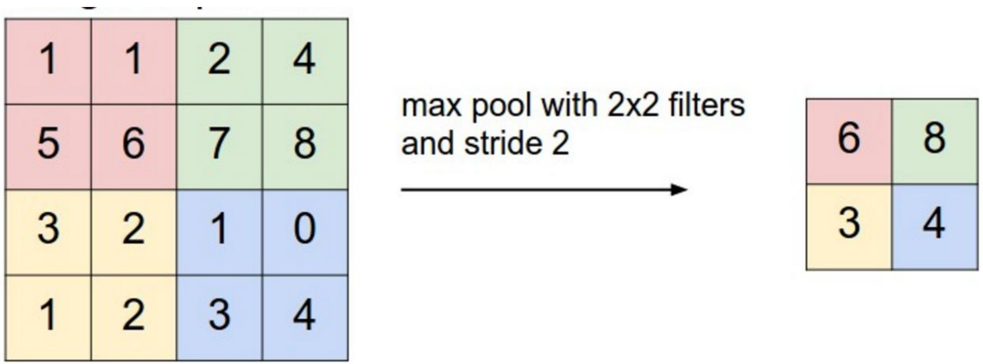
Strides

- Strides: increment step size for the convolution operator
- Reduces the size of the output map



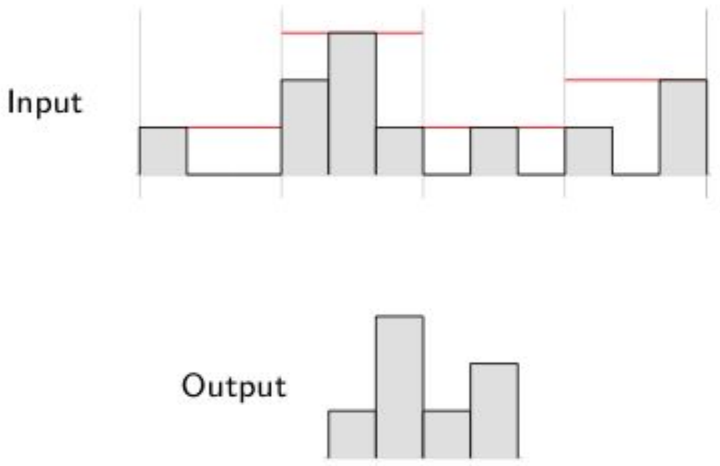
Pooling

- Spatial dimension reduction
- Local invariance
- No parameters: max or average of inputs



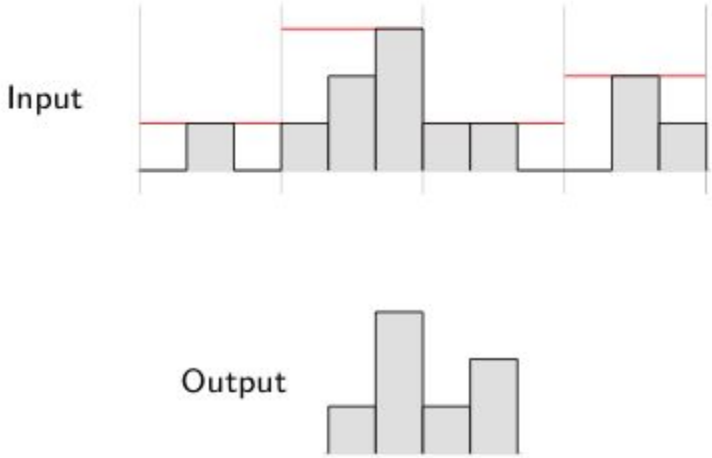
Pooling adds some translation invariance

Ex: Max pooling



Pooling adds some translation invariance

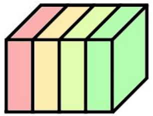
Ex: Max pooling



Layer and filter shapes

Kernel or **Filter** shape (F, F, C^i, C^o)

- $F \times F$ kernel size,
- C^i input channels
- C^o output channels



Number of parameters: $(F \times F \times C^i + 1) \times C^o$

Activations or **Feature maps** shape:

- Input (W^i, H^i, C^i)
- Output (W^o, H^o, C^o)

$$W^o = (W^i - F + 2P)/S + 1$$

Determining the sizes of the convolutional layers

Wi: size of input vector

F: size of filter

P: size of padding

S: size of strides

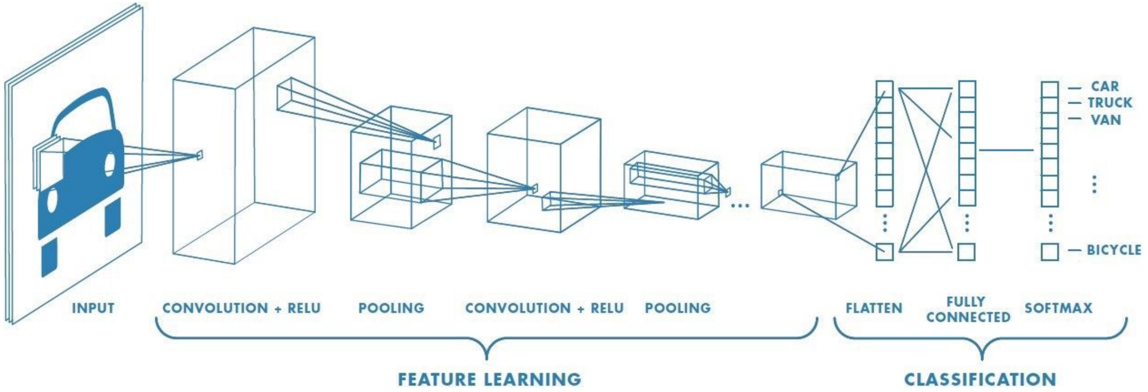
Wo: size of output

$$W^o = \texttt{floor}\left(\frac{W^i - F + 2P}{S} + 1\right)$$

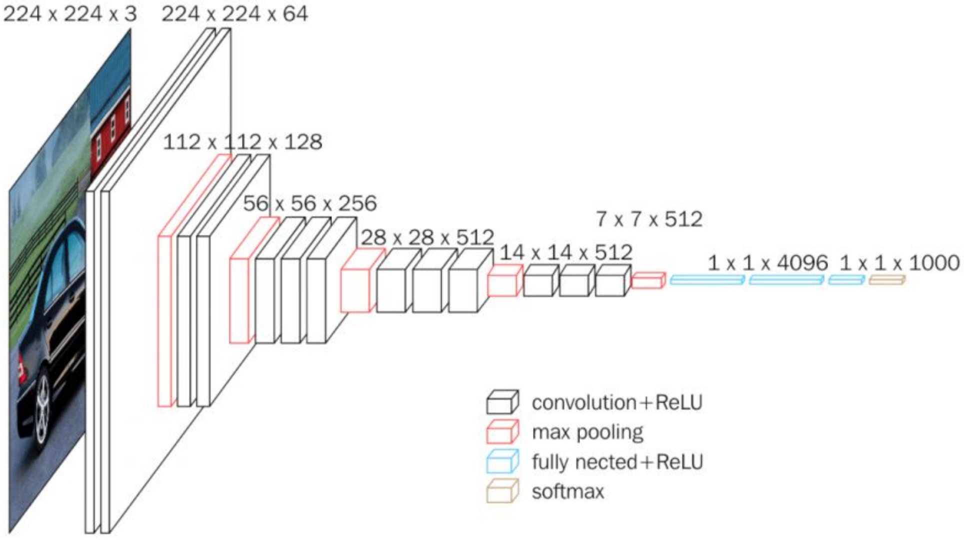
Putting it all together

Classic ConvNet architecture:

- Conv Blocks:
  - Convolution + ReLU
  - Convolution + ReLU
  - …
  - Max pooling
- Flatten output
- Fully connected layers
- Softmax (for multi-class)



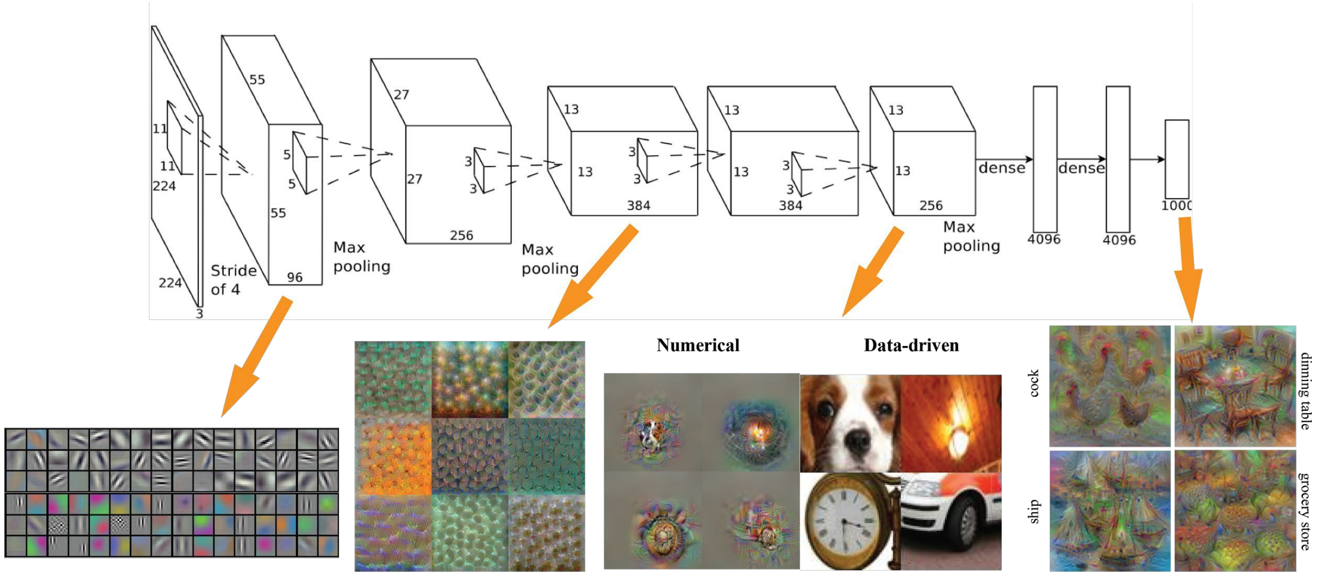
CNN example architecture and code (keras)




```python
model = Sequential()
model.add(ZeroPadding2D((1, 1), input_shape=(3, 224, 224)))
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(MaxPooling2D((2, 2), strides=(2, 2)))

model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(128, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(128, 3, 3, activation='relu'))
model.add(MaxPooling2D((2, 2), strides=(2, 2)))

model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(MaxPooling2D((2, 2), strides=(2, 2)))

model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2, 2), strides=(2, 2)))

model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))

# Add another conv layer with ReLU + GAP
model.add(Convolution2D(num_input_channels, 3, 3, activation='relu', border_mode="same"))
model.add(AveragePooling2D((14, 14)))
model.add(Flatten())
# Add the W layer
model.add(Dense(nb_classes, activation='softmax'))
```

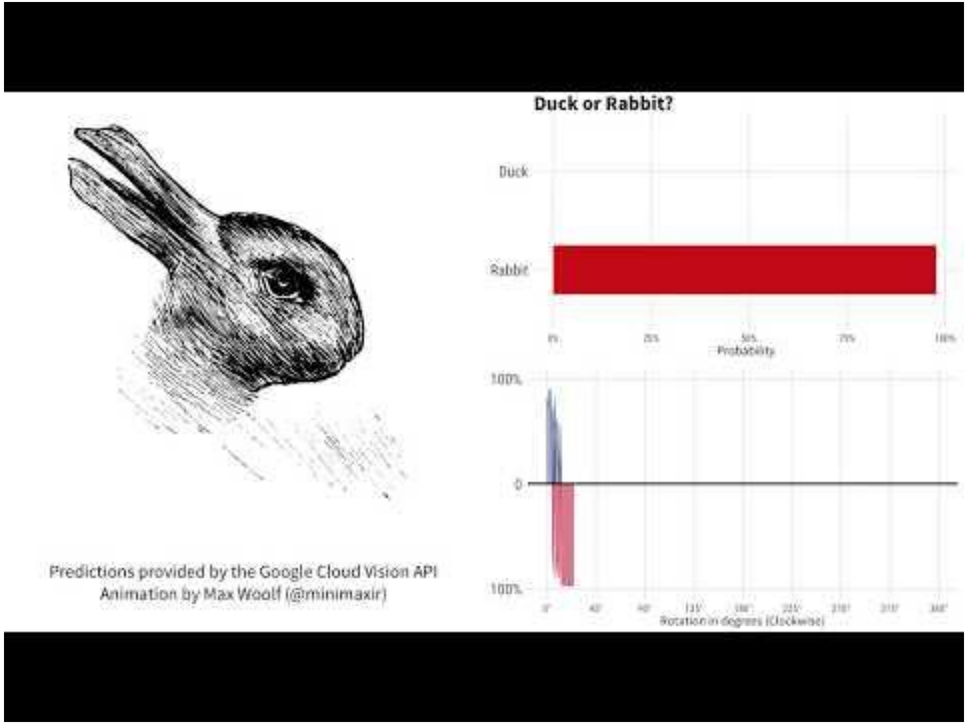
What do CNNs learn?

Low level --> High level



Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes

Duck or Rabbit?



CNN Helpers

- Correct your sizes: https://ezyang.github.io/convolution-visualizer/index.html
- A guide to convolution arithmetic in deep learning :in-depth practical guide.
- Convolutions from first principles