Convolutional Neural Networks

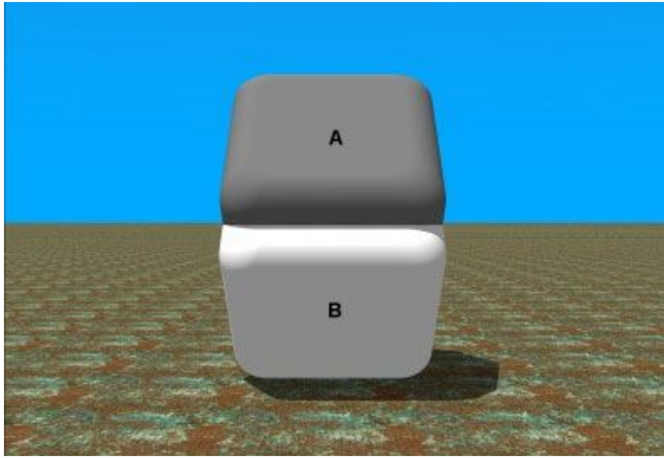
Andrew H. Fagg
Symbiotic Computing Laboratory
University of Oklahoma

Deep Networks for Image Recognition

- Images are composed of large numbers of pixels
- A particular pixel value can vary a lot:
  - Color, illumination
- Objects can vary a lot
  - Size, orientation, perspective

Individual pixels are irrelevant…

it is the groups of pixels that matter



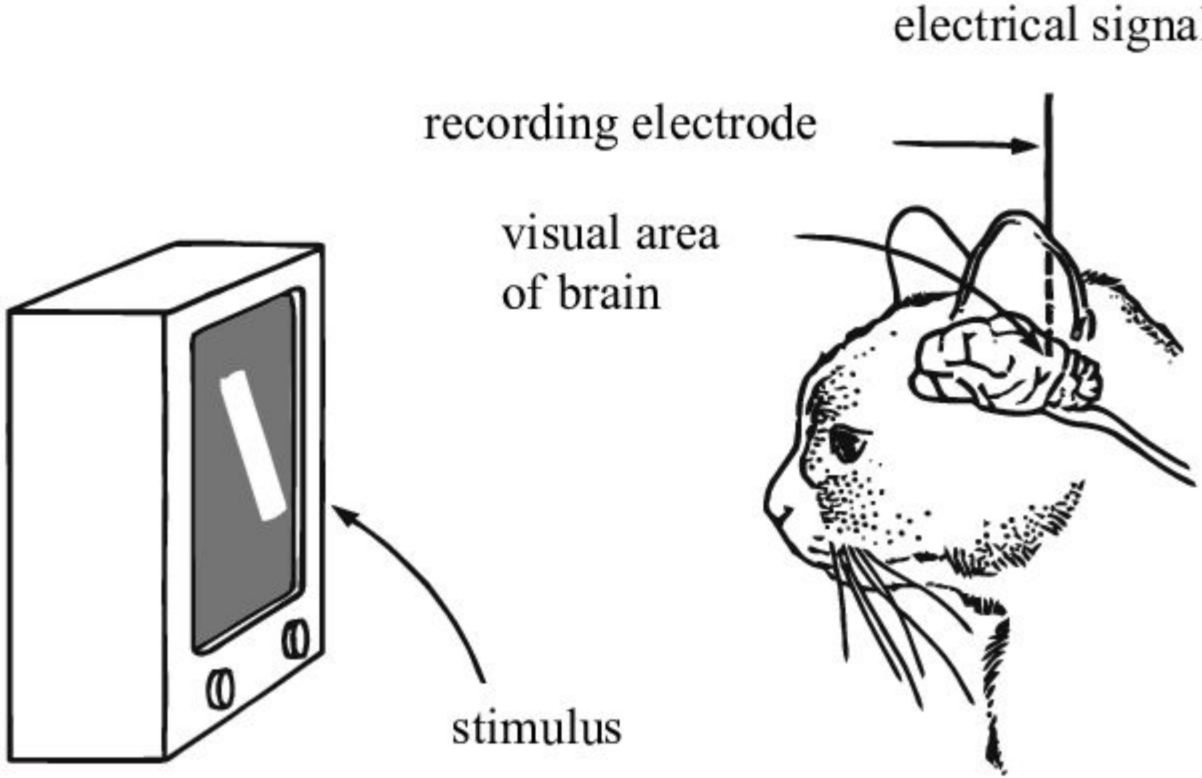
http://brainden.com/color-illusions.htm

Deep Networks for Image Recognition

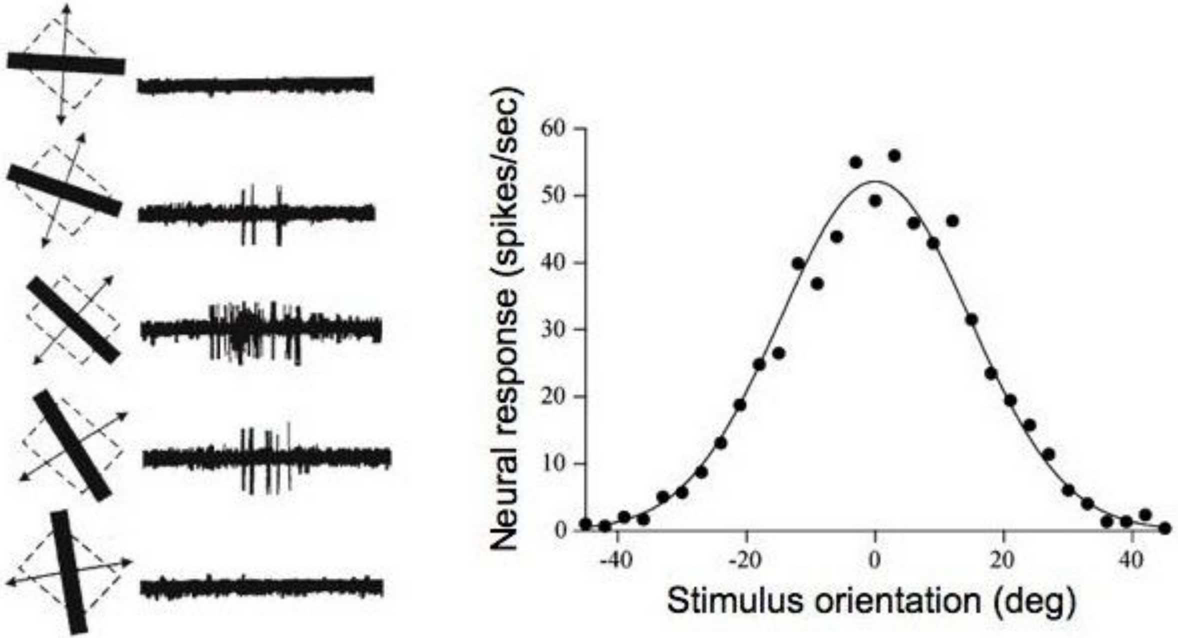
- 1920 (columns) x 1020 (rows) x 3 (channels = RGB) is almost 6 million inputs
- If the next dense layer has 1000 units, then we would have 6 billion parameters!

Need lots of examples and lots of training time. How do we get beyond this?

Hubel and Wiesel (1968)

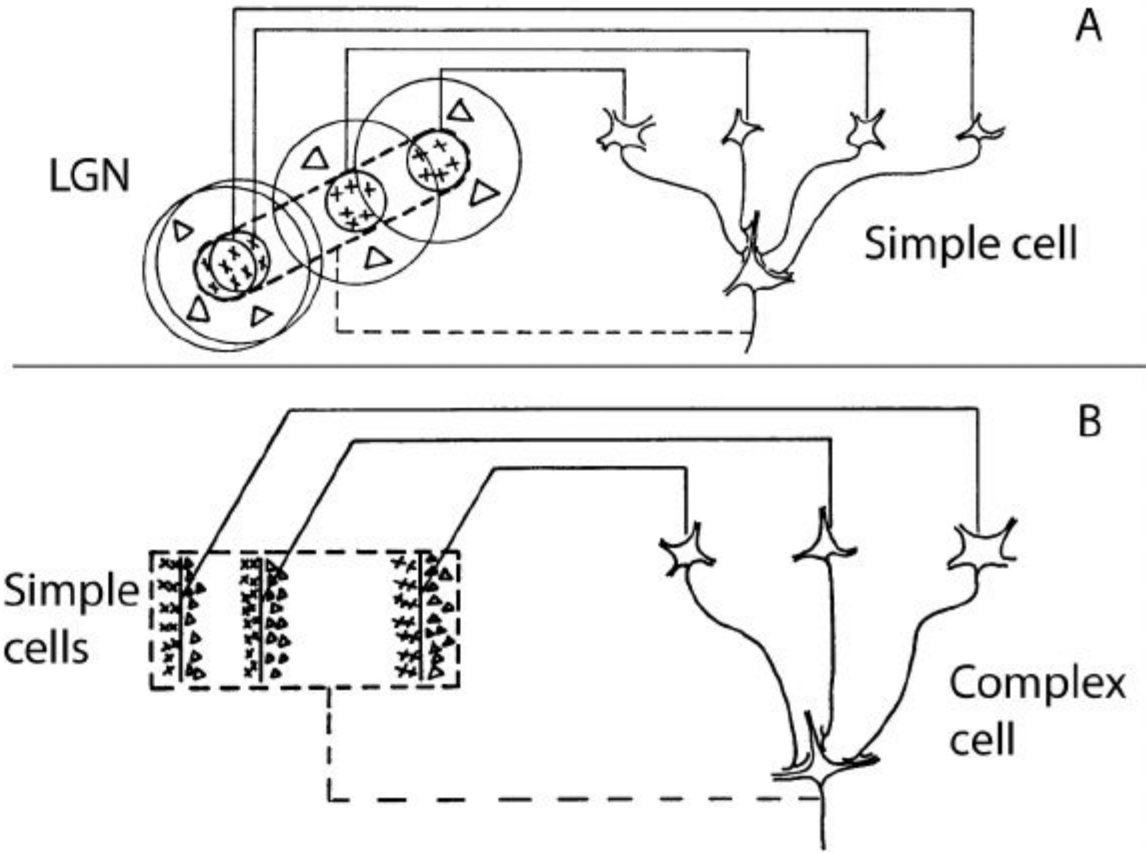


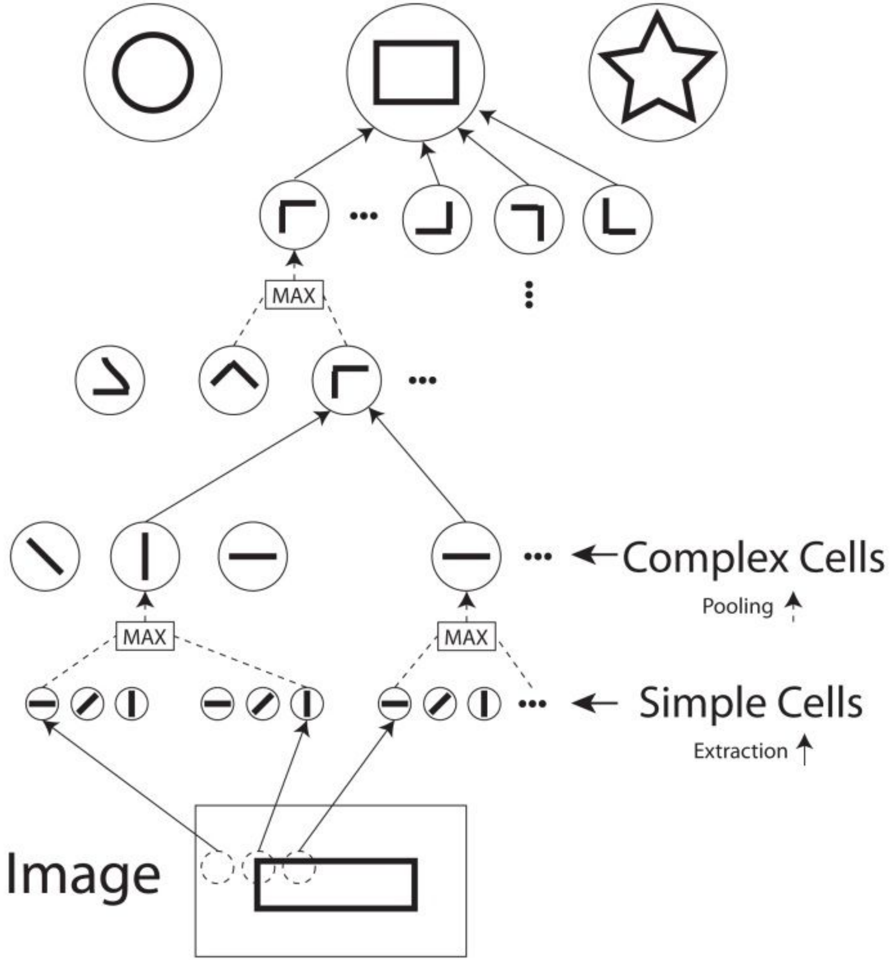
Orientation Sensitivity



Hubel & Wiesel, 1968

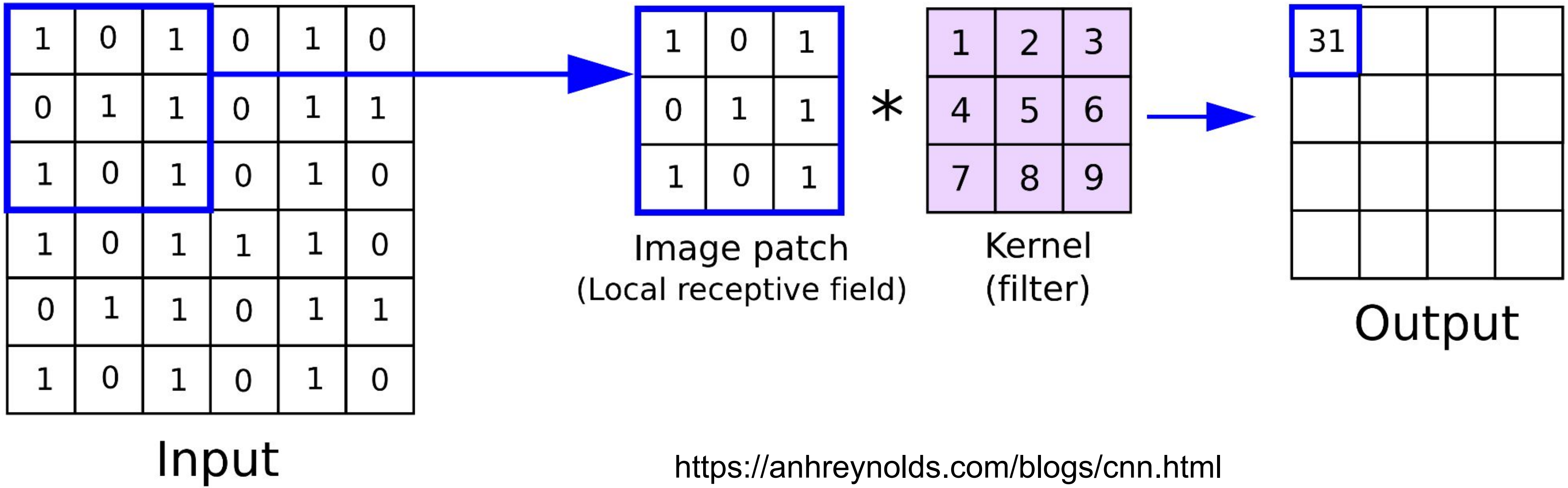
Complex Features Formed from Simple Ones





https://www.intechopen.com/books/visual-cortex-current-status-and-perspectives/models-of-information-processing-in-the-visual-cortex

Convolution

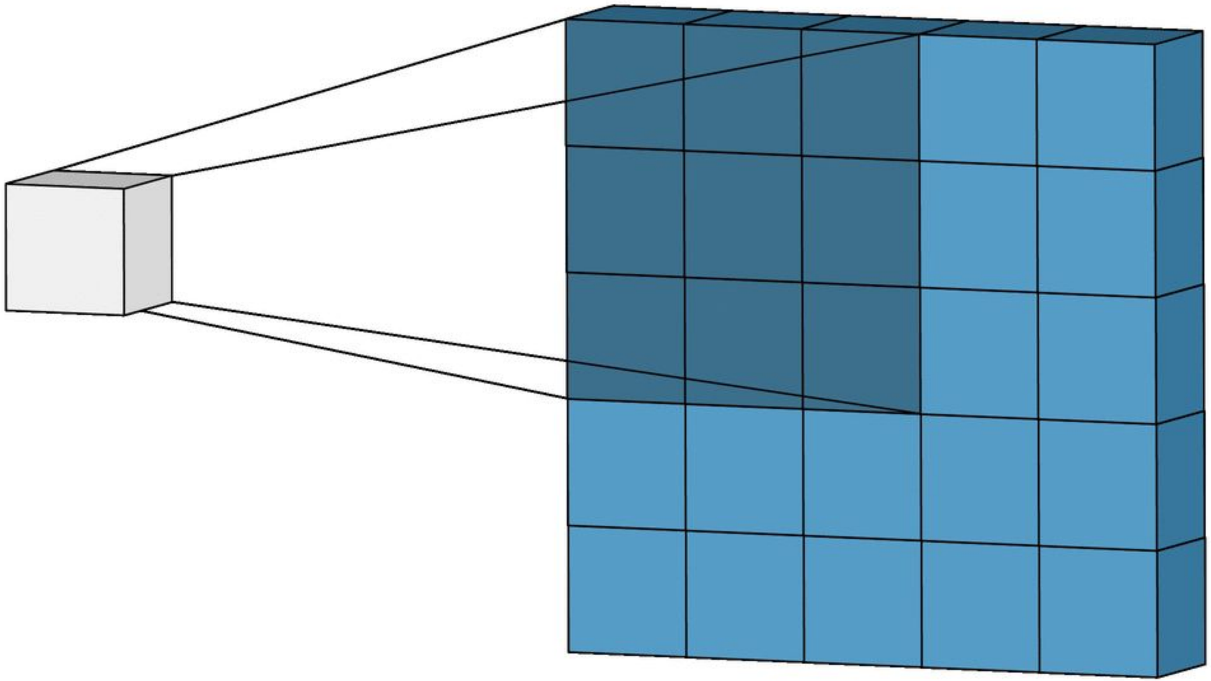


https://anhreynolds.com/blogs/cnn.html

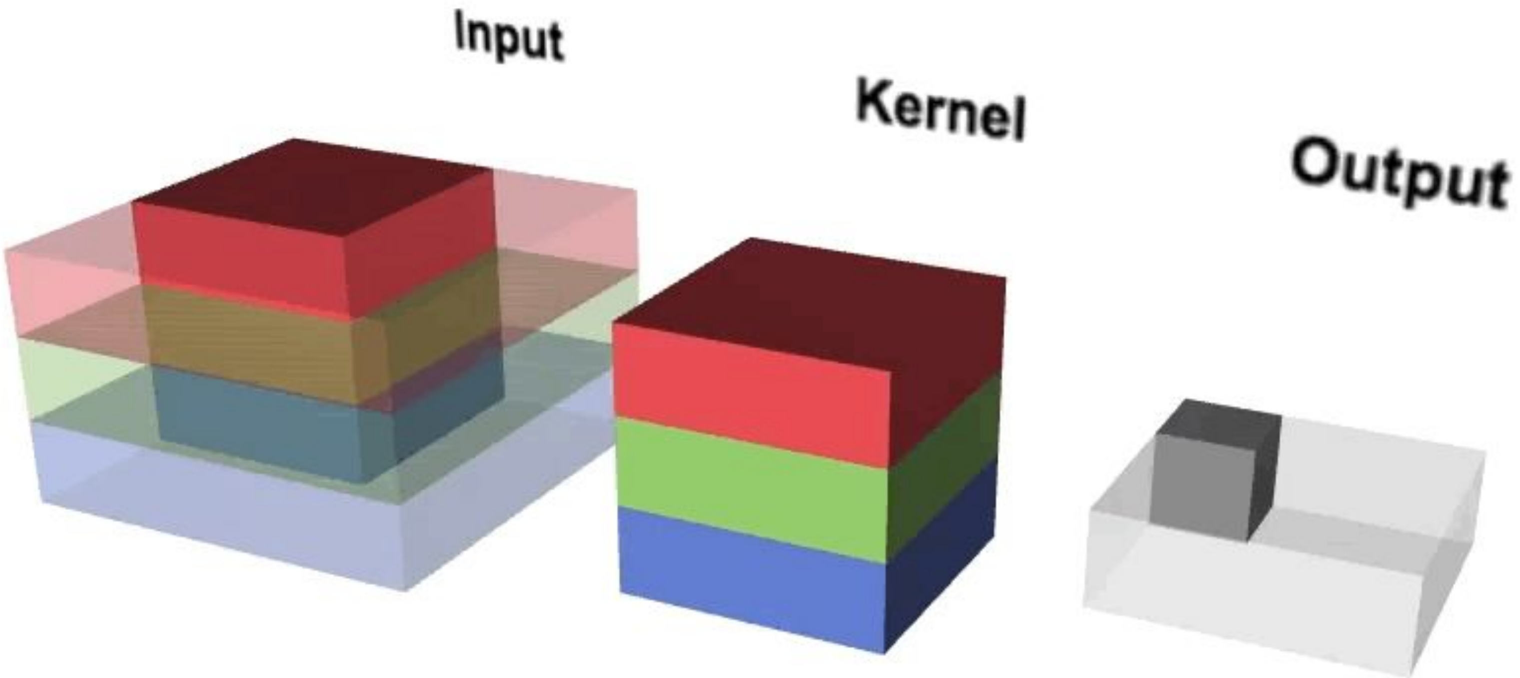
Convolution: Edge Detector

$$\begin{bmatrix} 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \\ 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \end{bmatrix} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{\text{Vertical}} = \begin{bmatrix} 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 \end{bmatrix}$$

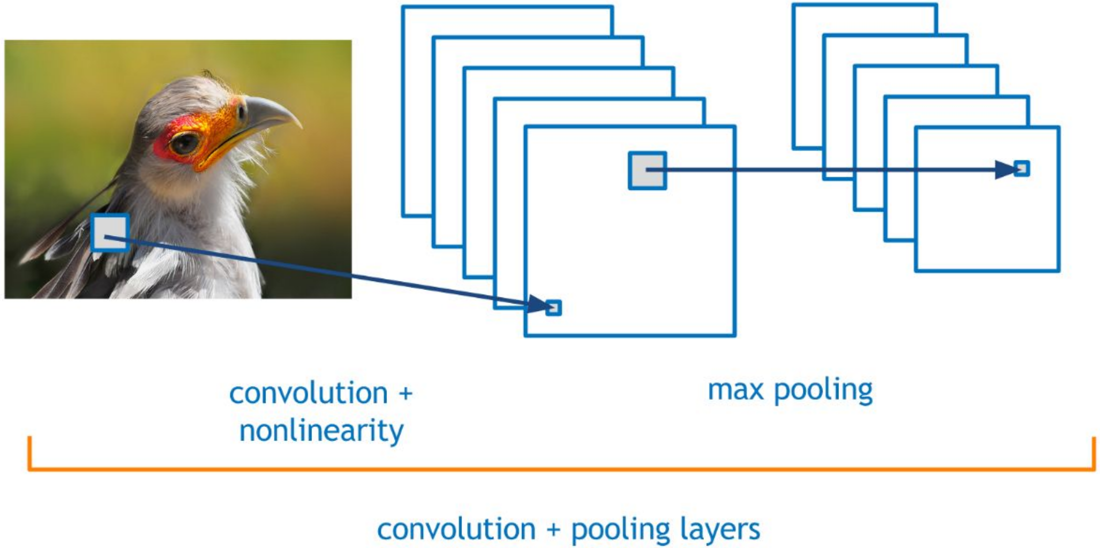
https://anhreynolds.com/blogs/cnn.html



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1


Input
Kernel
Output

Local Operators



wp.flickr.net

Operator Types

- Convolution: Feature detection - recognize some pattern over a small grid of inputs
  - At a given layer, have many different patterns that we are looking for in parallel
- Max Pooling: does there exist some pattern within a small grid of inputs?
- Scaling: Allows simple feature detection and pooling to apply at multiple visual scales

Local Operators

- Multiple stacked modules consisting of:
    - Pattern recognition (convolution),
    - Pooling (max)
    - Scaling (striding)
- With each module, our representation becomes more and more abstract
    - Ultimately: feathers, eyes, beaks …
    - All have specific visual patterns, though there may be many variations of each

Local Operators

Typical module:

- Reduce spatial dimensions by half
- Increase number of features by factor of 2
- In total: number of variables drops by a factor of 2

Module structure:

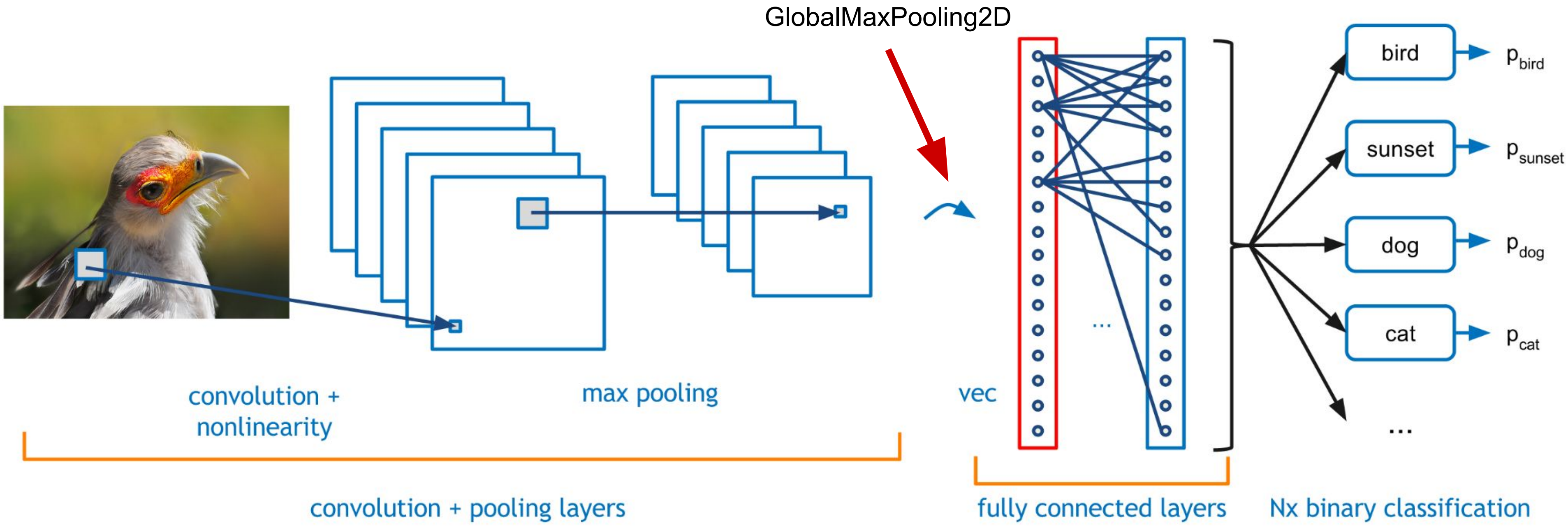
- (r,c,f) -C-> (r,c,f) -C-> … -C-> (r,c,f) -C-> (r,c,2f) -P->(r/2, c/2,2f)

Beyond the Primitives

How should the primitives be combined to form more of a semantic representation (dog, cat, grandma, etc.)?

- After computing the primitives in the first layers of our deep network, employ dense layers to allow for arbitrary combinations of the primitives

Combining Local Operators to Recognize Global Patterns



wp.flickr.net

Image Classification (Exclusive Classes)

Final layer:

- One output per class
- Nonlinearity: softmax
- Unusual nonlinearity:
  - Output i is a function of all of the other net inputs
  - Can interpret the output vector as a probability distribution (all elements are non-negative and they sum to 1)

CNN Details: Convolution

```
from keras.layers import Convolution2D
######

model = Sequential()

model.add(InputLayer(input_shape=(image_size[0],
                                        image_size[1],
                                        nchannels), name='input'))

# Input shape: (rows, cols, chans)

model.add(Convolution2D(filters=10,
                              kernel_size=3,               # Implies (3,3)
                              strides=1,
                              padding='valid',
                              use_bias=True,
                              name='C0',
                              activation='elu'))

# Output shape: (rows-2, cols-2, 10)
```

Convolution2D

Convolution2d other key properties:

- kernel_initializer
- bias_initializer
- kernel_regularizer
- bias_regularizer
- activity_regularizer

Pooling

```
from keras.layers import MaxPooling2D

######

# Input shape: (rows, cols, chans)

model.add(MaxPooling2D(pool_size=2,             # Implicit: (2,2)
                       strides=2,               # Also (2,2)
                       padding='same',
                       name='MP0'))

# Output shape: (rows//2, cols//2, chans)
```

Global Max Pooling

```
from keras.layers import GlobalMaxPooling2D

######

# Input shape: (rows, cols, chans)

model.add(GlobalMaxPooling2D())

# Output shape: (chans,)
```

Dropout

Drop entire channel at once

- (dropping single elements within a layer does not help)

```
from keras.layers import SpatialDropout2D

######

# Input shape: (rows, cols, chans)

model.add(SpatialDropout2D(p))

# Output shape: (rows, cols, chans)
```

CNN Notes

- 1D, 2D, and 3D versions built into Keras/TF
- Can use BatchNormalization() as usual
    - Applies individually to every element in the

      (rows, cols, chans) Tensor

CNN Modules

Sequence of layers:

- k x Conv2D
- MaxPooling2D
- SpatialDropout2D
- BatchNormalization

CNN for Image Classification

- n x CNN Module
  - Decreasing rows & cols while increasing filters (product should decrease)
- GlobalMaxPooling2D
- m x Dense
  - Decreasing number of hidden units
- Dense(nclasses, activation='softmax')
  - Classes are exclusive

Different N-Class Network Configs

All: N output units

	Exclusive Classes	**Multi-Class (any combination of classes)**
Nonlinearity	softmax	sigmoid
Desired output: binary encoding of class	**One-hot encoding** categorical_crossentropy categorical_accuracy	**Any binary vector** binary_crossentropy binary_accuracy
Desired output: 1 integer (class number)	sparse_categorical_crossentropy sparse_categorical_accuracy	**X**

Applications of CNNs

- Image classification
- Image recoding: deblurring, colorization, semantic segmentation
- Image generation

1D and 3D data are possible, too