An Introduction to TensorFlow for Deep Learning, Predictive Analytics and Data Science

Limassol, Cyprus
21 June 2018

https://goo.gl/FpQptB

Special thanks!

- **PyData Cyprus**, for organizing the event
- **Cyprus University of Technology**, for the venue
- **DisruptCyprus,** for the live link: https://www.facebook.com/disruptcyprus/videos/1657219760998625/

Cyprus University of Technology

PyData

An educational program of NUMFOCUS

OPEN CODE = BETTER SCIENCE

Contents

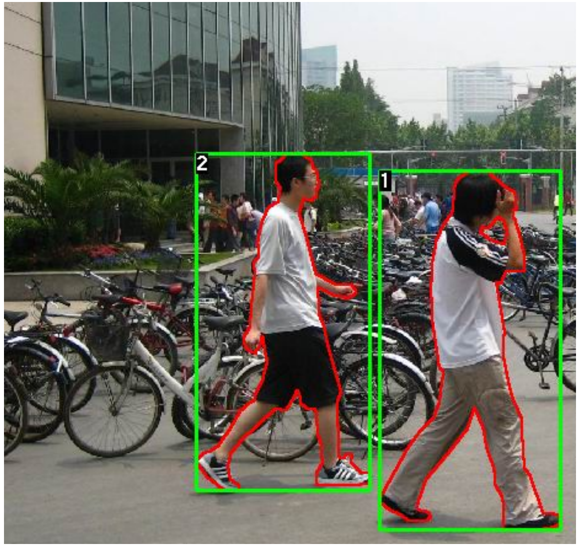
Motivation

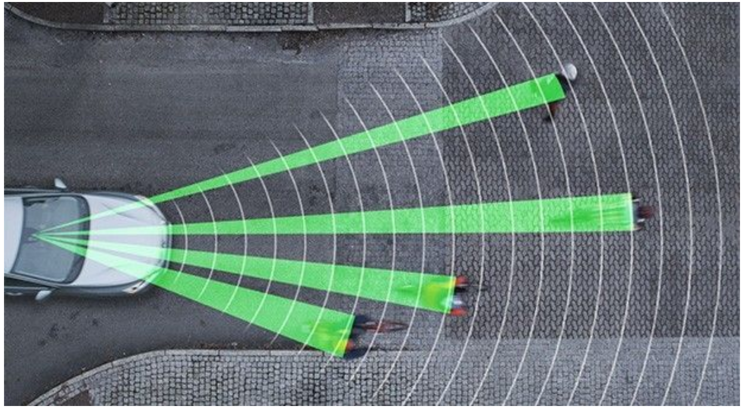
- Artificial Intelligence (AI) & Machine Learning (ML) are all over the news online
- All Tech Giants are particularly active in these fields
  - Google Research, Google Brain, Facebook AI Research (FAIR), IBM Watson, ...
- **TensorFlow** is an open source library and it was released in 2015 by Google
- It recently became the “de facto” software library for *Deep Learning*
- It is used for both **Research** and **Production Development** at Google
- Additional companies using TensorFlow include:
  - DeepMind, Twitter, Dropbox, SAP, Lenovo, Intel, NVIDIA, Qualcomm, ARM, AMD, Airbus, ebay, airbnb, UBER and many other…
- Predictive Analytics and Data Science make use of Machine Learning
- Machine Learning is applicable to any kind of data, under certain *assumptions*

https://goo.gl/FpQptB

Motivation

- Typical tasks in ML include:
  - Handwritten Digit Recognition (e.g MNIST dataset)
  - Pedestrian Detection and Recognition
  - Speech Recognition
  - Natural Language Processing, e.g. chat bots
  - Automated Translation
  - Robot Localization and Mapping
  - Autonomous vehicles
  - Financial prediction or any kind of prediction
  - Genomics and Bioinformatics
  - etc, etc, ...





https://goo.gl/FpQptB

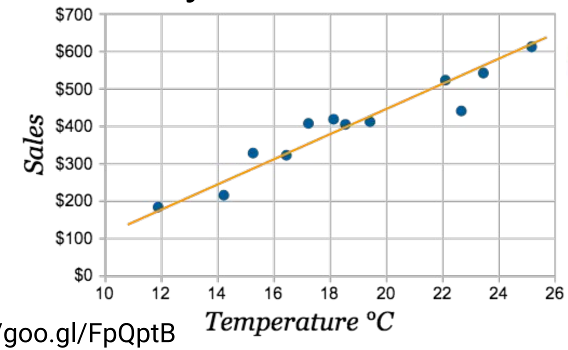
What is Machine Learning

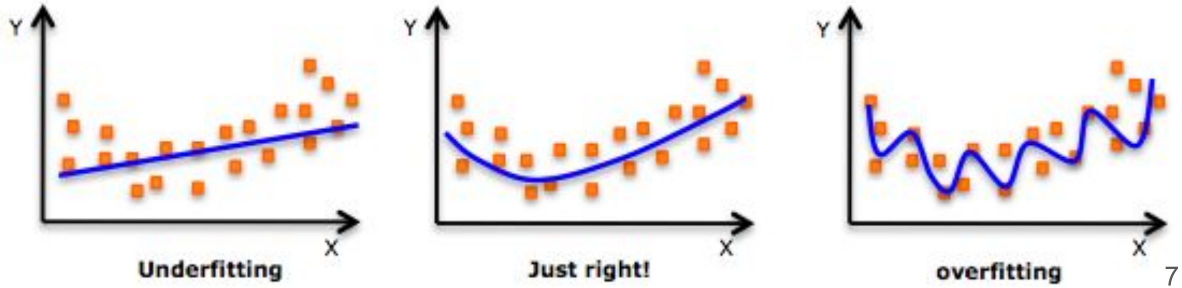
- Let’s say a Machine Learning ***Model*** follows certain *assumptions*
- Think of this *Model* as a **Black Box** function, for now
- In *Systems Theory*, this black box is considered to be an **Open System**
- Imagine Input to be a *matrix* **A** (i.e. the *Dataset*) with **m** rows and **n** columns
- Imagine Output to be a *vector* **b** with **m** rows (and 1 column)
- Rows are the number of *observations* or *experiments* or *examples*
- Each column represents a *variable* or *attribute* or *feature*
- Linear case: **A***x = **b**, where x is a vector with **n** rows (and 1 column)

https://goo.gl/FpQptB

What is Machine Learning

- Linear case: **A*x = b**, where x is a vector with **n** rows (and 1 column)
- **Regression**: **m** (distinct) equations and **n** unknown variables
- If **m=n**, then there is a single solution, so you don't need ML
- If **m<n**, then ML *cannot* be used (*undetermined system*)
- If **m>n**, then ML can possibly be used (*overdetermined system*)
- Simplest examples in *Regression*: **Best-fit Line** or **Best-fit Curve**
- Usually ML is more concerned with **Classification/Prediction** than *Regression*





https://goo.gl/FpQptB

What is Machine Learning

- This (*classification*) dataset **A** has **m=10** rows and **n=3** columns
- This target vector **b** has **m=10** rows (and 1 column), called *labels*

	Cloudy? (Yes/No)	Temperature (C)	Humidity
1	1	38	80
2	0	18	60
3	0	26	45
4	0	13	78
5	1	14	75
6	1	37	55
7	0	23	50
8	1	35	48
9	1	19	78
10	0	15	61

	Raining? (Yes/No)
1	0
2	1
3	0
4	1
5	1
6	0
7	0
8	0
9	1
10	1

https://goo.gl/FpQptB

What is Machine Learning

- All Models make assumptions! (not only in ML)
- Assumptions need to be valid and taken into account during Interpretation
- “All models are wrong, but some are useful” - a famous quote
- “Garbage in, Garbage out (GIGO)” - so be careful ...
- Once a *Model* is set-up and dataset is used, we need a Computational ***Method***
- Each *Method* relies on one or more **Algorithms**
- What is the best Algorithm? It depends!
- In general, Algorithms were known since Antiquity (e.g. Euclidean Algorithm)

https://goo.gl/FpQptB

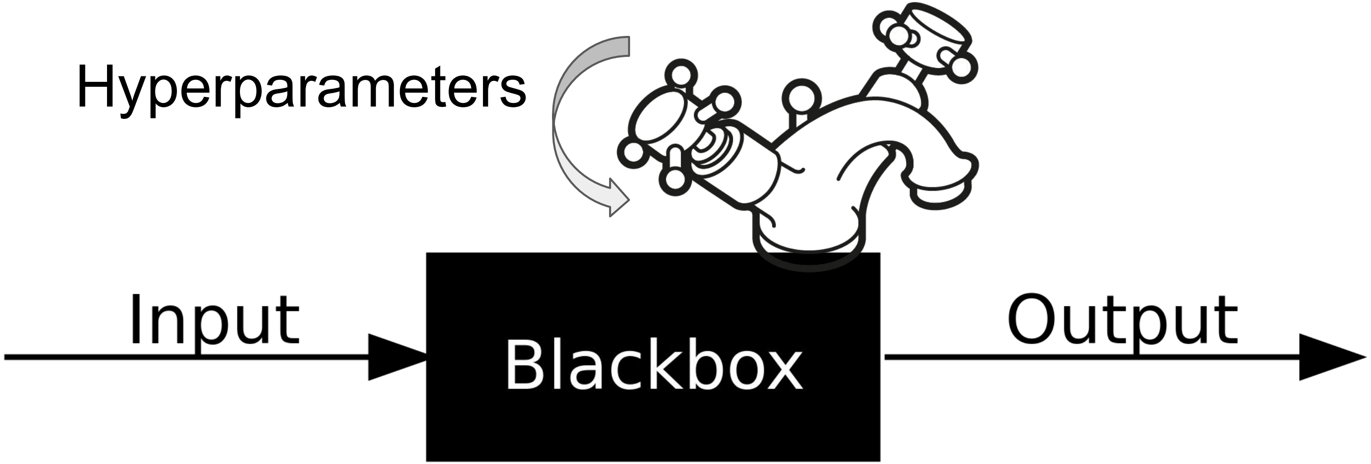
What is Deep Learning

- A very powerful *Model* (or *System*) is the **Artificial Neural Network (ANN)**
- This is not a new concept and it has been around for more than 50 years
- Machines did not have enough computational power to use ANNs
- Today CPUs are much faster than before and GPUs are even faster!
- Apart from *offline* (i.e. at our local machine), any Machine Learning Model can also run *online* (i.e. in the cloud) and also in a ***parallel*** or ***distributed*** way
- Let’s consider ANN to also be a Black Box for now
- In addition to Input and Output, ANNs also have **Hyperparameters**

https://goo.gl/FpQptB

What is Deep Learning

- You can think of **Hyperparameters** as neither Input nor Output
- Usually, the human (e.g. Data Scientist or AI Researcher) needs to find the "best" (i.e. as "optimal" as possible) value for each Hyperparameter
- Very high value is not desirable and very small value is also not desirable
- There needs to be a '**trade-off**' but ANN does *not* follow linear relationships



https://goo.gl/FpQptB

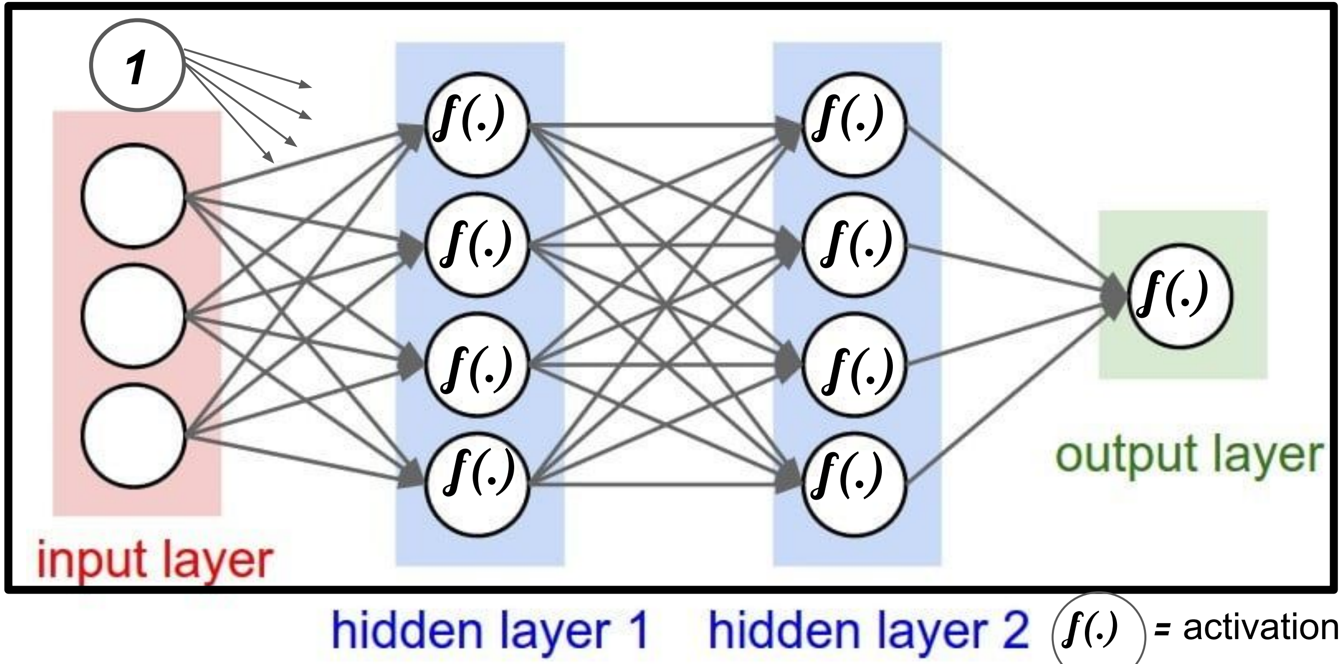
What is Deep Learning

- ANNs try to imitate the biological neurons of the brain, using simplifications
- Let’s look at inside the Black Box, to better understand what is going on:

Cloudy? (Yes/No)

Temperature

Humidity



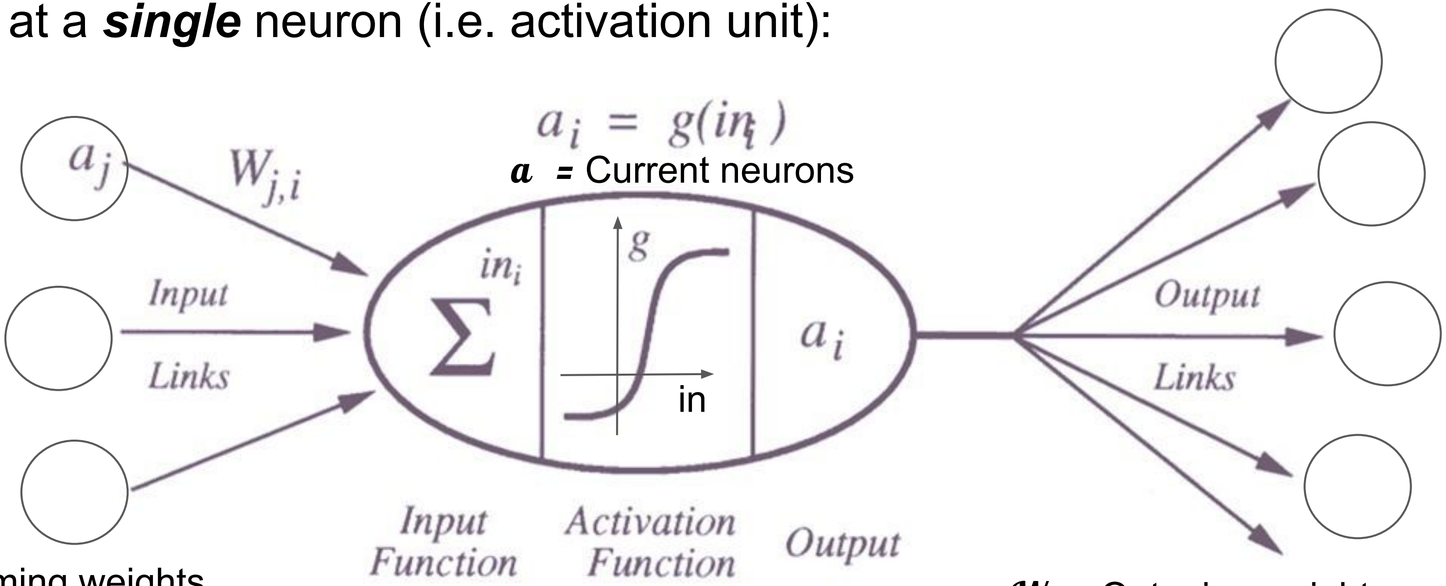
Raining? (Yes/No)

f(.) = activation function

https://goo.gl/FpQptB

What is Deep Learning

- Let’s look at a ***single*** neuron (i.e. activation unit):



W = Incoming weights

a = Previous neurons

$$\boxed{in_i = \sum_j W_{j,i} a_j}$$

W = Outgoing weights

a = Next neurons

$g(.)$ = activation function

https://goo.gl/FpQptB

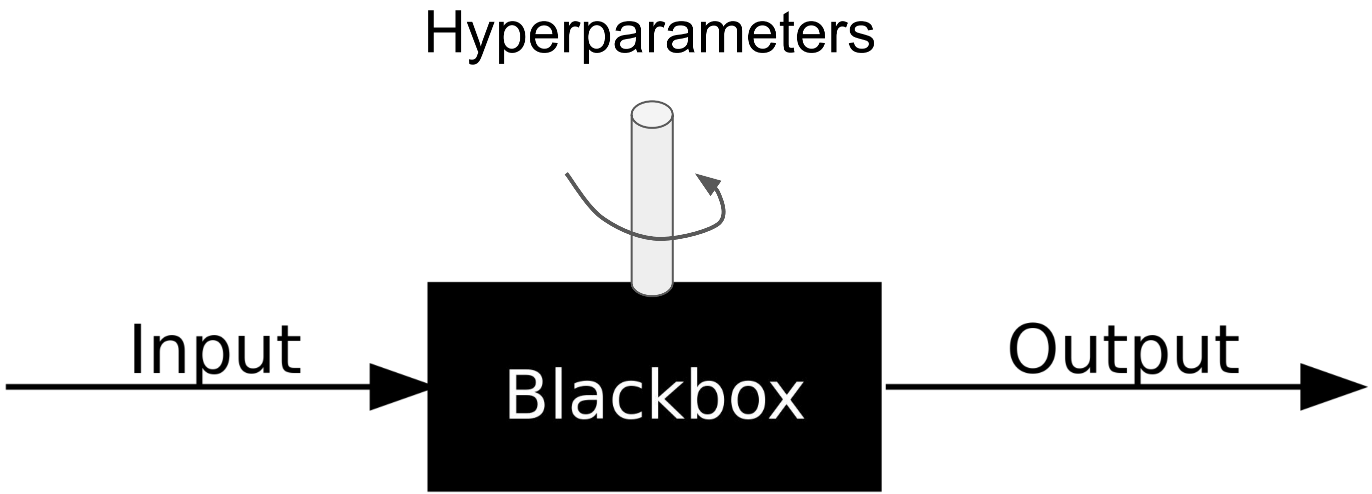
What is Deep Learning

- The *number of hidden layers* and *number of neurons* in each layer, are two of the most common **Hyperparameters** that need to be ‘tuned’
- The human (e.g. Data Scientist or AI Researcher) adjusts the hyperparameters and then ‘trains’ the ANN and this is repeated many times
- During training, some iterative Methods, called **Optimization** Algorithms, “go through” the data (**Backpropagation** Algorithm) to infer the *Weight* values
- The Optimization Methods also include Hyperparameters (e.g. ***learning rate***)
- The whole process is indeed very complicated, but mathematically sound
- The whole **Training** is of ‘random’ or ‘stochastic’ nature, i.e. the exact same data and the exact same Hyperparameters give different results every time!
- A lot of *intuition* is needed from the human during the training phase of ANNs

https://goo.gl/FpQptB

What is Deep Learning

- Let's go back to the Black Box of ANNs
- After the Training phase, Hyperparameters should be kept fixed
- The ANN is now fully trained and ready to make **Predictions** or **Classifications**, i.e. compute the Output based on (previously unseen) Input data



https://goo.gl/FpQptB

Model Evaluation Metrics

- Now the ANN has already been fully trained using the **Training set**
- But how “good” is the ANN that we have trained?
- Certain Model Evaluation Metrics are used on an (unseen) **Test set**
- The Test set usually has the same number of columns, but not rows
- ANN make predictions for Output based on Test set as Input
- Then a comparison can be made between *Predicted* and *Actual* values
- Definitions:
  - **True positive (TP):** Predicted label was POSITIVE and the actual label was indeed POSITIVE
  - **True negative (TN):** Predicted label was NEGATIVE and the actual label was indeed NEGATIVE
  - **False positive (TF):** Predicted label was POSITIVE but the actual label was actually NEGATIVE
  - **False negative (FN):** Predicted label was NEGATIVE but the actual label was actually POSITIVE

https://goo.gl/FpQptB

Model Evaluation Metrics

- Most metrics are based on the **Confusion Matrix**
- The most common and simple metric is the **Accuracy**

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

specificity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

precision or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

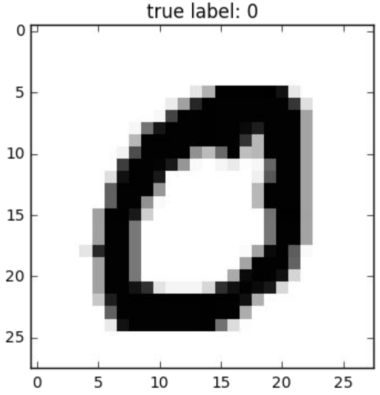
accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

https://goo.gl/FpQptB

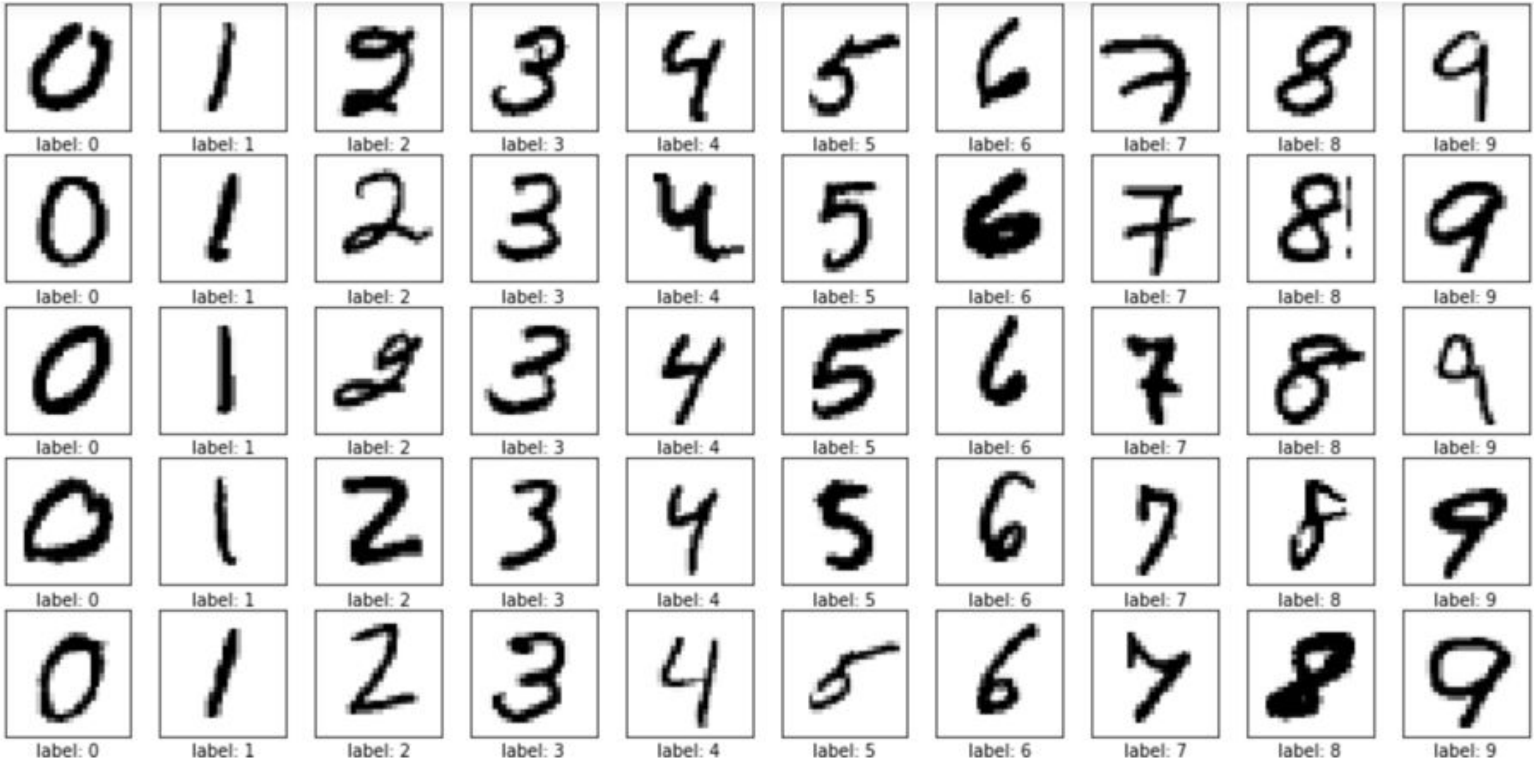
MNIST dataset

- This is the most famous dataset in ML and one of the oldest ones
- In the world of datasets, it is equivalent to the ‘Hello World’ phrase
- Most publications by universities and companies will demonstrate that their newly-invented ML Algorithm is indeed working at least on MNIST dataset
- It is also considered a good example for educational purposes
- It contains many rows of ‘scanned’ handwritten digits (0 to 9) as ***digital images***
- Each pixel of the (grayscale) image (**28x28** pixels) has a value which represents the intensity of this pixel
- Full black is 255 and full white is 0, grey is in-between
- So, each row is one digital image and the **784** *columns* represent the intensity (0 to 255) of each pixel



https://goo.gl/FpQptB

MNIST dataset



https://goo.gl/FpQptB

Using the cloud

- TensorFlow, just like most libraries, can run in the cloud
- Any cloud service can be used for running TensorFlow, including:

amazon web services™

DigitalOcean

Microsoft Azure

Google Cloud

IBM® BlueMix

https://goo.gl/FpQptB

TensorFlow + Keras libraries

- Import the ***libraries*** (TF, Keras) and create the sessions [Mandatory first step]

```
import tensorflow as tf
sess = tf.Session()

from keras import backend as K
K.set_session(sess)
```

- Import the MNIST ***dataset*** (in this case, MNIST is part of TensorFlow)

```
from tensorflow.examples.tutorials.mnist import input_data
mnist_data = input_data.read_data_sets('MNIST_data', one_hot=True)
# this placeholder will contain our input images, as flat vectors (28x28=784)
input_img = tf.placeholder(tf.float32, shape=(None, 784)) # number of rows (None) can vary
act_labels = tf.placeholder(tf.float32, shape=(None, 10)) # labels for 10 digits (0 to 9)
```

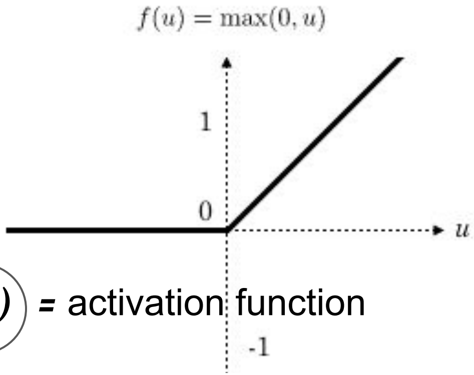
https://goo.gl/FpQptB

TensorFlow + Keras libraries

- Define the ANN ***Model*** with 2 hidden layers and 128 neurons for each layer

```python
from keras.layers import Dense # Dense refers to the fully-connected Artificial Neural Network

# Keras layers can be called on TensorFlow tensors:
h1 = Dense(128, activation='relu')(input_img)   # Hidden Layer with 128 neurons and ReLU activation
h2 = Dense(128, activation='relu')(h1)          # Another Hidden Layer, similar to the above
out_pred = Dense(10, activation='softmax')(h2)  # Output Layer with 10 units and Softmax activation
```

- **ReLU** (Rectified Linear Units) is the simplest nonlinear function you can think of
- **Softmax** function is a bit more complicated, but still a nonlinear function too



f(.) = activation function

https://goo.gl/FpQptB

TensorFlow + Keras libraries

- Define the ***Method*** (i.e. Optimization) to be used for training the *Model*

```
from keras.objectives import categorical_crossentropy

loss = tf.reduce_mean(categorical_crossentropy(act_labels, out_pred)) # To be minimized
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(loss)    # Optimization
```

- *Gradient Descent* is the name of the Optimizer and 0.5 is the *learning rate*
- Loss (or Cost) is the function the Optimizer will need to minimize and in this case it is the **Cross Entropy**, adapted for *categorical* data (our 10 categories)

$\hat{\mathbf{y}}$ $\mathbf{y}$

Predicted labels
Range: (0,1)

$$D(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_j y_j \ln \hat{y}_j$$

Actual labels
Range: [0,1]

https://goo.gl/FpQptB

TensorFlow + Keras libraries

- Initialize all variables and “run” the session [Mandatory step in TensorFlow]

```
init_op = tf.global_variables_initializer() # Initialize all variables
sess.run(init_op)
```

- Training phase (inside a For-Loop) in ***Mini-Batches*** (i.e. not all data at once)

```
with sess.as_default():
    for i in range(100): # Let’s try 100 iterations (called “epochs”)
        batch = mnist_data.train.next_batch(50) # Select just 50 rows as Input (Mini-Batch)

        # Training phase (Optimization)
        train_step.run(feed_dict={input_img: batch[0],    # Predict Labels based on Input
                                  act_labels: batch[1]})  # Use Actual Labels for BackBrop
```

https://goo.gl/FpQptB

TensorFlow + Keras libraries

- Training phase has now finished, so the ANN has been trained
- Perform Model Evaluation (using the **Test set** and *not* the Training Set)

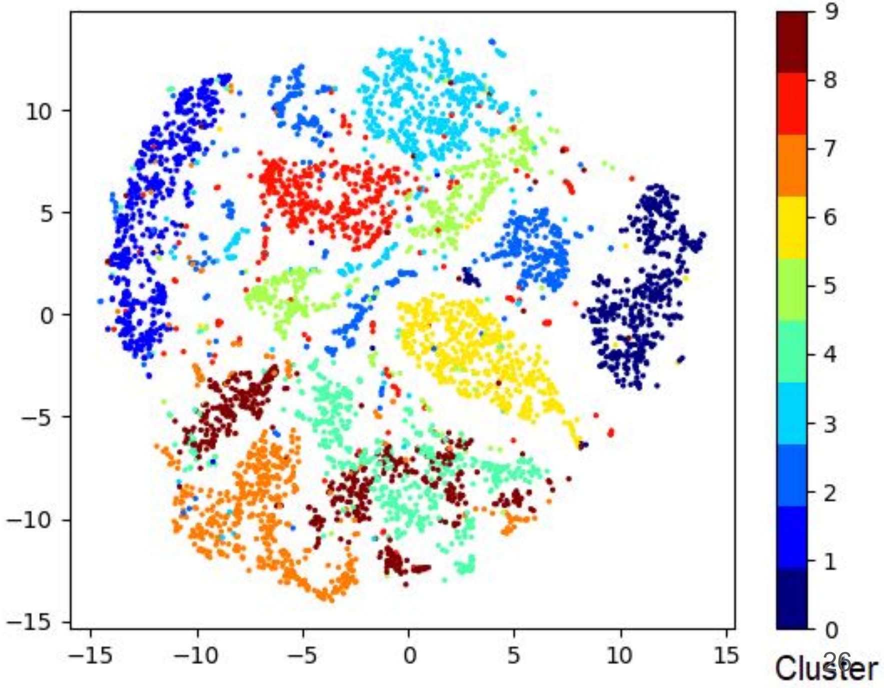
```
from keras.metrics import categorical_accuracy as accuracy


# Define and then calculate Accuracy based on Actual and Predicted Labels
acc_value = accuracy(labels, out_pred) with sess.as_default():
    print acc_value.eval(feed_dict={input_img: mnist_data.test.images,   # Predict Labels
                                    act_labels: mnist_data.test.labels}) # Actual Labels
```

- Usually, the final step and most crucial step is the **Data Visualization**
- Data Visualization will be discussed at a next presentation and talk

https://goo.gl/FpQptB

Data Visualization

- Famous Data Visualization libraries in Python include:
    - matplotlib
    - Seaborn
    - ggplot
    - Bokeh
    - pygal
    - Plotly
    - geoplotlib
    - Gleam
    - missingno
    - Leather



https://goo.gl/FpQptB

Resources and Further Information

- TensorFlow
  https://www.tensorflow.org/
- Keras
  https://keras.io/
- My personal wiki
  https://wiki.kourouklides.com/wiki/Machine_Learning
- Kaggle: Practise ML online and also participate in competitions
  https://kaggle.com/
- KDnuggets (blog posts)
  https://www.kdnuggets.com/