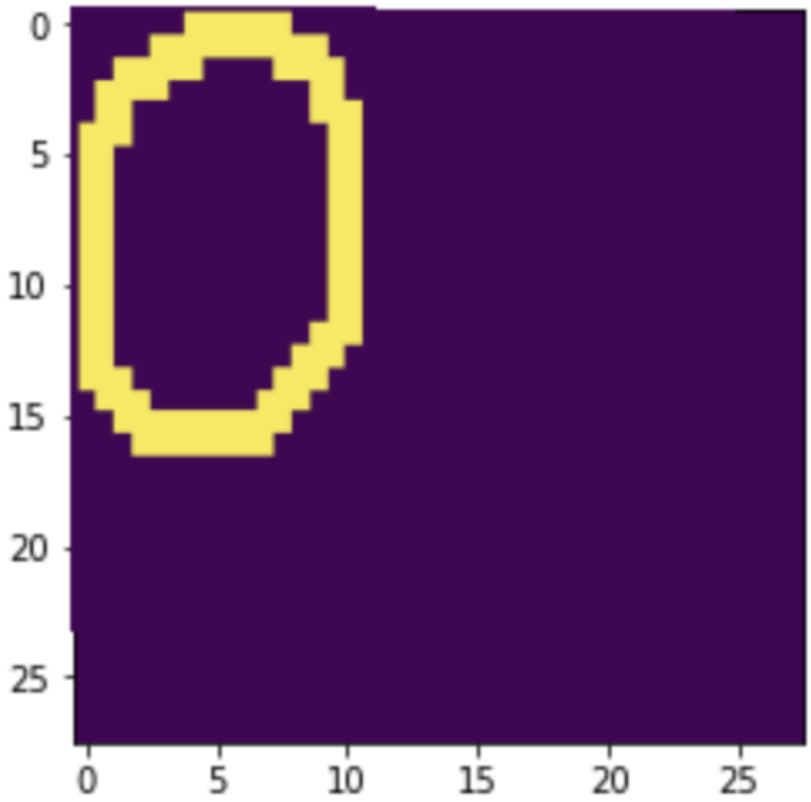
Mathematical techniques in data science

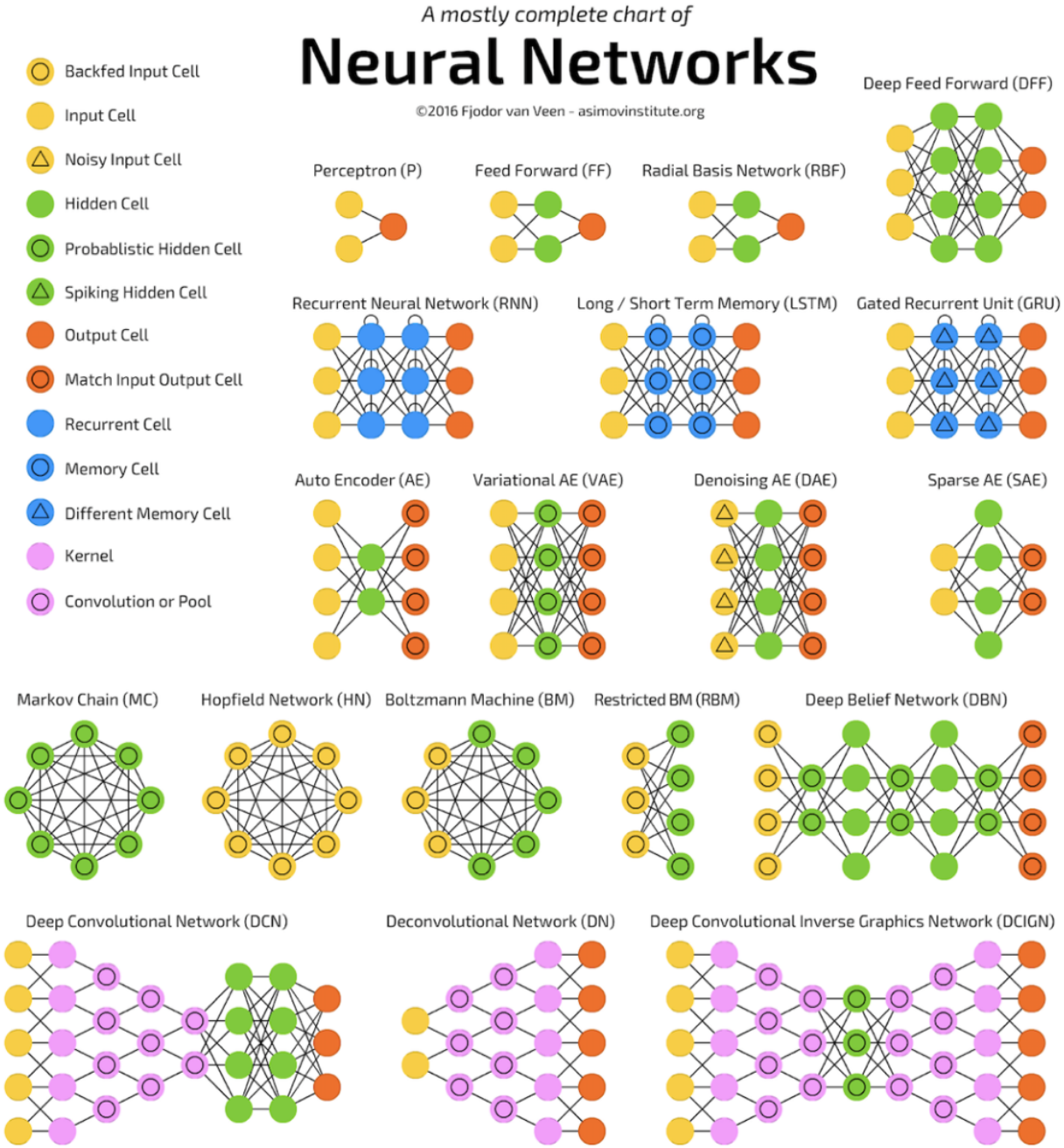
Lecture 7: Convolutional neural networks

Feed-forward neural networks

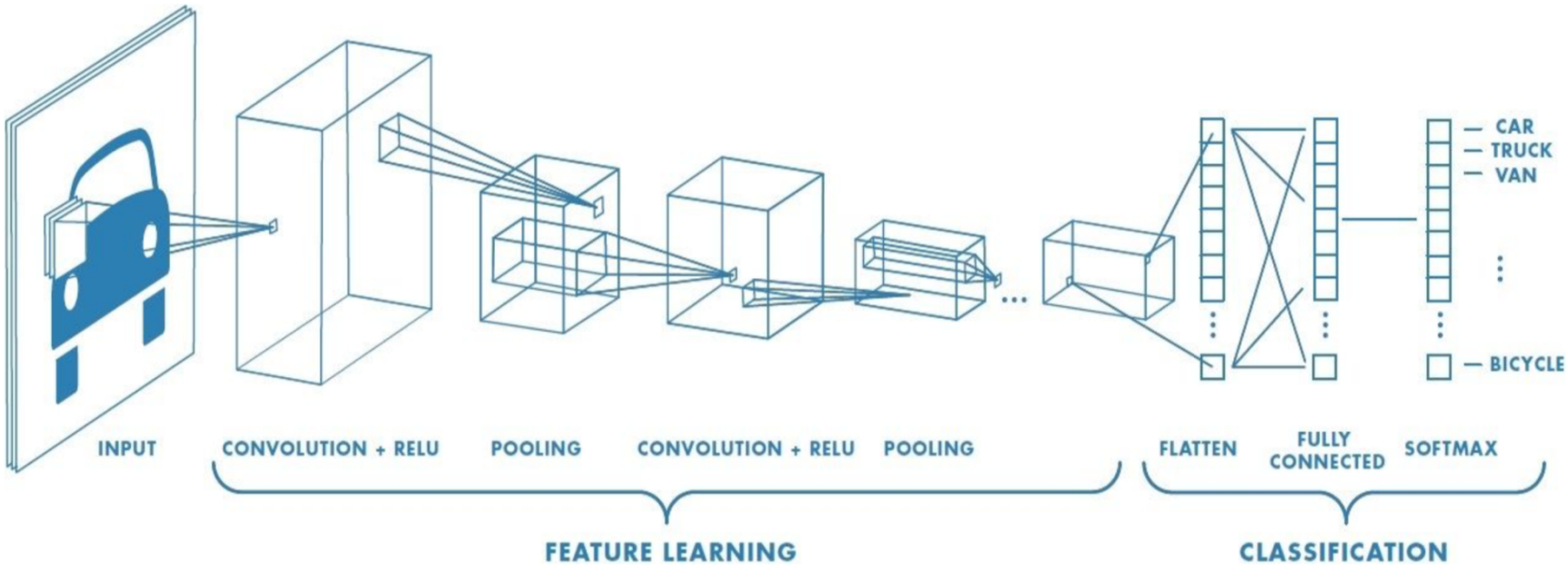
- Structure:
  - Graphical representation
  - Activation functions
- Issues (in computer vision applications)
  - Flat vectors lose spatial information
  - Sensitive to the location of the object
  - Cannot capture small regions within an image
  - Cannot capture relative differences
  - Redundant parameters

MNIST



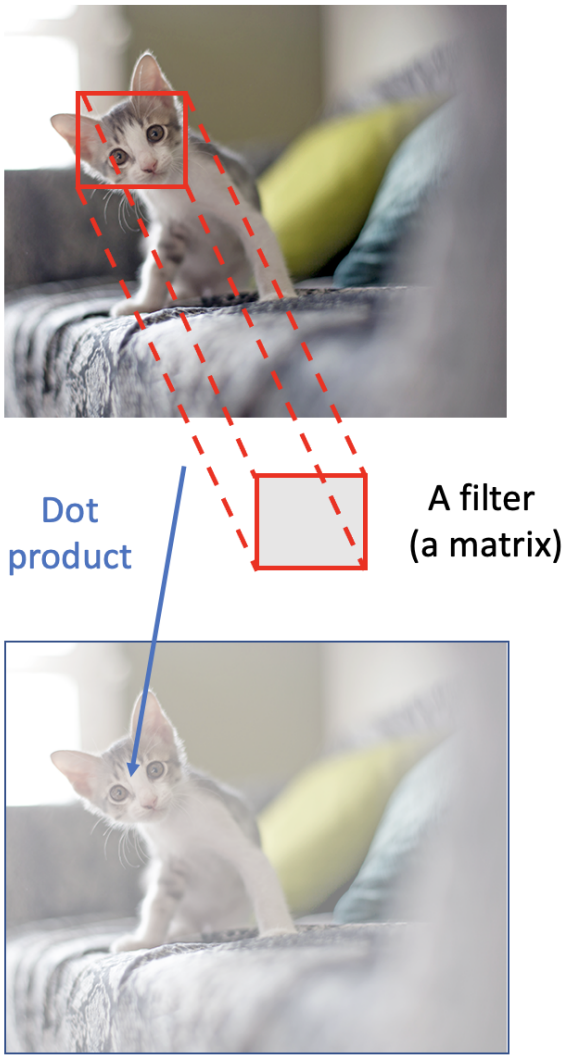

A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)

Convolutional neural networks



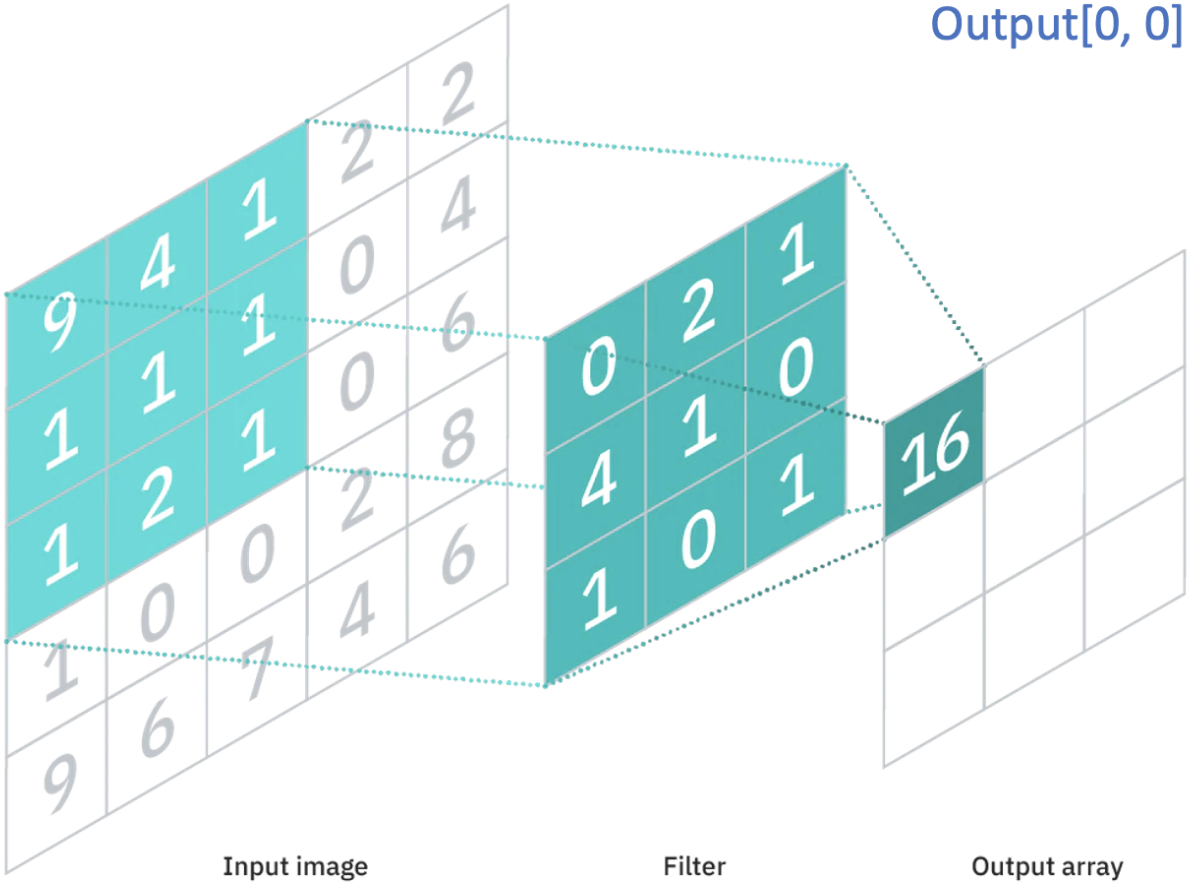
- pixel position and neighborhood have semantic meanings
- elements of interest can appear anywhere in the image

Convolutional layer



- Do not flatten the input image
- Apply a filter (kernel) to each local region of the image
- Slide the filter through all spatial locations to get the output

Applying a kernel



Output[0, 0] = (9*0) + (4*2) + (1*1) +
(1*4) + (1*1) + (1*0) +
(1*1) + (2*0) + (1*1)
= 16

(Adapted from IBM)

Examples of kernels

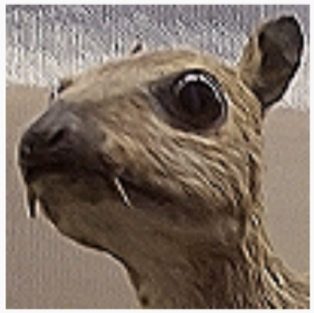
Edge detection

$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

Kernel

Sharpen

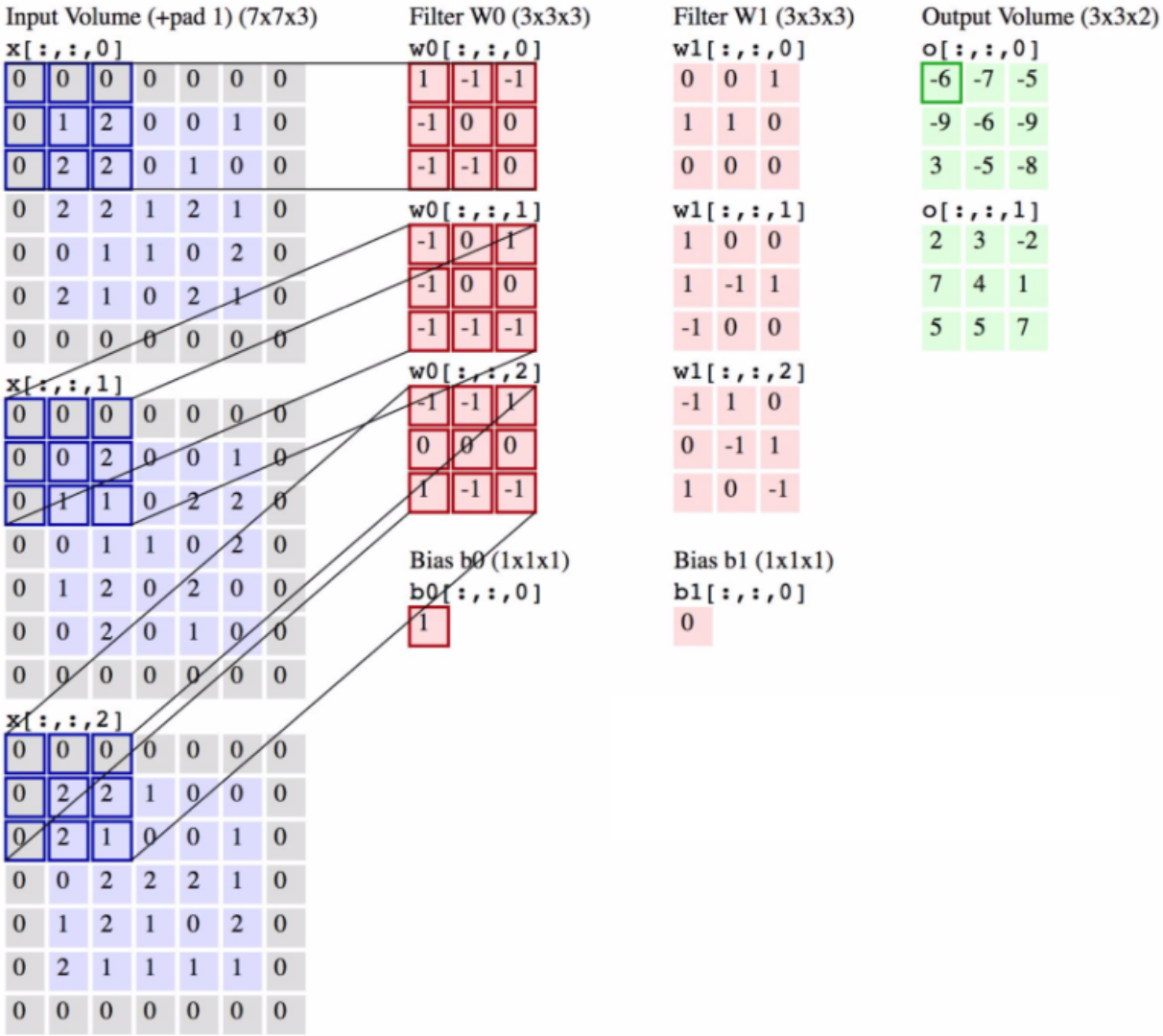
$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$



Feature extraction

- Historically, convolutional filters have been used to extract image features
- CNNs automate that process by considering the entries of the filters as model parameters
- This leads to a (somewhat technically) correct but misleading claim that "CNNs automatically design the features for image prediction"

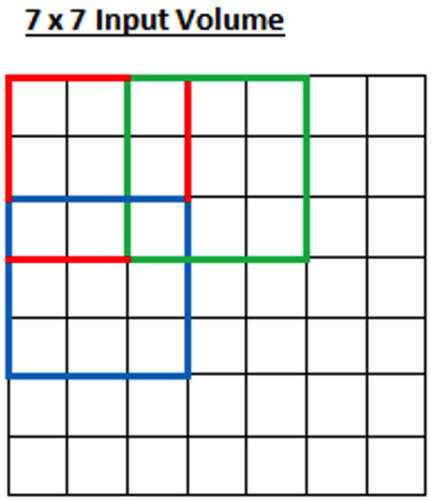
Feature extraction

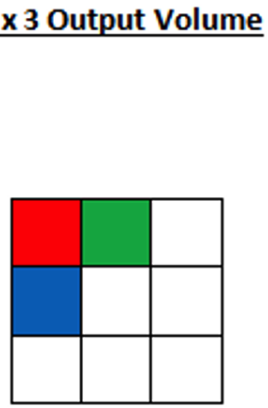


Stride

- Number of pixels to shift the filter
- Can be different for each dimension

strides = (2, 2)

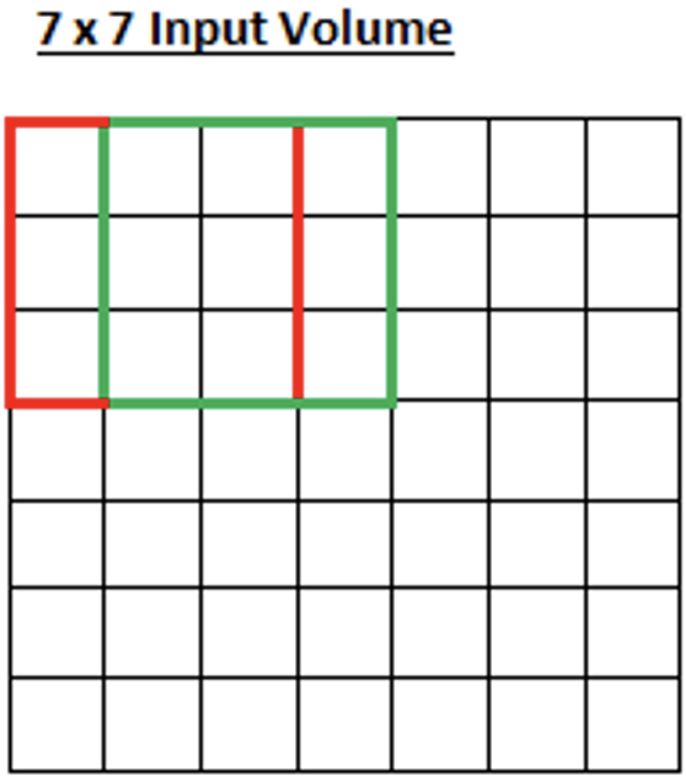


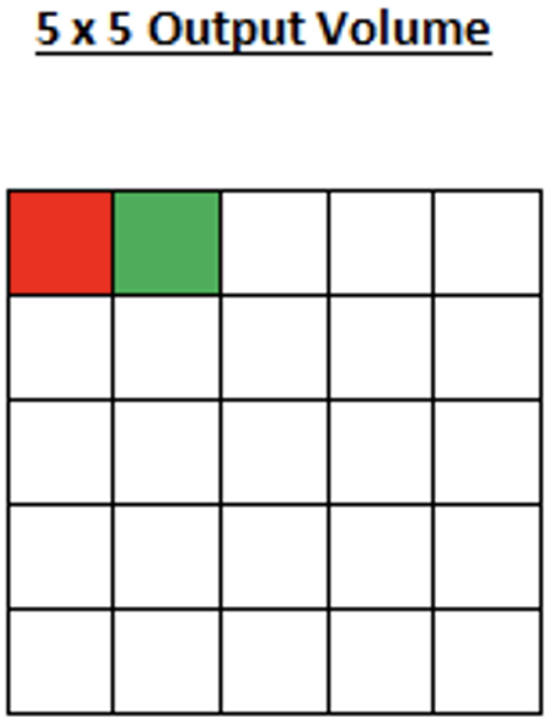


(Source: Adit Deshpande)

Padding

2 possible settings: Valid or Same

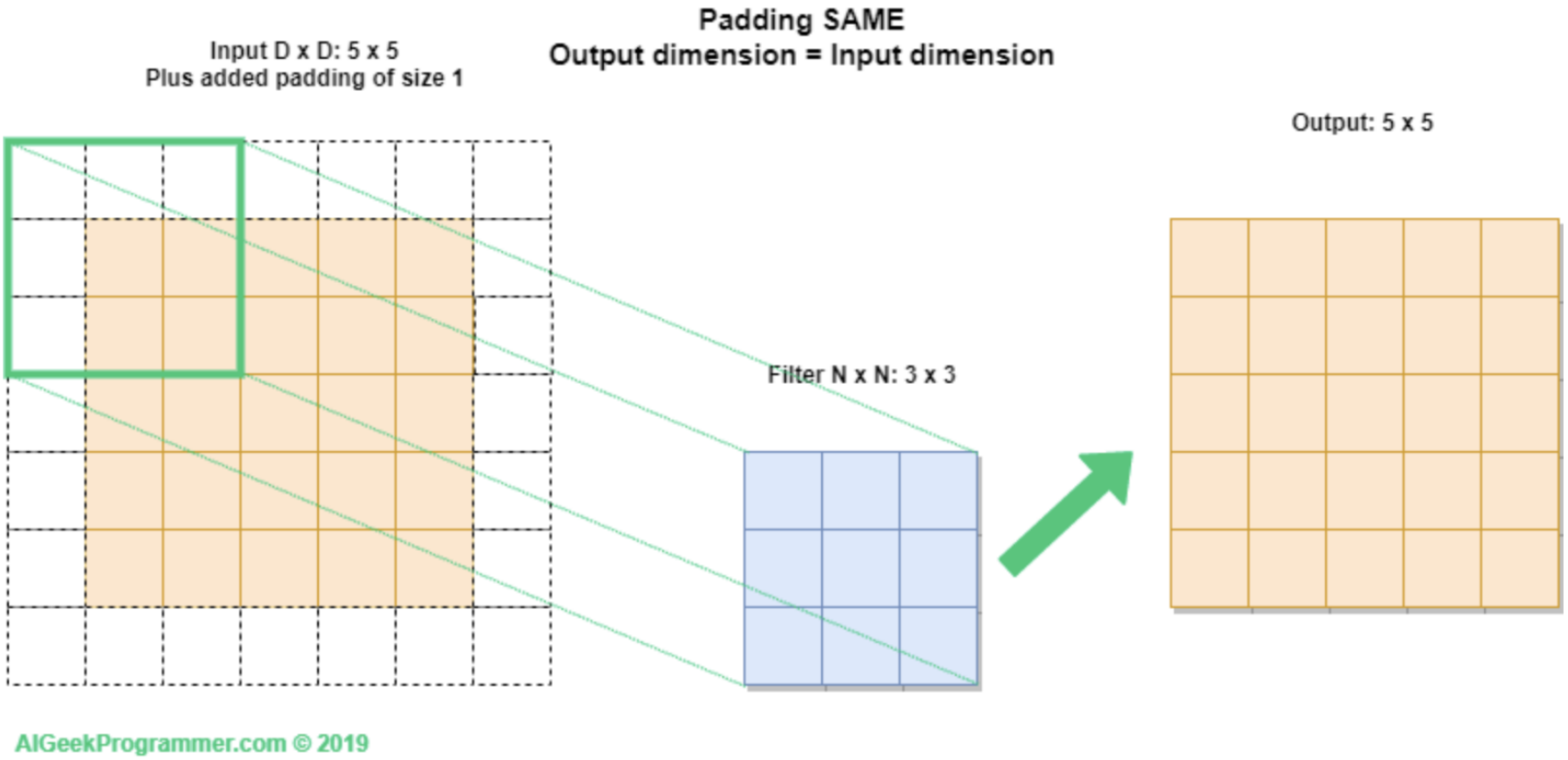




(Valid)

Padding

Same padding: add 0 pixels to the boundary of the input image to get a similar output shape



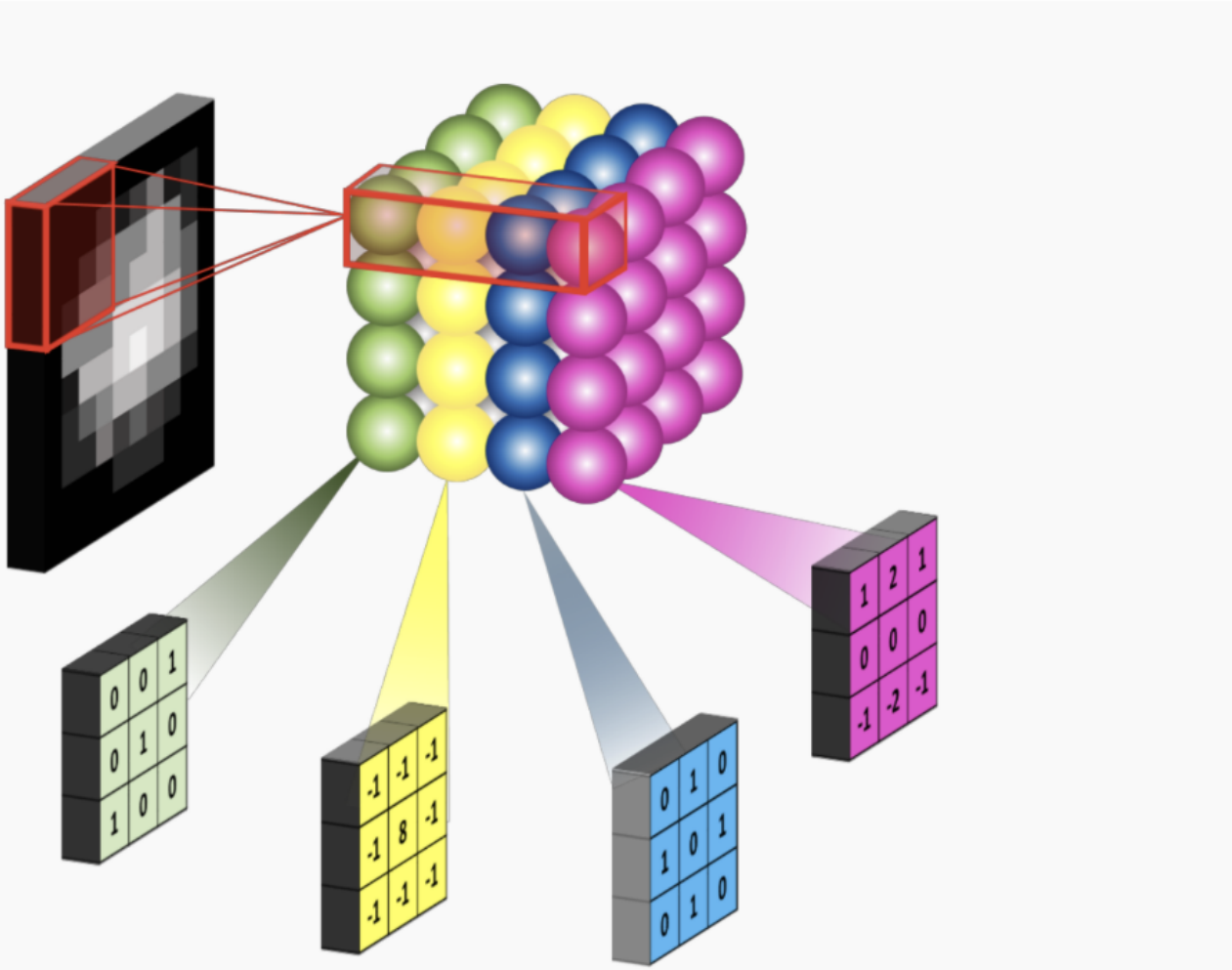
(Same)

Applying a kernel

https://i.stack.imgur.com/0rs9l.gif

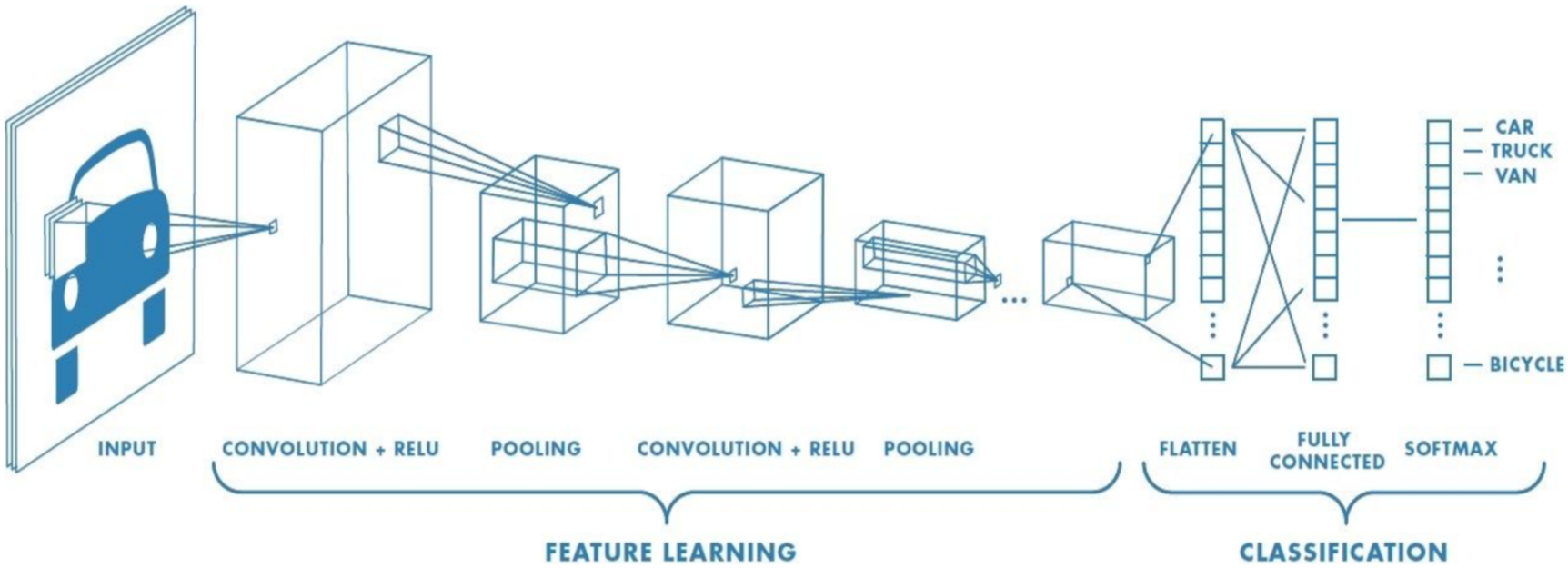
Visualization credits: vdumoulin@GitHub

Convolutional layer



Convolutional layer with four 3x3 filters on a black and white image (just one channel)

Convolutional neural networks

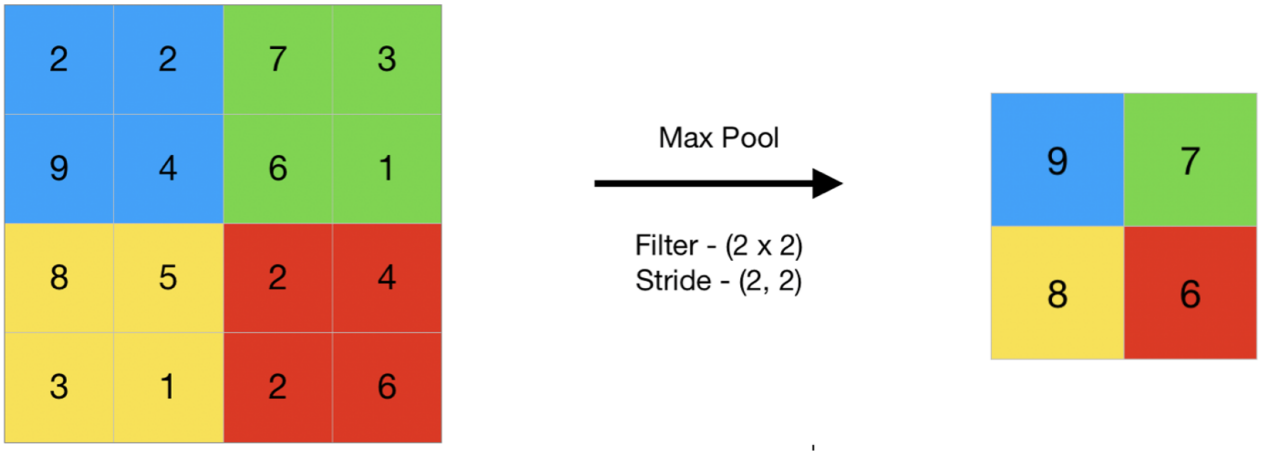


Pooling layer

- Down-sample the input image along its spatial dimensions
- Common types: max pooling and average pooling

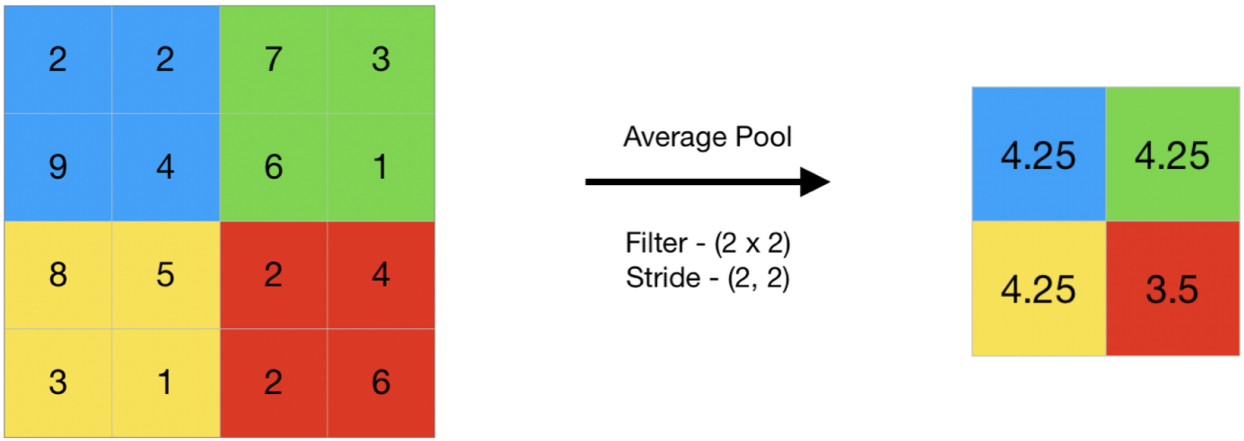
Max pooling

- Return max value when applying the filter
- Default strides = filter size

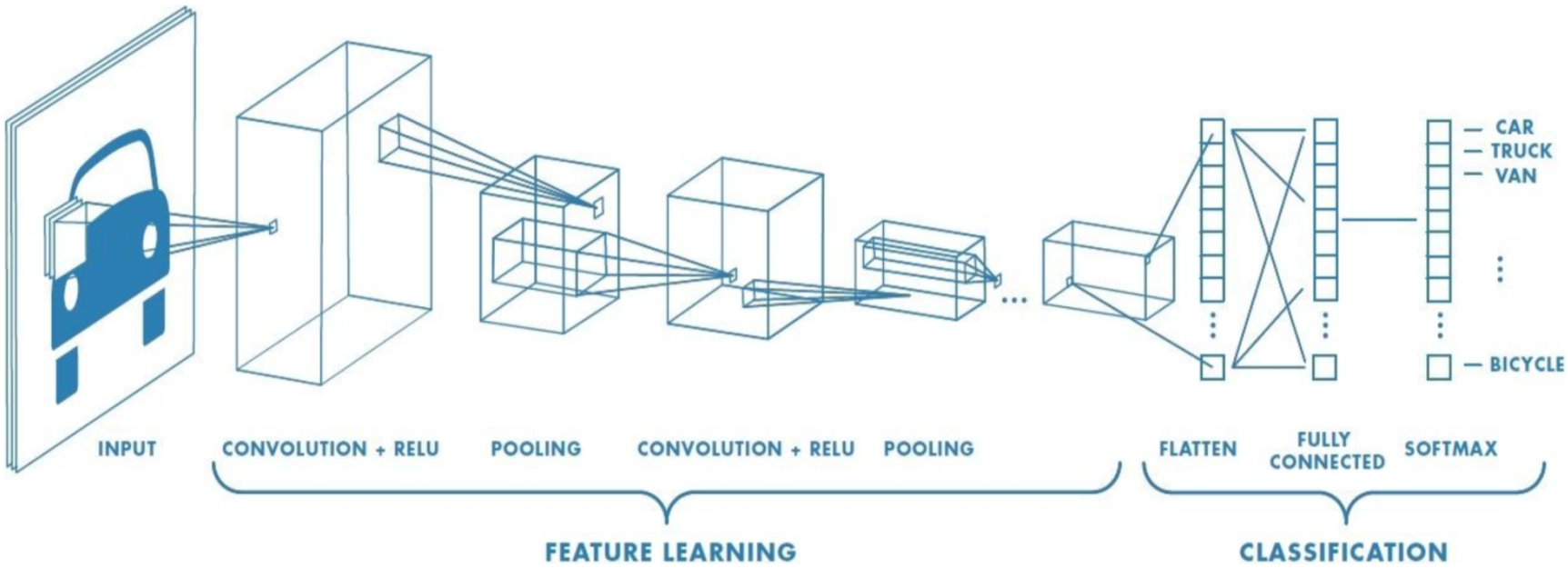


Average pooling

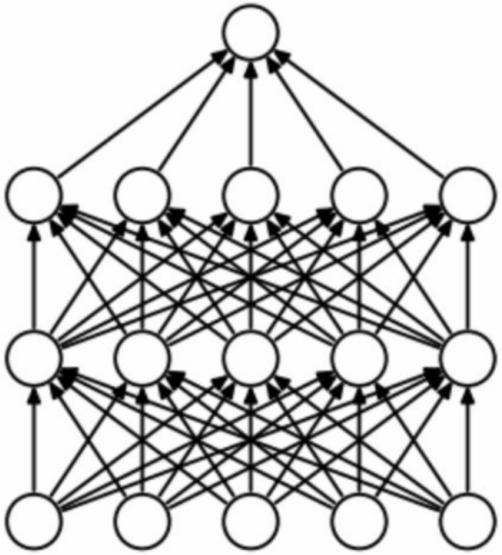
- Return average value when applying the filter
- Default strides = filter size



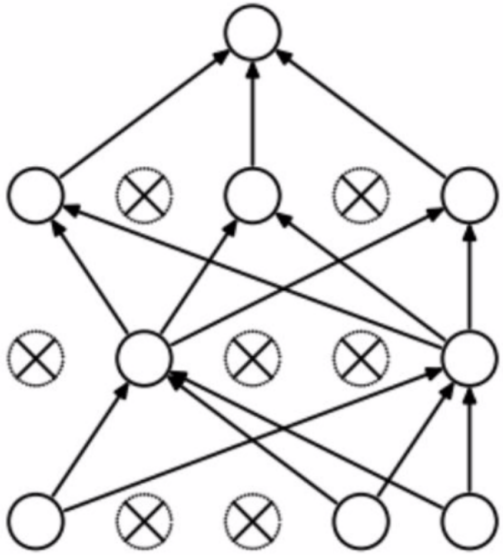
Convolutional neural networks



Other layer: Drop out

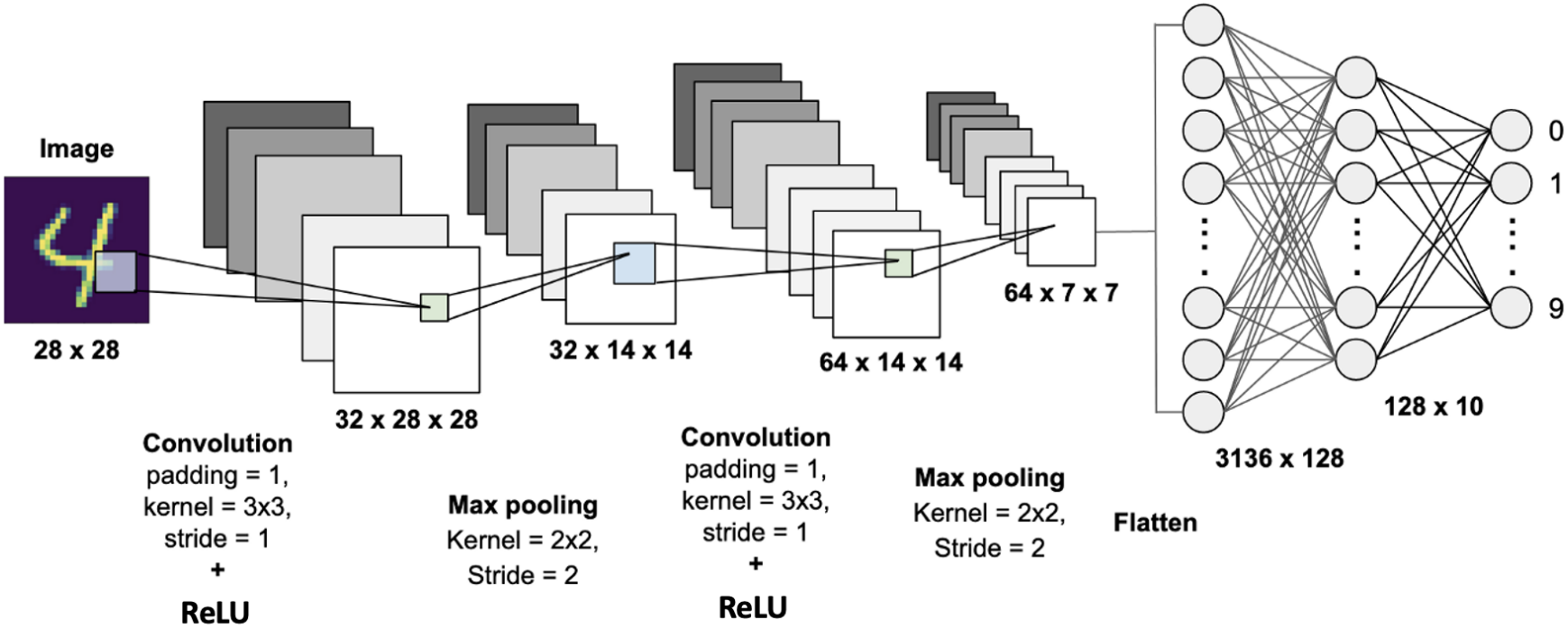


(a) Standard Neural Net

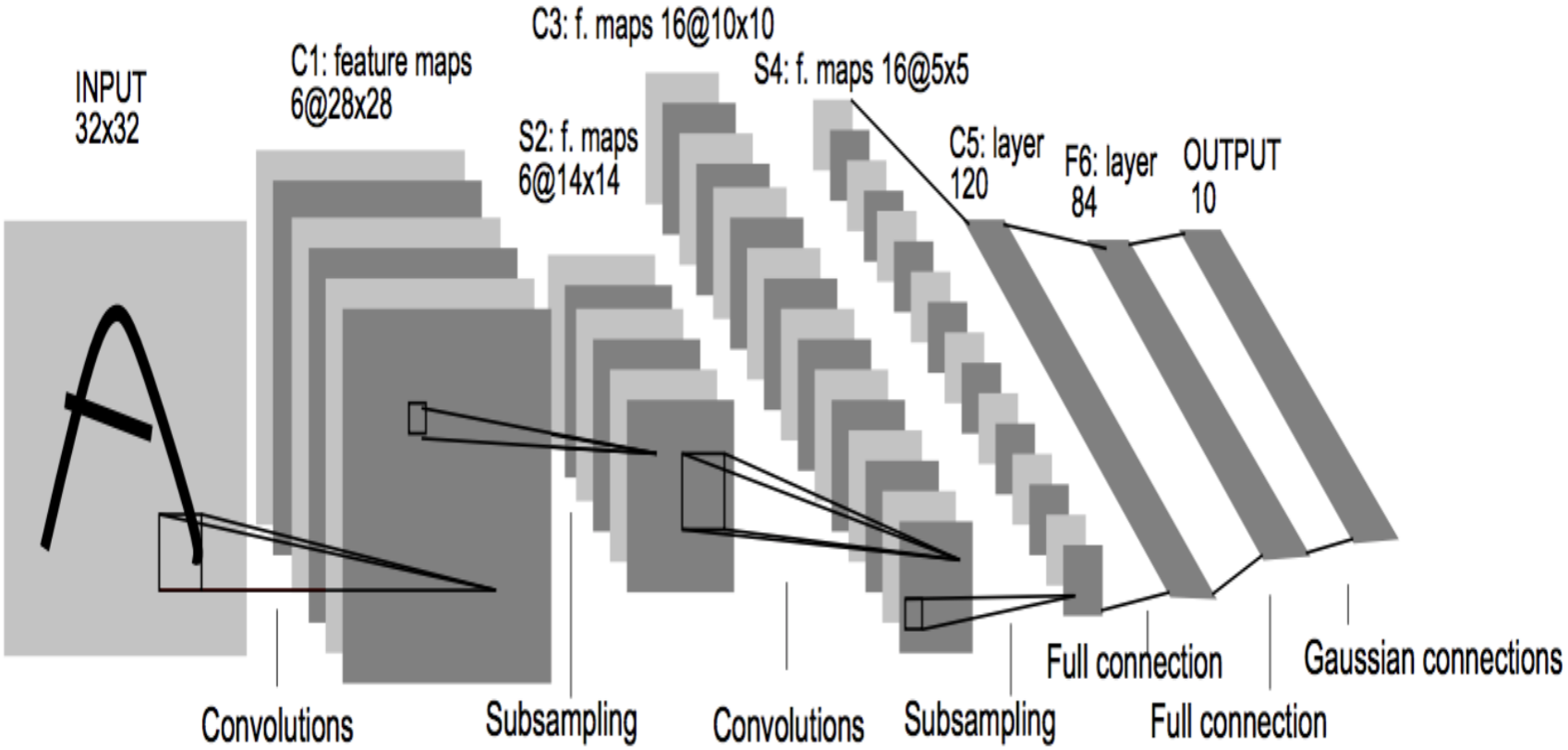


(b) After applying dropout.

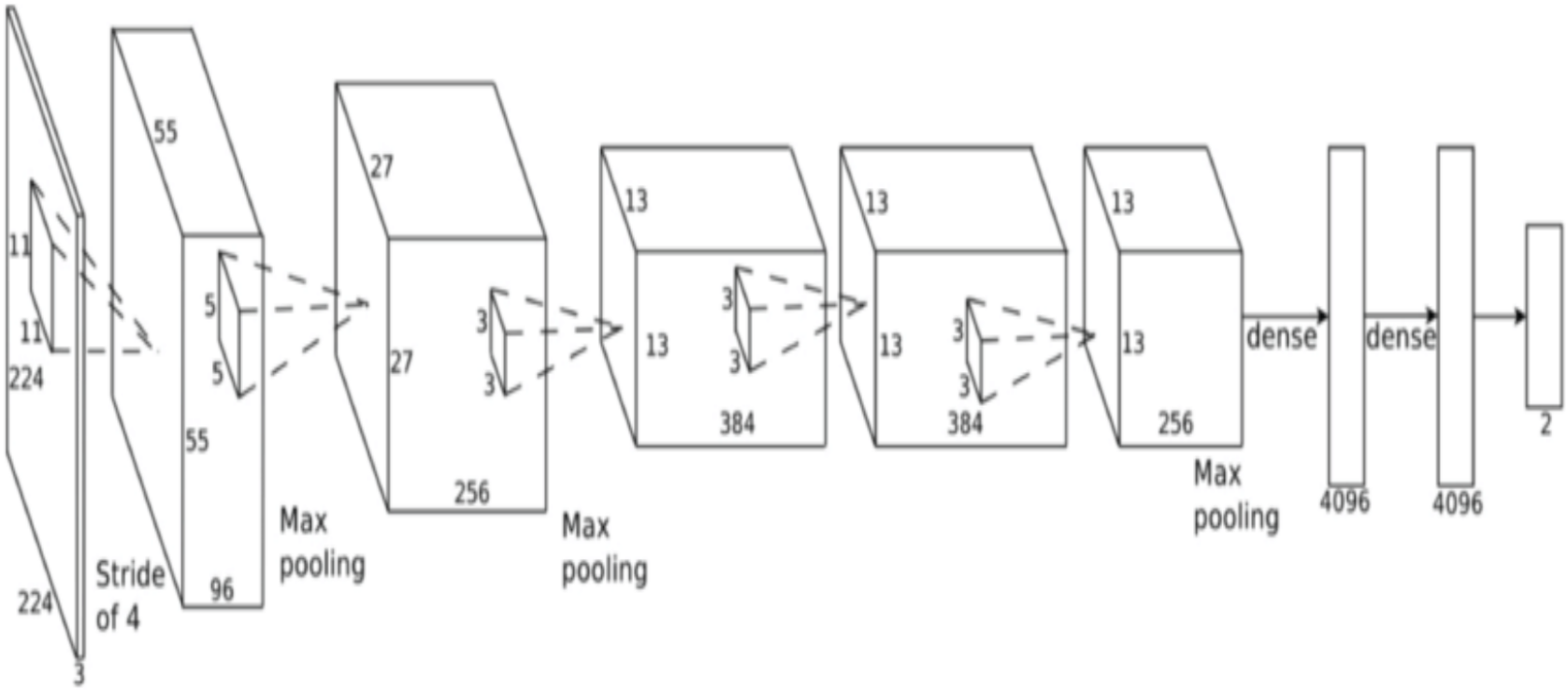
Example of a complete CNN



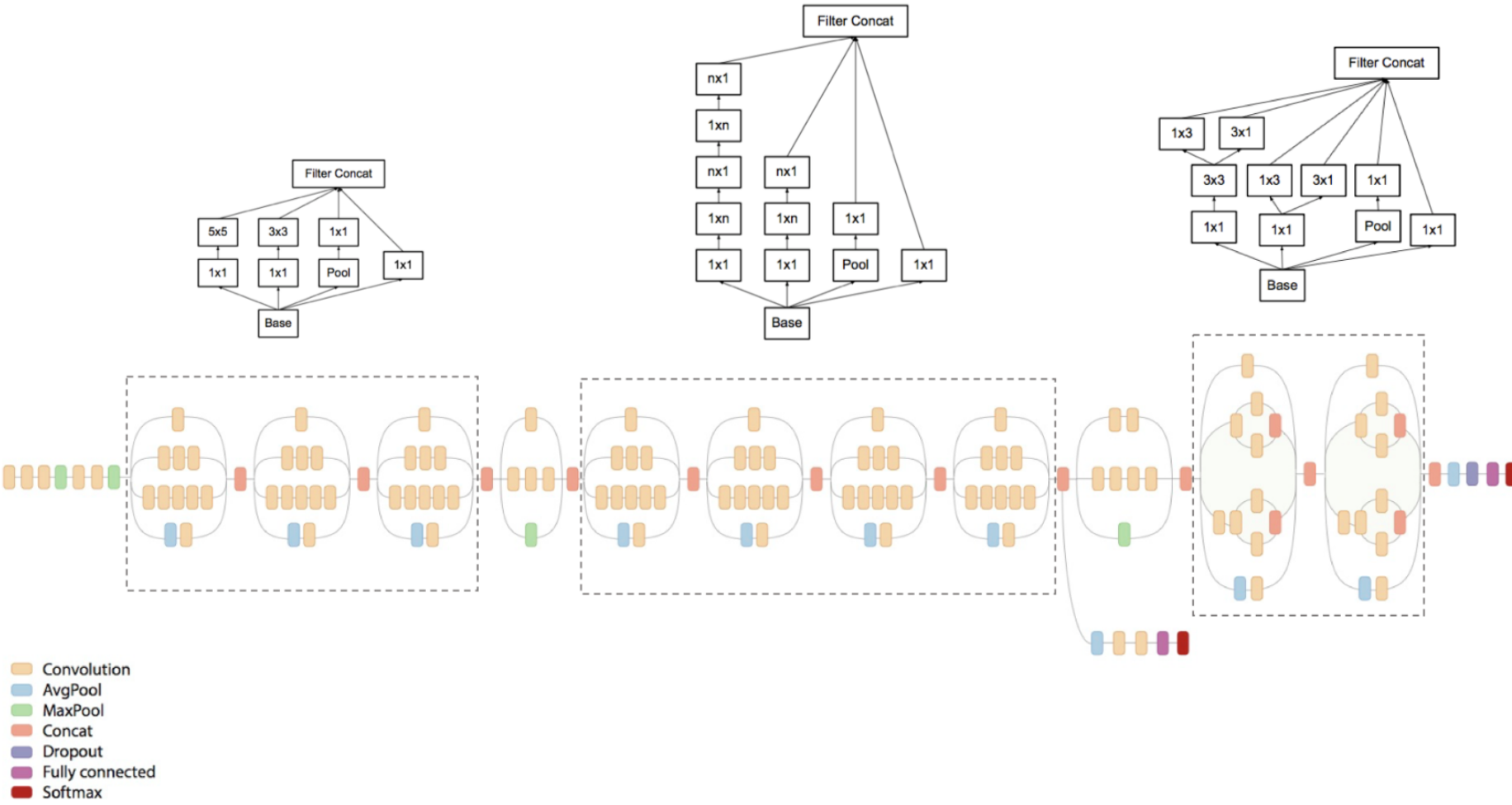
Yann Lecun's LeNet-5 (1998)



AlexNet (2012)



Google's Inception (2014)



Convolutional layer on Keras

Conv2D class

```
tf.keras.layers.Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding="valid",
    data_format=None,
    dilation_rate=(1, 1),
    groups=1,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

Pooling layer on Keras

`MaxPooling2D` class

```
tf.keras.layers.MaxPooling2D(
    pool_size=(2, 2), strides=None, padding="valid", data_format=None, **kwargs
)
```

`AveragePooling2D` class

```
tf.keras.layers.AveragePooling2D(
    pool_size=(2, 2), strides=None, padding="valid", data_format=None, **kwargs
)
```

CIFAR10 dataset

- Low-resolution color images of size 32 x 32
- 10 classes

Demo: train a CNN using Keras