Deep Learning

for Image Classification and Segmentation

Learning Objective

- First getting you acquainted with the basics of Tensorflow framework and ready to work.
- The contents focus on Deep learning theory and application together.
- You will learn things from the fundamentals and implement them throughout this course.
- Dive deeper into the word of Deep Learning with exciting and practical examples.

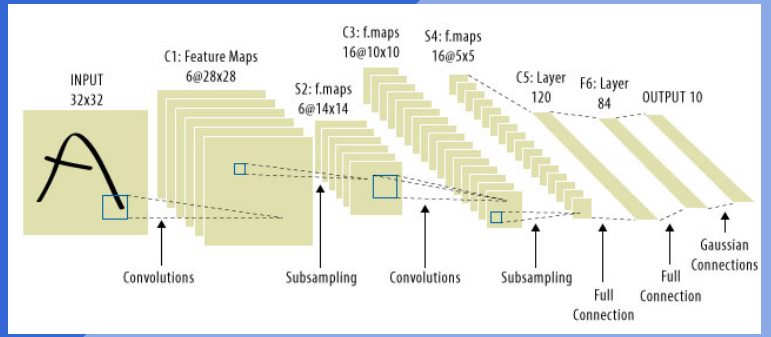
The study plan

- Unit 1: What is Tensorflow and how to solve linear regression problem in Python and Tensorflow?
- Unit 2: What is image classification and how to implement a solution?
- Unit 3: What is Neural Networks and how to apply on image classification?
- Unit 4: What is Convolutional Neural Networks and Deep Neural Networks?
- Unit 5: What is image segmentation and how to implement a solution?

Unit 6: How to improve Deep Neural Networks?

Short-Term Project Report Presentation

The study plan



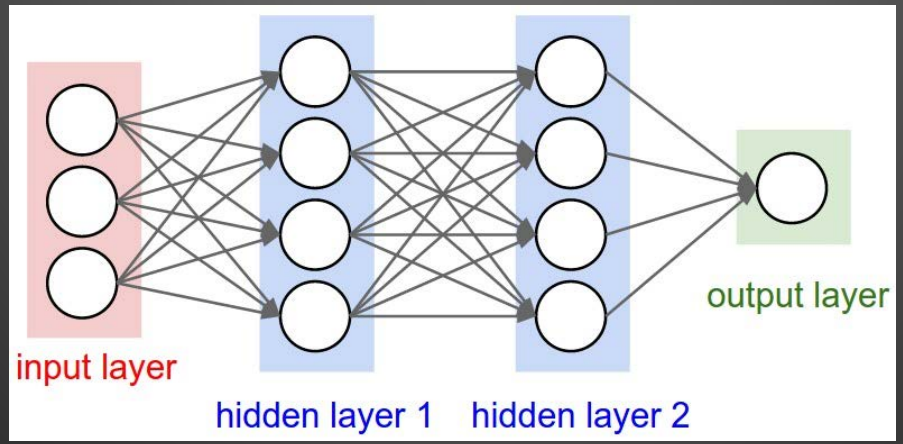
- Unit 4: What is Convolutional Neural Networks and Deep Neural Networks?
- Convolutional Neural Networks (CNNs / ConvNets)
- http://cs231n.github.io/convolutional-networks/

CS231n: Convolutional Neural Networks for Visual Recognition

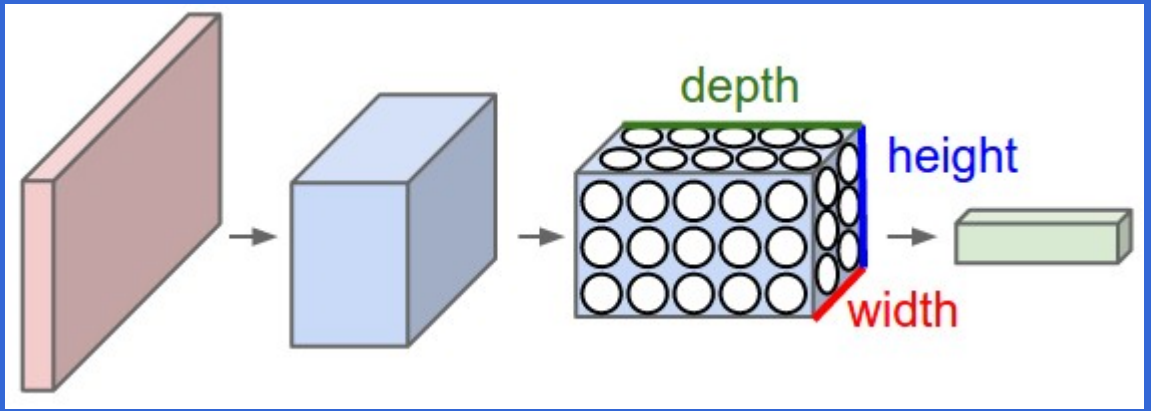
- Table of Contents:
- Architecture Overview
- ConvNet Layers
- Convolutional Layer
- Pooling Layer
- Normalization Layer
- Fully-Connected Layer
- Converting Fully-Connected Layers to Convolutional Layers
- http://cs231n.github.io/convolutional-networks/

CS231n: Convolutional Neural Networks for Visual Recognition

- A regular 3-layer Neural Network.

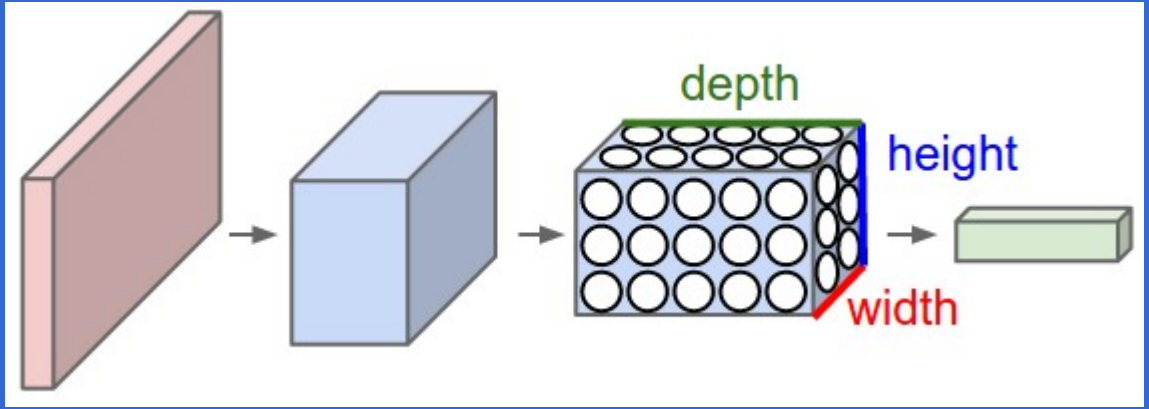


ConvNet



- A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations.
- In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels)

ConvNet



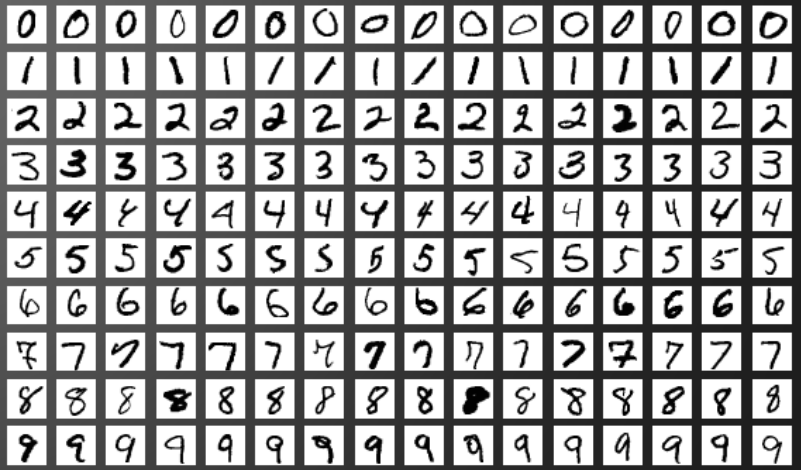
- Layers used to build ConvNets
- As we described above, a simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function.
- We use three main types of layers to build ConvNet architectures:
- Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet architecture.

ConvNet

- CIFAR-10 DATA SET
- https://en.wikipedia.org/wiki/MNIST_database

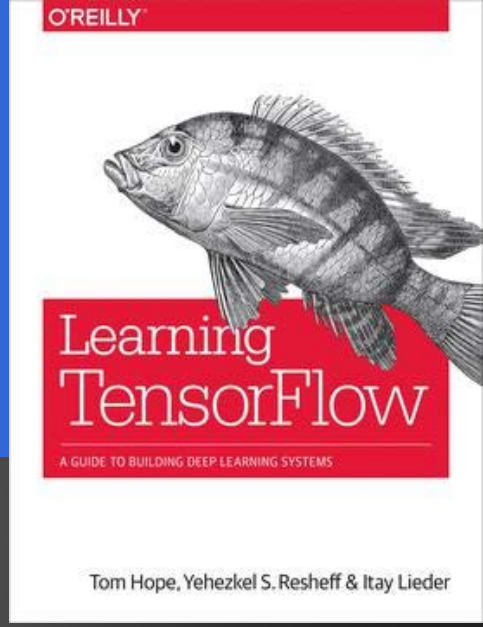
This is a table of some of the machine learning methods used on the database and their error rates, by type of classifier:

Type	Classifier	Distortion	Preprocessing	Error rate (%)
Linear classifier	Pairwise linear classifier	None	Deskewing	7.6[9]
K-Nearest Neighbors	K-NN with non-linear deformation (P2DHMDM)	None	Shiftable edges	0.52[18]
Boosted Stumps	Product of stumps on Haar features	None	Haar features	0.87[19]
Non-linear classifier	40 PCA + quadratic classifier	None	None	3.3[9]
Support vector machine	Virtual SVM, deg-9 poly, 2-pixel jittered	None	Deskewing	0.56[20]
Neural network	2-layer 784-800-10	None	None	1.6[21]
Neural network	2-layer 784-800-10	elastic distortions	None	0.7[21]
Deep neural network	6-layer 784-2500-2000-1500-1000-500-10	elastic distortions	None	0.35[22]
Convolutional neural network	6-layer 784-40-80-500-1000-2000-10	None	Expansion of the training data	0.31[15]
Convolutional neural network	8-layer 784-50-100-500-1000-10-10	None	Expansion of the training data	0.27[16]
Convolutional neural network	Committee of 35 CNNs, 1-20-P-40-P-150-10	elastic distortions	Width normalizations	0.23[8]
Convolutional neural network	Committee of 5 CNNs, 6-layer 784-50-100-500-1000-10-10	None	Expansion of the training data	0.21[17]



O’Reilly learning Tensorflow github

- https://github.com/Hezi-Resheff/Oreilly-Learning-TensorFlow
- https://www.safaribooksonline.com/library/view/learning-tensorflow/9781491978504/

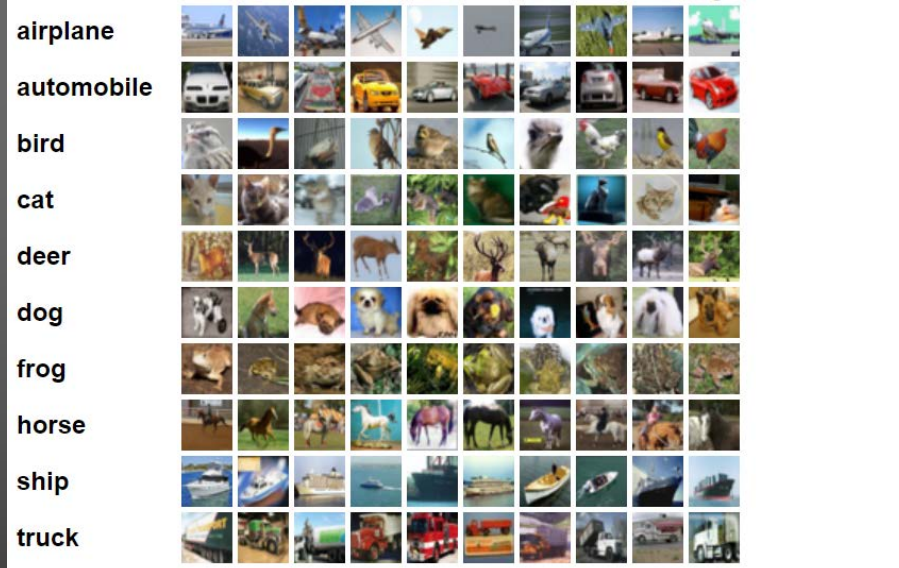


ConvNet

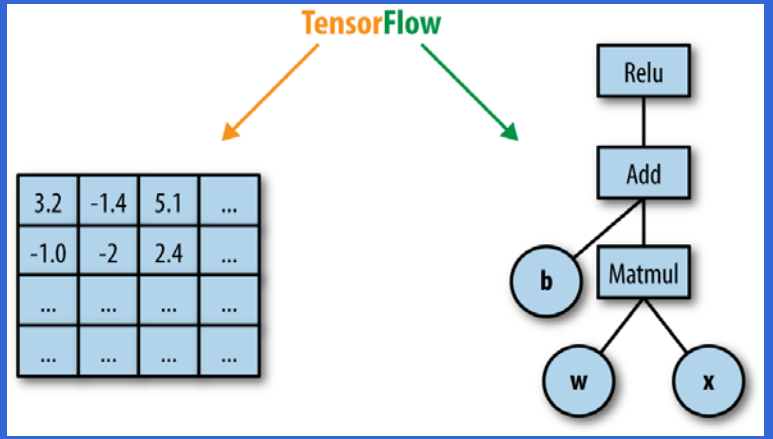
- CIFAR-10 DATA SET
- https://en.wikipedia.org/wiki/CIFAR-10

Research Paper	Error rate (%)	Published Date
Convolutional Deep Belief Networks on CIFAR-10[6]	21.1	August, 2010
Densely Connected Convolutional Networks[7]	5.19	August 24, 2016
Wide Residual Networks[8]	4.0	May 23, 2016
Neural Architecture Search with Reinforcement Learning[9]	3.65	November 4, 2016
Fractional Max-Pooling[10]	3.47	December 18, 2014
Shake-Shake regularization[11]	2.86	May 21, 2017
Improved Regularization of Convolutional Neural Networks with Cutout[12]	2.56	Aug 15, 2017
ShakeDrop regularization[13]	2.31	Feb 7, 2018
Regularized Evolution for Image Classifier Architecture Search[14]	2.13	Feb 6, 2018

Here are the classes in the dataset, as well as 10 random images from each:

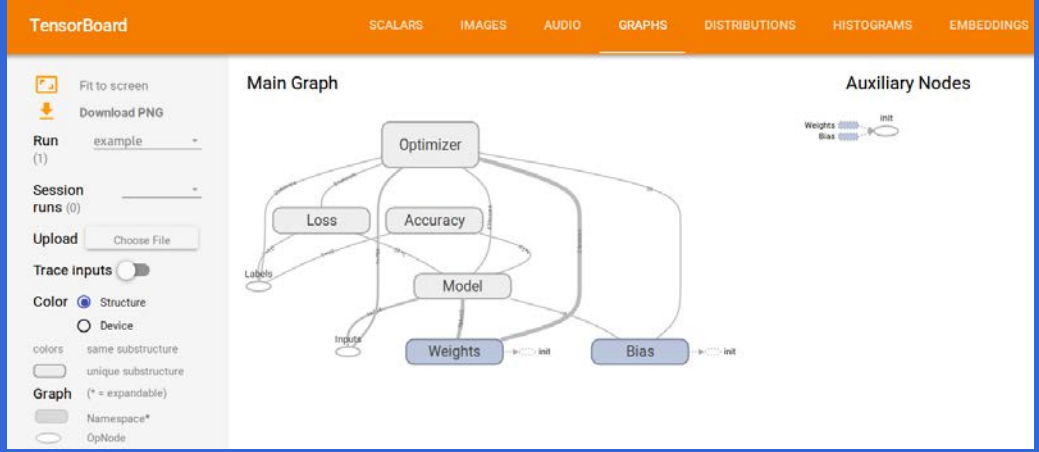


ConvNet



- A dataflow computation graph. Data in the form of tensors flows through a graph of computational operations that make up our deep neural networks.

ConvNet

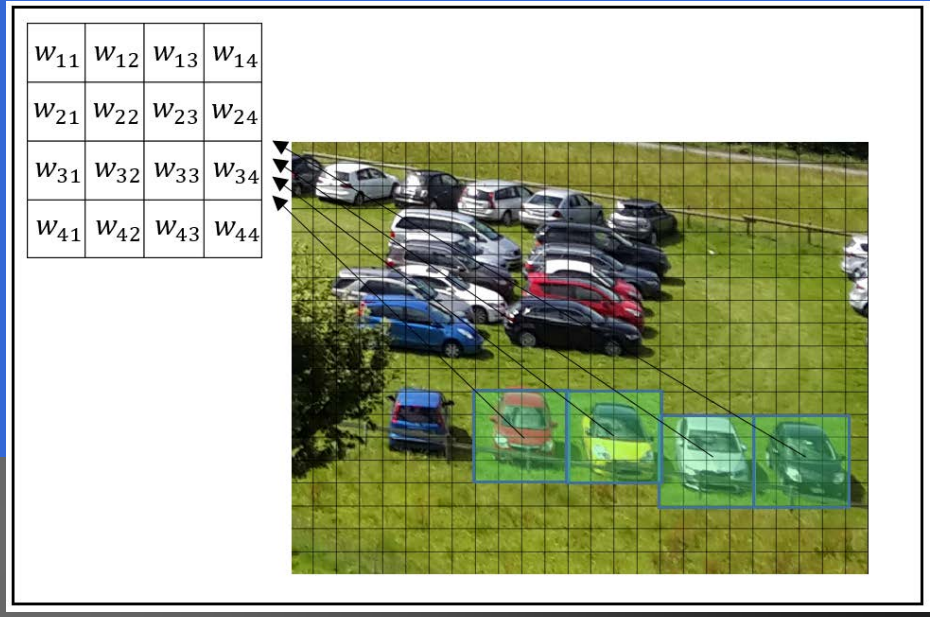


- TensorFlow’s visualization tool, TensorBoard, for monitoring, debugging, and analyzing the training process and experiments.

ConvNet

- Example Architecture: Overview. We will go into more details below, but a simple ConvNet for CIFAR-10 classification could have the architecture
- [INPUT - CONV - RELU - POOL - FC].
- In more detail:

ConvNet



- In more detail:

- INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.

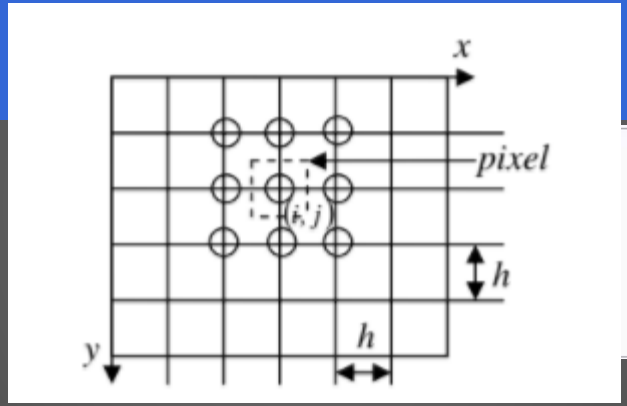
Convolution

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] = (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + (c \cdot 7) + (b \cdot 8) + (a \cdot 9).$$

- Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel. This is related to a form of mathematical convolution. It should be noted that the matrix operation being performed - convolution - is not traditional matrix multiplication, despite being similarly denoted by *. https://en.wikipedia.org/wiki/Kernel_(image_processing)
- For example, if we have two three-by-three matrices, the first a kernel, and the second an image piece, convolution is the process of flipping both the rows and columns of the kernel and then multiplying locally similar entries and summing. The element at coordinates [2, 2] (that is, the central element) of the resulting image would be a weighted combination of all the entries of the image matrix, with weights given by the kernel:

Convolution

- Blur



$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Convolution

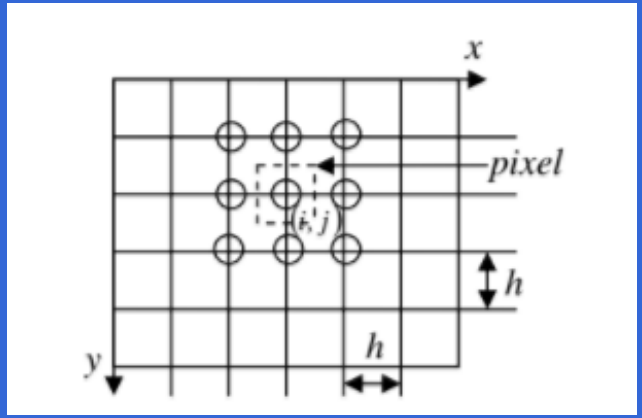
- Edge detection

$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Edge detection

- first-order methods
- Different gradient operators can be applied to estimate image gradients from the input image or a smoothed version of it.
- The simplest approach is to use central differences:

Edge detection



- The derivative value is a slope.
- Numerical Approximation.
- The simplest approach is to use central differences:

$$L_x(x,y) = -\frac{1}{2}L(x-1,y) + 0 \cdot L(x,y) + \frac{1}{2} \cdot L(x+1,y)$$

$$L_y(x,y) = -\frac{1}{2}L(x,y-1) + 0 \cdot L(x,y) + \frac{1}{2} \cdot L(x,y+1),$$

Edge detection

- corresponding to the application of the following filter masks to the image data:

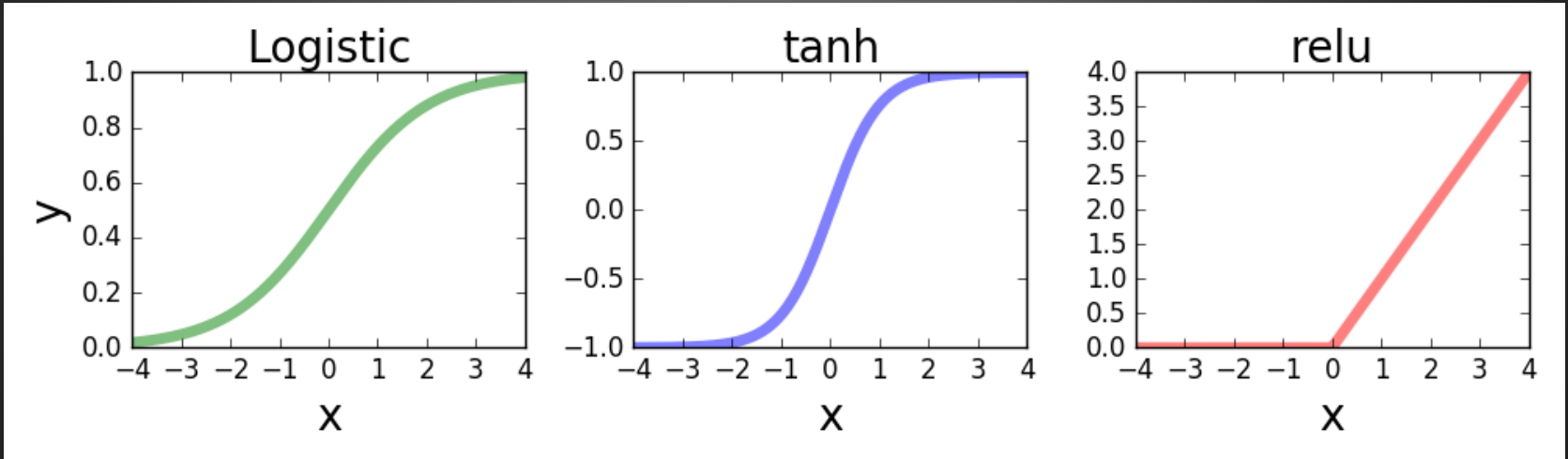
$$L_x = \begin{bmatrix} -1/2 & 0 & 1/2 \end{bmatrix} L \quad \text{and} \quad L_y = \begin{bmatrix} +1/2 \\ 0 \\ -1/2 \end{bmatrix} L.$$

Edge detection

- The well-known and earlier Sobel operator is based on the following filters:

$$L_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} L \quad \text{and} \quad L_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} L.$$

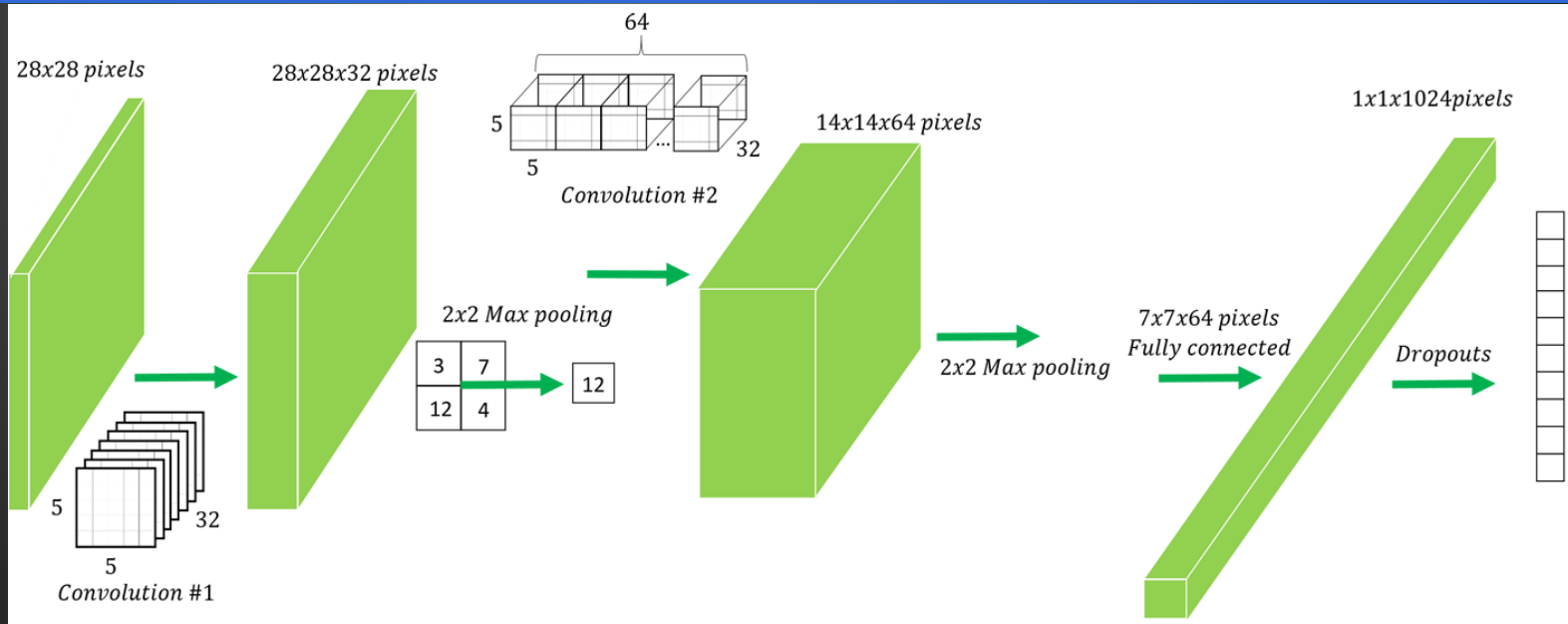
activation function



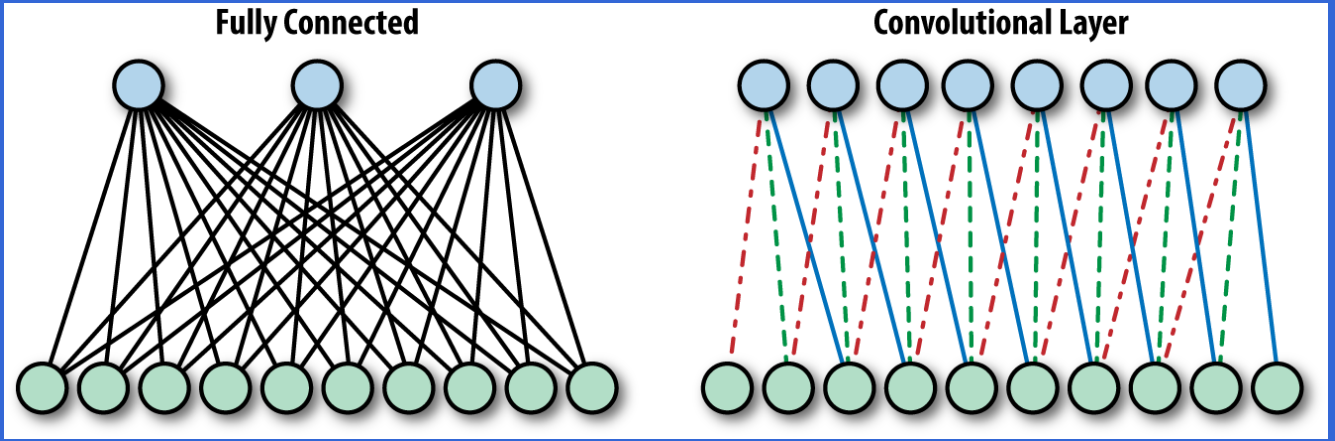
- RELU

ConvNet

- Mnist Data Set



ConvNet



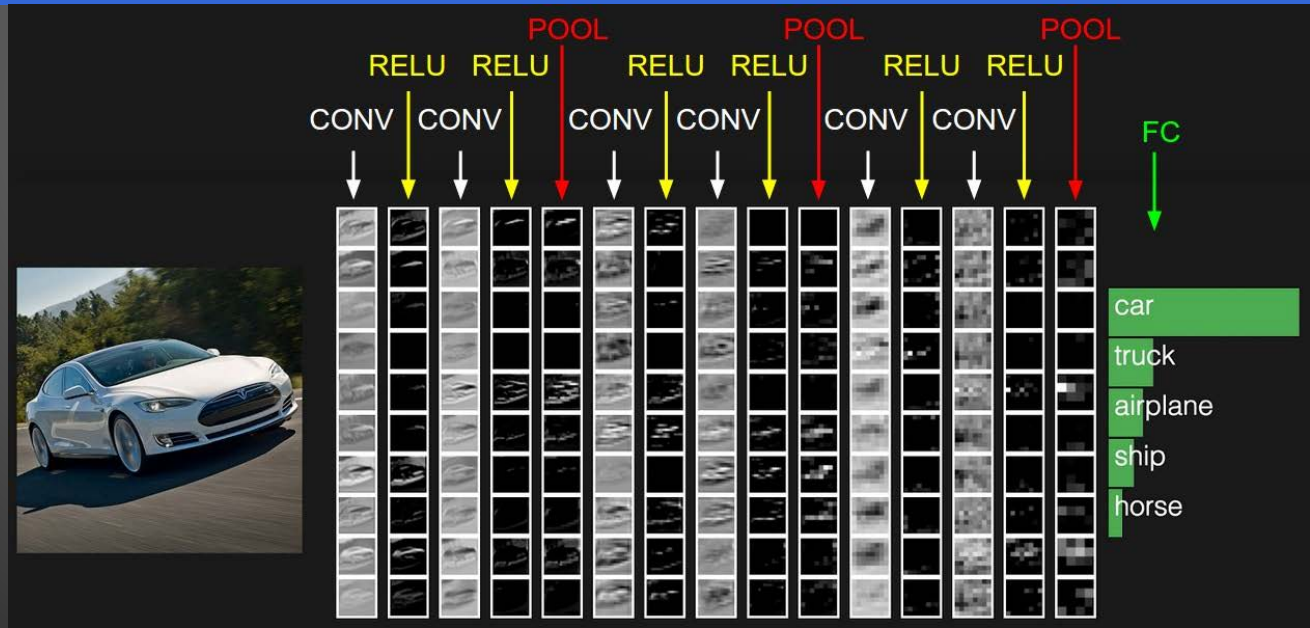
- RELU layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

ConvNet:In summary:

- A ConvNet architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores)
- There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular)
- Each Layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function
- Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don’t)
- Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn’t)

ConvNet:In summary:

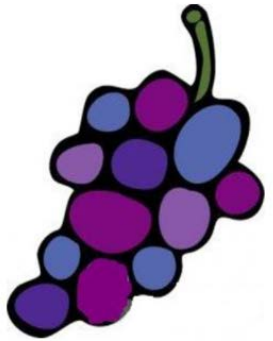
- A ConvNet architecture



Let the course make you feel like eating Grapes!

GRAPE Retention

- **G**rowth
- **R**ecognition
- **A**chievement
- **P**articipation
- **E**njoyment



Make you Growth. Make you Recognition.
Make you Achievement. Make you Participation.
Make you Enjoyment.