Machine Learning TensorFlow

Bolous AbuJaber

- Software Engineer
- Robotics
- Machine learning (Computer vision)

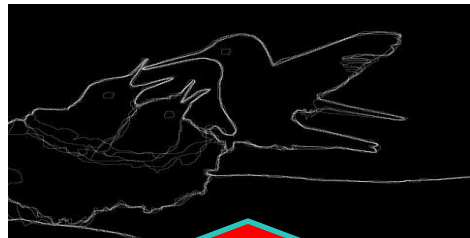# What do we want ML to do?

- Given image, predict complex high-level patterns:
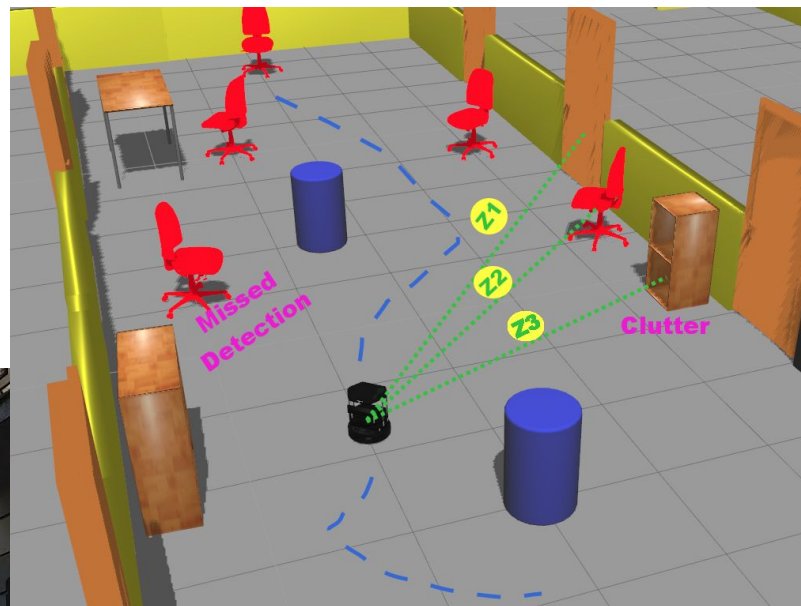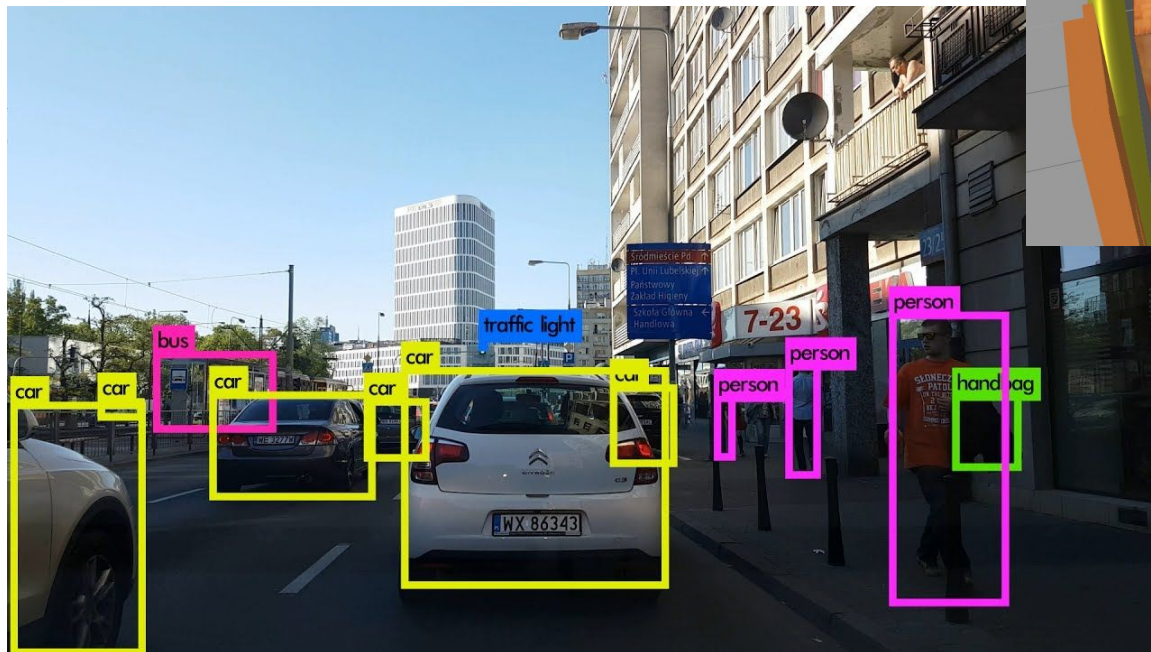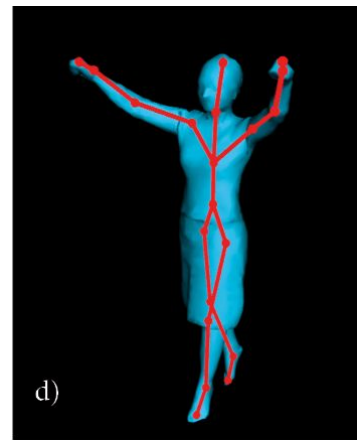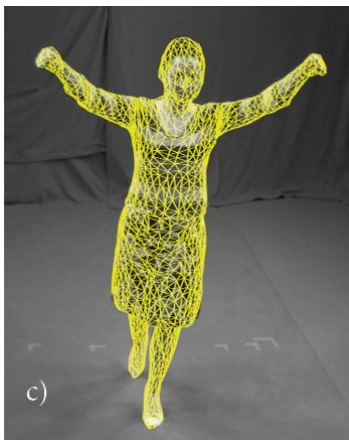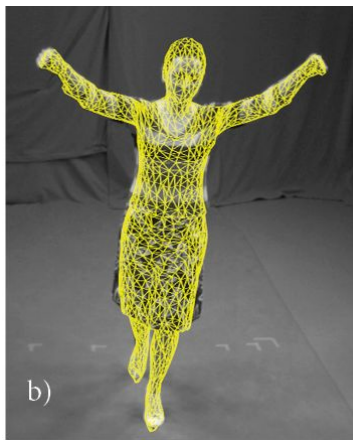
"Cat"

Object recognition

Detection

Segmentation
[Martin et al., 2001]

Can machine really see ?

More complex abillities

Why Deep learning

Why deep learning

How do data science techniques scale with amount of data?
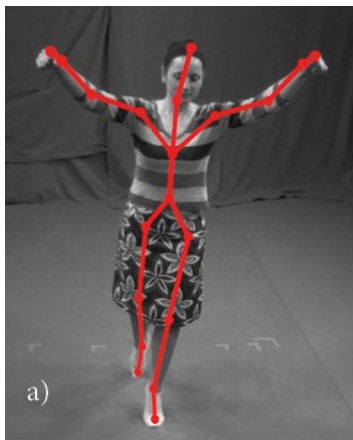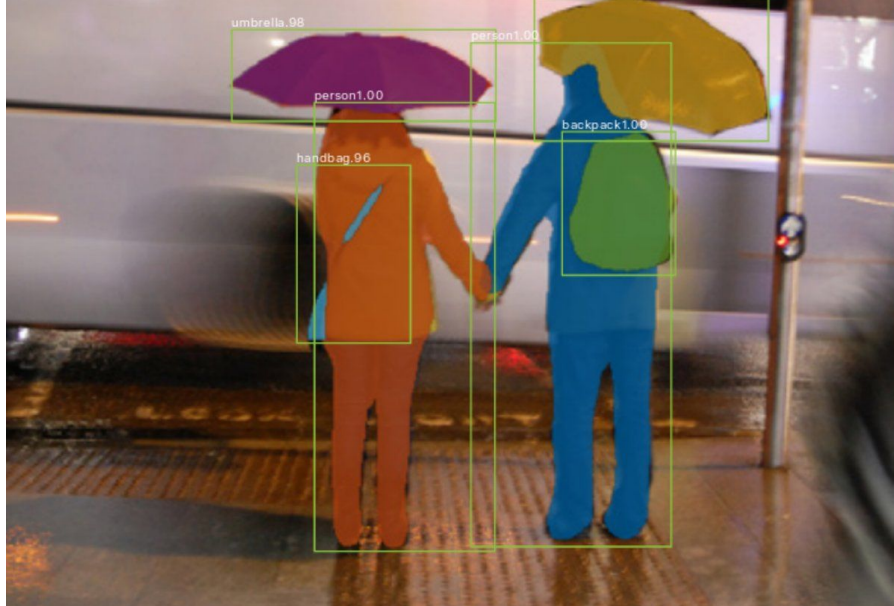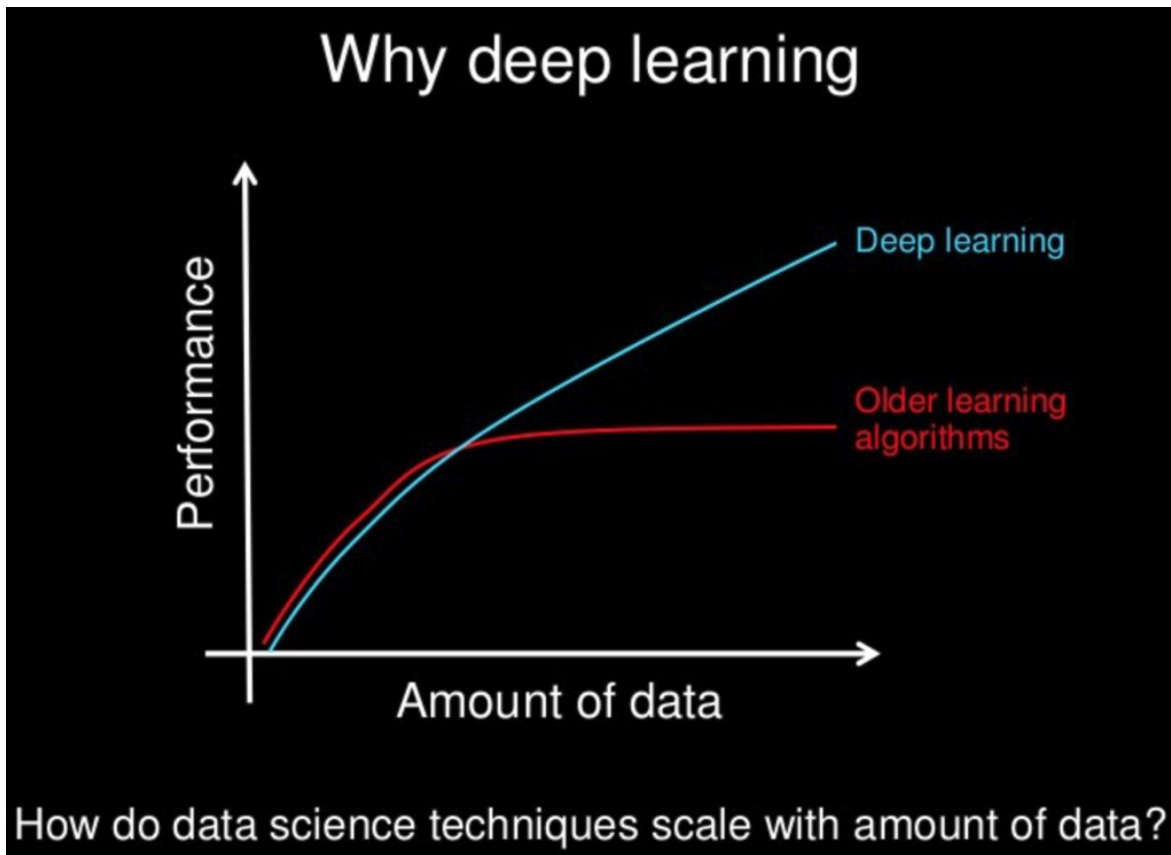
How is ML done?

- Machine learning often uses common pipeline with hand-designed feature extraction.
- Better models use End to End Approaches (CNN)

Feature Extraction → Machine Learning Algorithm → "Cat"?

Prior Knowledge, Experience

Why is vision so hard?

"Coffee Mug"

Pixel Intensity

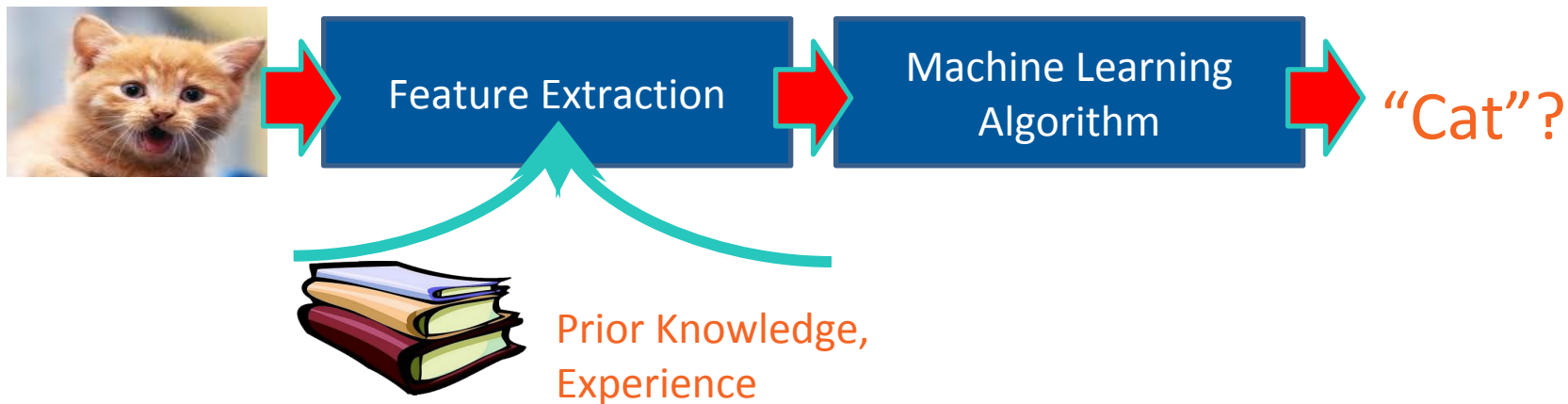| 177 | 153 | 118 | 91 | 85 | 100 | 124 | 145 |
151	124	93	77	86	115	148	168
115	93	78	83	108	145	177	191
88	79	84	104	136	168	190	197
82	85	103	127	152	170	180	182
91	101	120	138	150	157	159	159
103	114	127	136	140	140	140	141
111	119	126	130	130	129	128	130

Pixel intensity is a very poor representation.

Machine Learning

" Gives computer the ability to learn without being explicitly programmed "
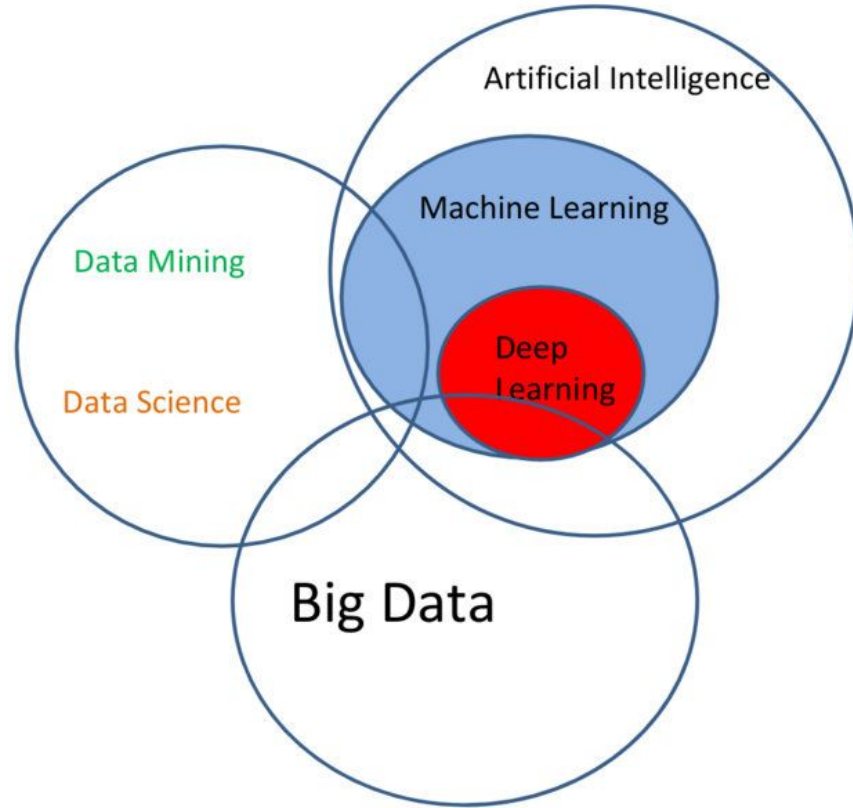
Types of ML

- Unsupervised learning

- Supervised learning (Focus)

- Reinforced Learning

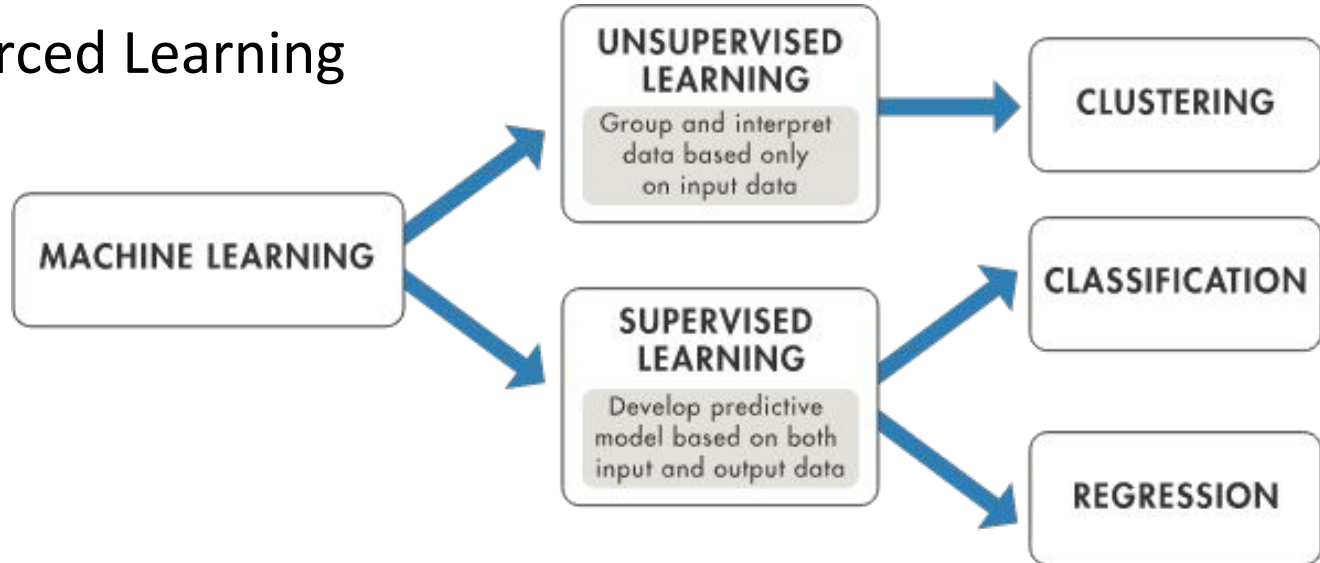MACHINE LEARNING

UNSUPERVISED LEARNING
Group and interpret data based only on input data

CLUSTERING

SUPERVISED LEARNING
Develop predictive model based on both input and output data

CLASSIFICATION

REGRESSION

Supervised Learning

- We train a model (Function, Program)

- Learn from training examples (Lots)

- Need labeled data
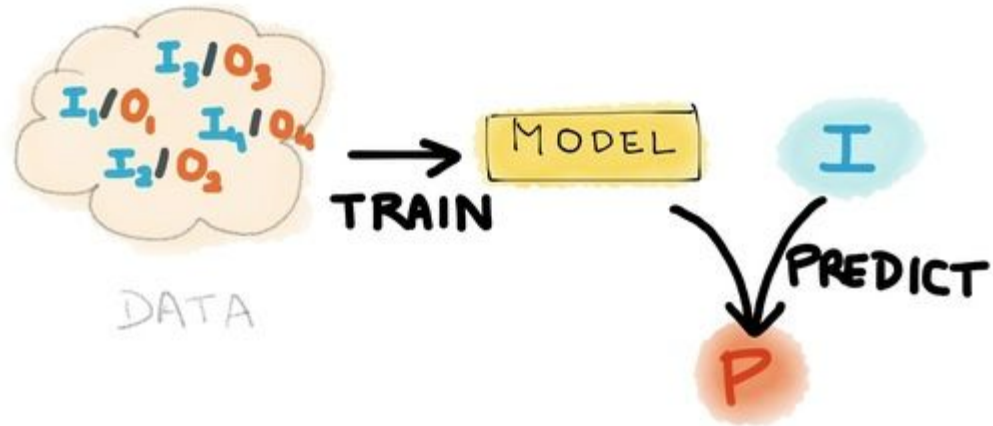
Supervised Deep Learning

Supervised Learning

- Given *labeled* training examples:

$$\mathcal{X} = \{(x^{(i)}, y^{(i)}) : i = 1, \ldots, m\}$$
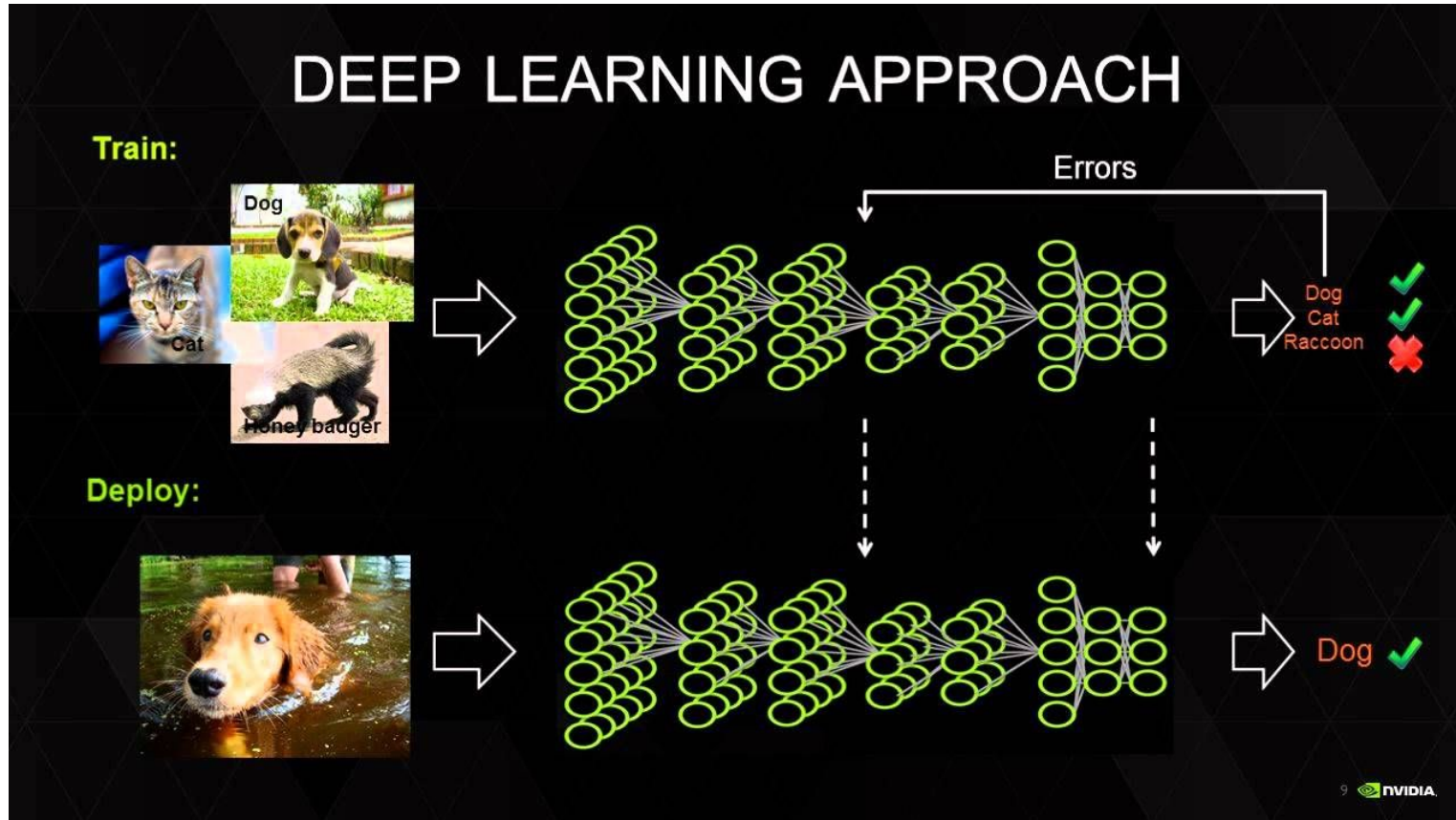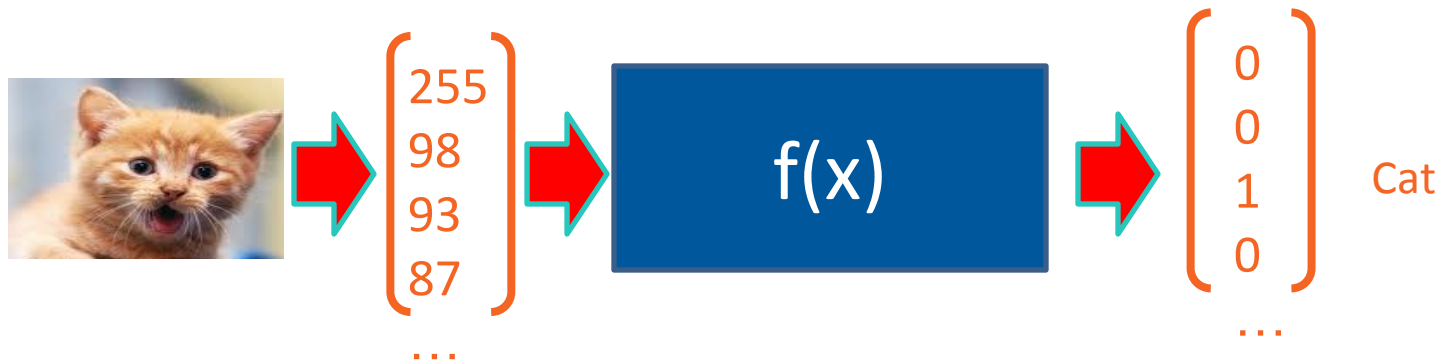
- Goal: find f(x) to predict y from x on training data.
  - Hopefully: learned predictor works on "test" data.

Neural network
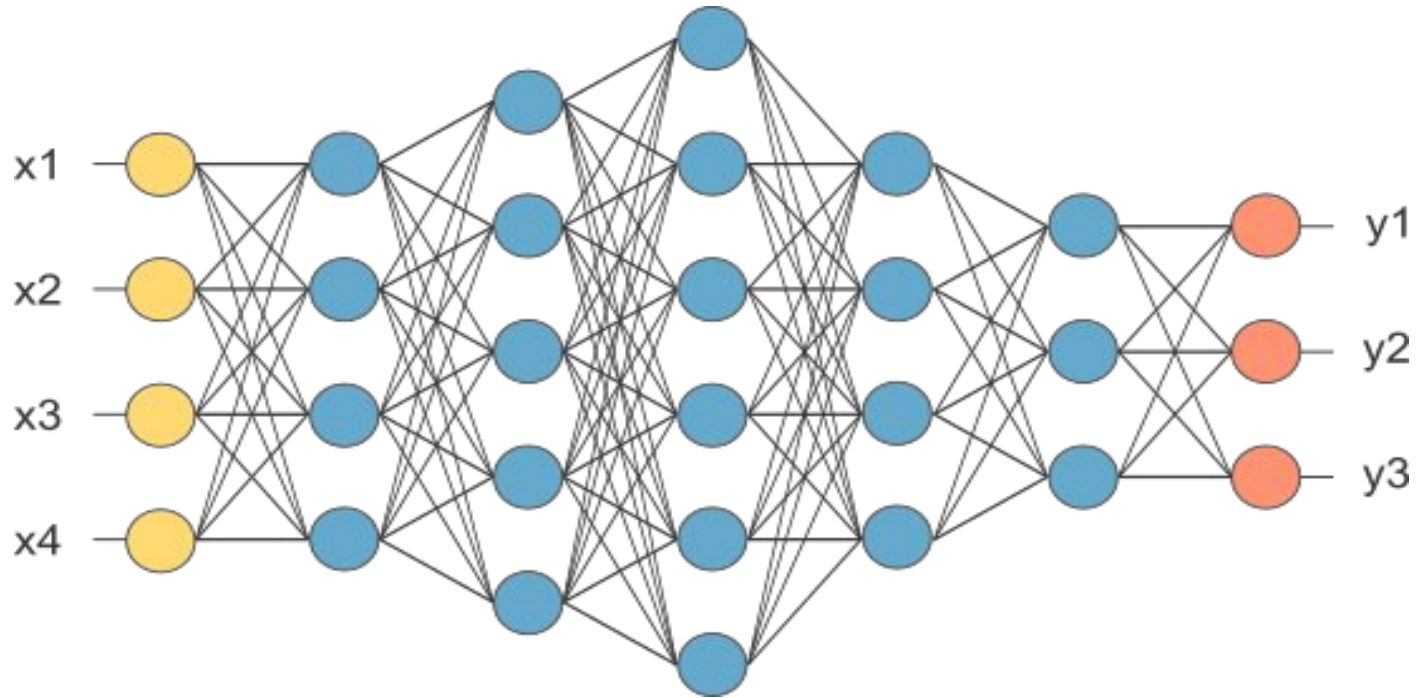
Neurons, Inspiration

A Typical Neuron
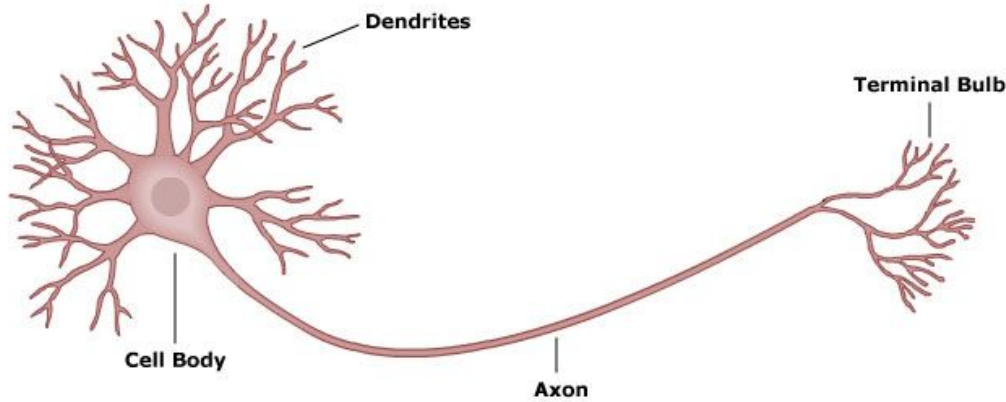
Neuron (Perceptron)

$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$
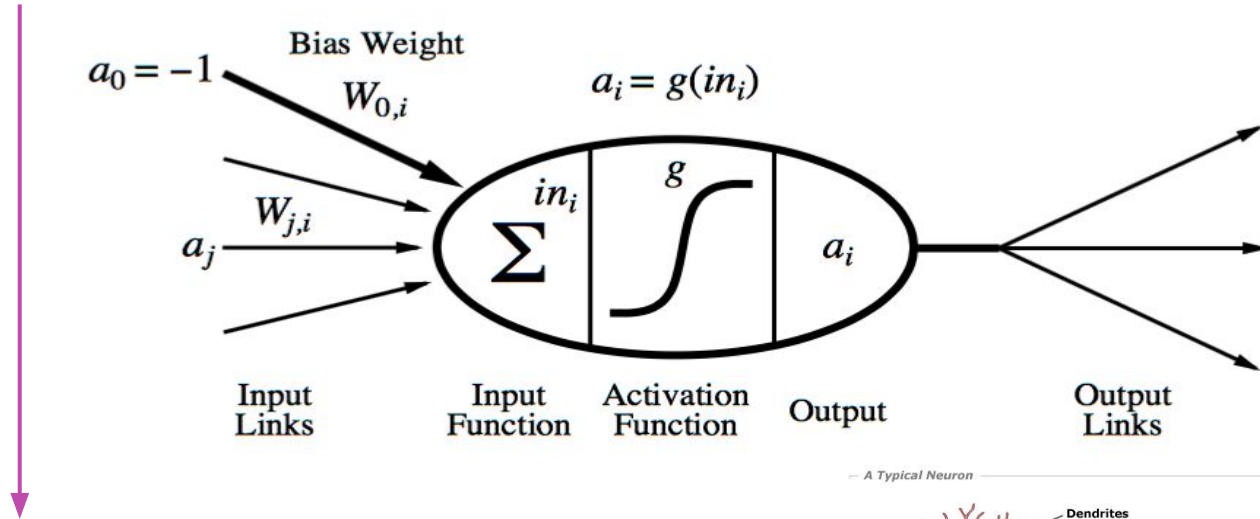
A Typical Neuron

Activations

tanh(z)

1

-1

abs(z)

1

ReLu(z) = max{0, z}

1

"Rectified Linear Unit"
→ Increasingly popular.
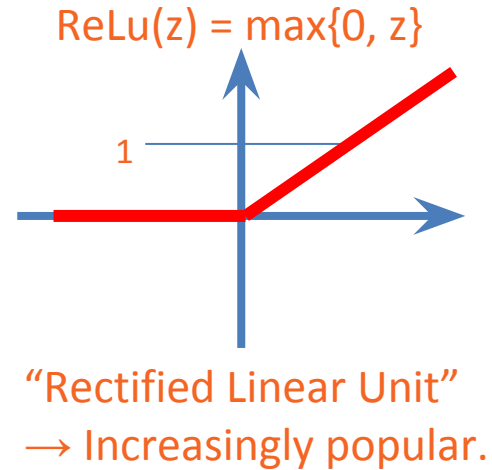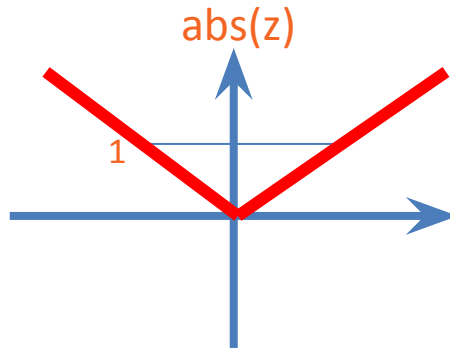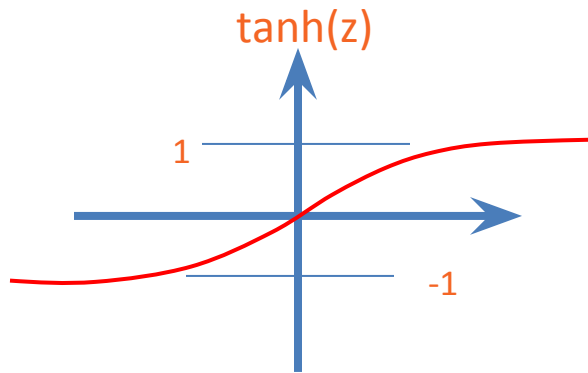
Neural network

- Can stack up several layers: Must learn multiple stages of internal "representation".

input layer

hidden layer 1   hidden layer 2

output layer

Neural network

- Can stack up several layers:

stretch pixels into single column

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W

| 56 |
| 231 |
| 24 |
| 2 |

x_i

$+$

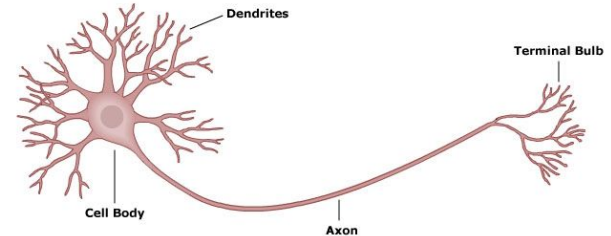| 1.1 |
| 3.2 |
| -1.2 |

b

$\rightarrow$

-96.8	cat score
437.9	dog score
61.95	ship score

$f(x_i; W, b)$

input image
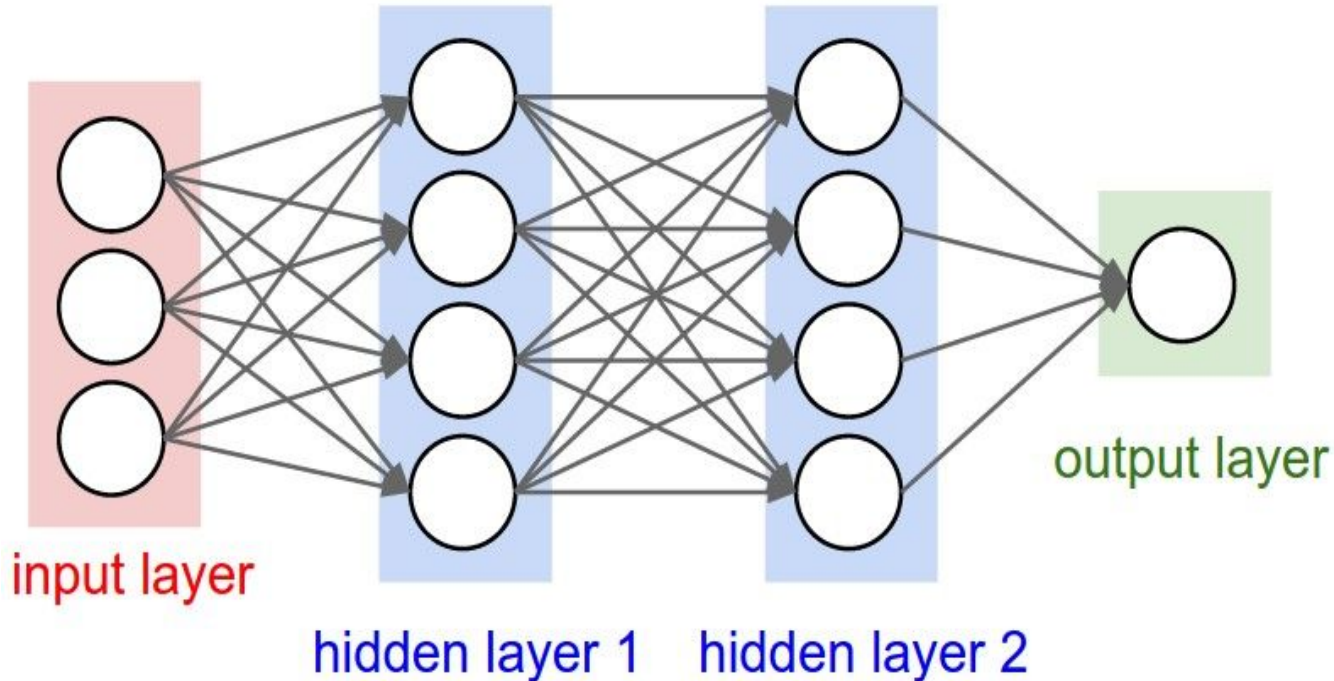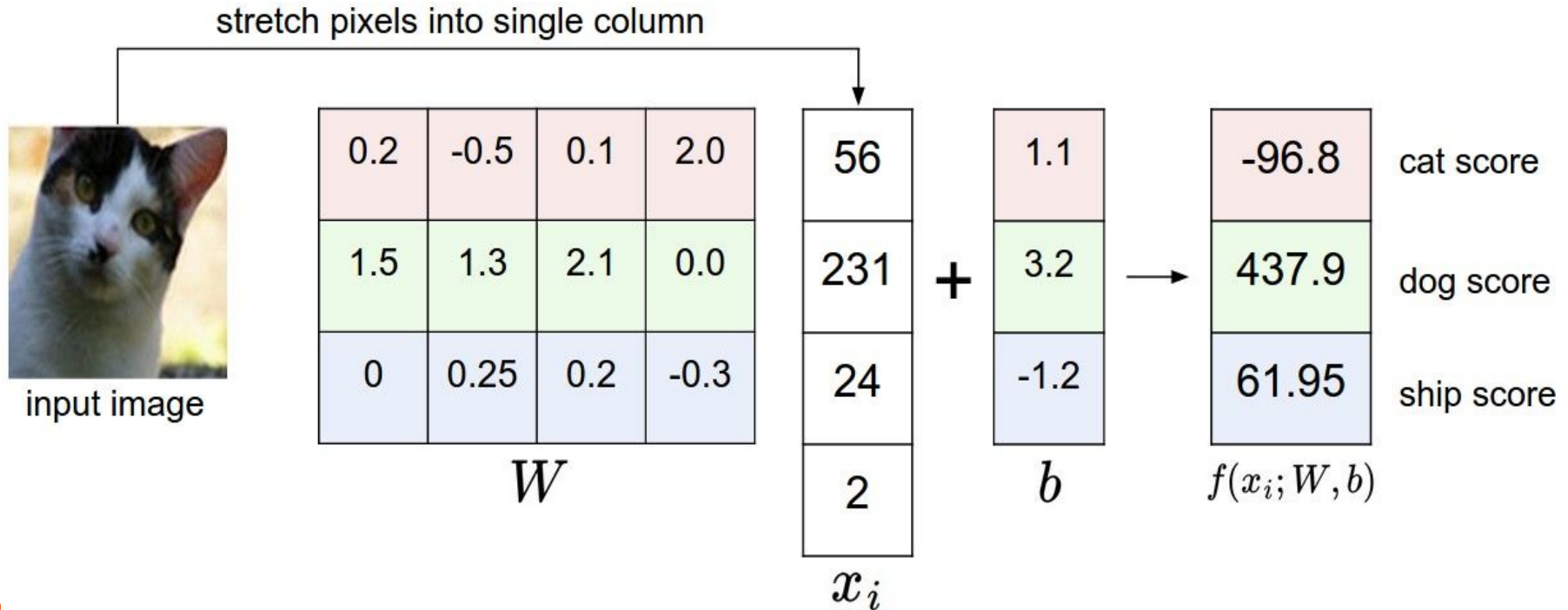
TensorFlow

- Deep Learning framework developed by Google
- The idea is to give a simple Framework for building Neural Networks
- Python & C++
- C++ Acceleration
- CUDA + GPU acceleration
- Open source

Tip

TensorFlow has amazing tutorials online on YouTube and in their website

Installation

- Install Python3.5 (Windows), add Environment variables
- Use pip3 to install packages
- pip3 install --upgrade tensorflow
- pip3 install --upgrade tflearn

Tip

TensorFlow has amazing tutorials online on YouTube

Steps

Build a model

First we need to define our model, using TFLearn classes
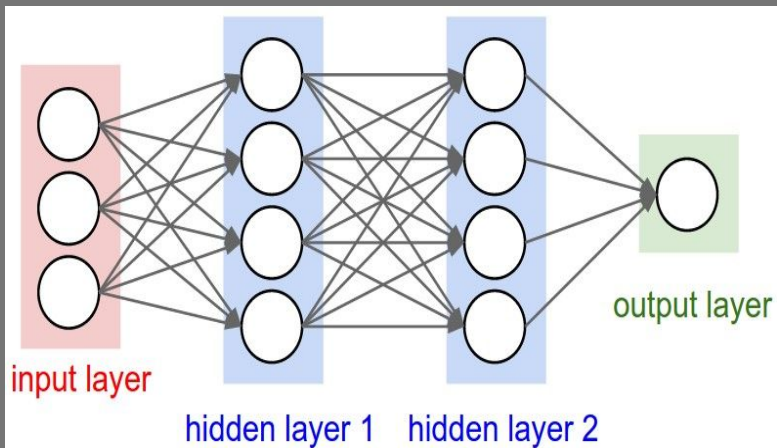
Train the model

We will use the MNIST dataset in-order to train our model to recognize digits

Use the model

Use our trained model to detect digits.

Building the model

- Input Layer (28*28 = 784)
- Fully Connected (64), tanh
- Fully Connected (64), tanh
- Fully Connected (10), softmax

input layer

hidden layer 1   hidden layer 2

output layer

```
# Building deep neural network
input_layer = tflearn.input_data(shape=[None, 784])
dense1 = tflearn.fully_connected(input_layer, 64, activation='tanh',
                                 regularizer='L2', weight_decay=0.001)
dropout1 = tflearn.dropout(dense1, 0.8)
dense2 = tflearn.fully_connected(dropout1, 64, activation='tanh',
                                 regularizer='L2', weight_decay=0.001)
dropout2 = tflearn.dropout(dense2, 0.8)
softmax = tflearn.fully_connected(dropout2, 10, activation='softmax')
```

Training the model

- Use MNIST dataset - we download it from the web
- 55k Images - labeled and separated to training and validation
- One-hot : Labels are encoded as binary vectors

```
# Data loading and preprocessing
import tflearn.datasets.mnist as mnist
X, Y, testX, testY = mnist.load_data(one_hot=True)
```

- SGD Optimizer: Loss update function (Training)
- Cross Entropy loss: Loss calculation function

```
sgd = tflearn.SGD(learning_rate=0.1, lr_decay=0.96, decay_step=1000)
top_k = tflearn.metrics.Top_k(3)
net = tflearn.regression(softmax, optimizer=sgd, metric=top_k,
                         loss='categorical_crossentropy')
```

Running the session

- Training set: our Training dataset. Images and labels
- Validation set: Small part of the data for validation
- N_epoch: Number of times we train the model

```
# Training
model = tflearn.DNN(net, tensorboard_verbose=0)
model.fit(X, Y, n_epoch=20, validation_set=(testX, testY),
          show_metric=True, run_id="dense_model")
```

Saving the network

- Save a model to a .model file (Saves network + weights)
- Load model weights from a file

```
model.save('mnist.model')
model.load('mnist.model')
```

Using the network

- Load a trained model
- Load a new image for detection
- Reshape image to suit our input
- Use the trained model to predict the image label

```python
def loadImage(path):
    return np.reshape(Image.open(path).resize((28, 28)), [28*28*3])

image = loadImage("images/test/test2.jpg")
model.predict([image])
```