FRAMEWORKS

- Frameworks play a crucial role in the domain of data science
- It is a collection of individual software components available in code form and ready to run (what we call as libraries)
- Collection of packages and libraries which help in simplifying the overall programming experience for building a specific kind of application

1. TensorFlow
2. Scikit-learn
3. Keras
4. Pandas
5. Spark MLib
6. PyTorch
7. Caffe
8. Deeplearning4j
9. Seaborn
10. Theano

Keras & TensorFlow

- **Keras** is an open-source software library that provides a Python interface for artificial neural networks
- It supports multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML

- **Tensorflow** is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers

Keras

TensorFlow

- **Keras is a neural network library while TensorFlow is the open-source library for a number of various tasks in machine learning**

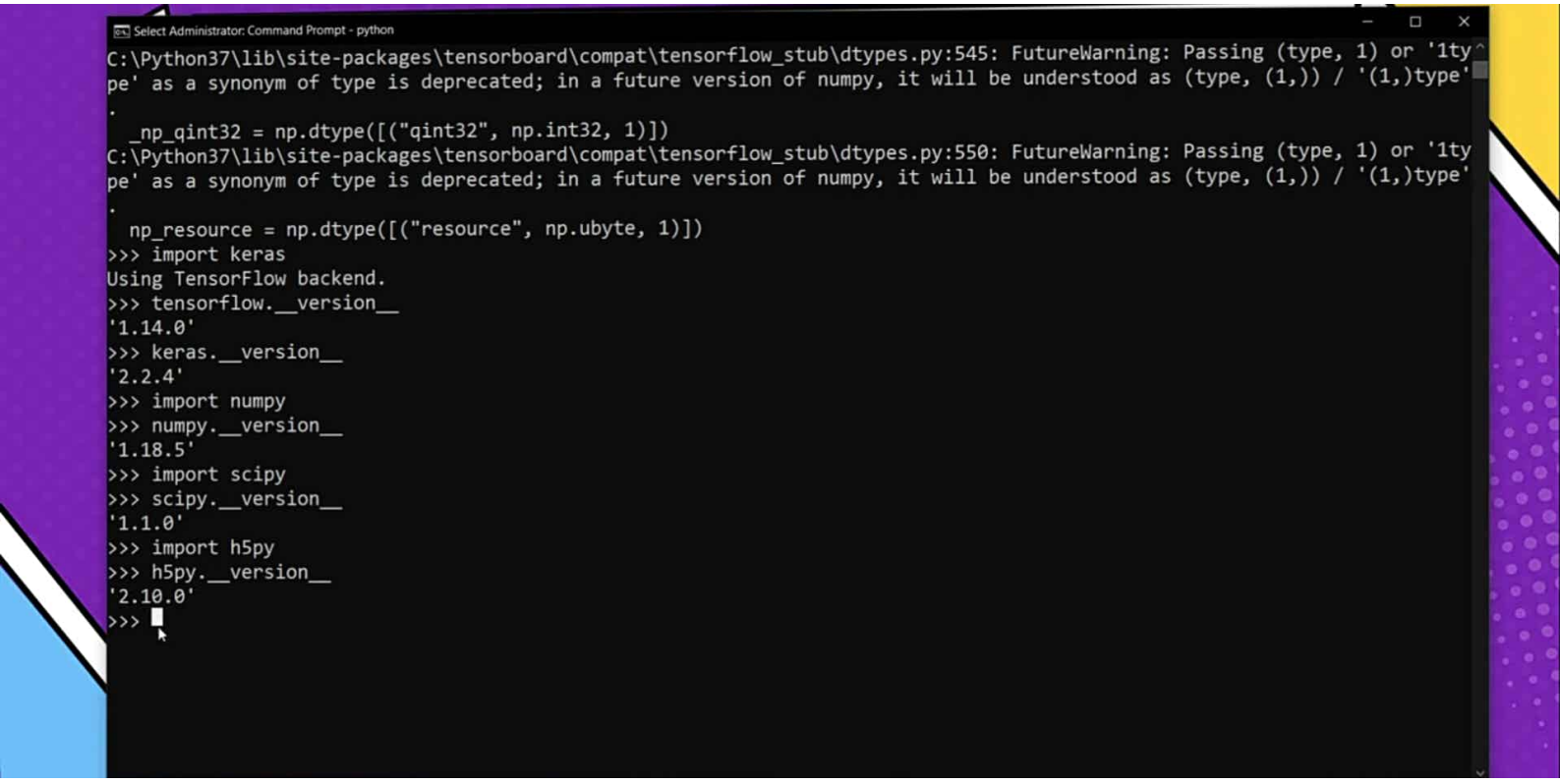
- **TensorFlow provides both high-level and low-level APIs while Keras provides only APIs**

Installing Keras & Tensorflow

Pip install tensorflow==1.14.0

Pip install keras==2.2.4

Pip install h5py==2.10.0


Select Administrator: Command Prompt - python
C:\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Python37\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
np_resource = np.dtype([("resource", np.ubyte, 1)])
>>> import keras
Using TensorFlow backend.
>>> tensorflow.__version__
'1.14.0'
>>> keras.__version__
'2.2.4'
>>> import numpy
>>> numpy.__version__
'1.18.5'
>>> import scipy
>>> scipy.__version__
'1.1.0'
>>> import h5py
>>> h5py.__version__
'2.10.0'
>>>

Method of training ur model

Pre-trained model

Building from scratch

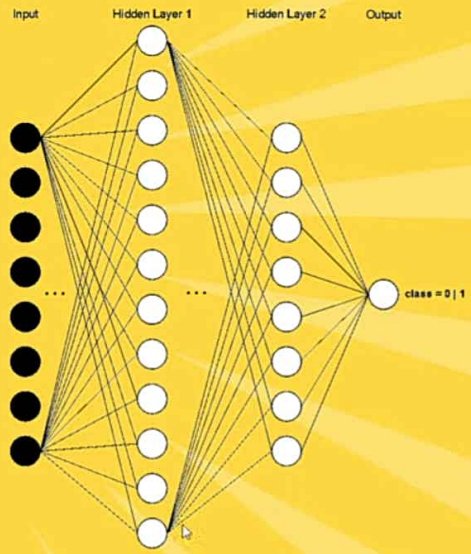
Transfer Learning

Keras Basic Syntax

- **Sequential API** is simply arranging the Keras layers in a sequential order and so, it is called Sequential API
- Most of the ANN also has layers in sequential order and the data flows from one layer to another layer in the given order until the data finally reaches the output layer.
- **Adding Layers**

```
model.add(Dense(12, input_dim=8, init='uniform', activation='relu'))

model.add(Dense(8, activation='relu'))
```



Keras Basic Syntax

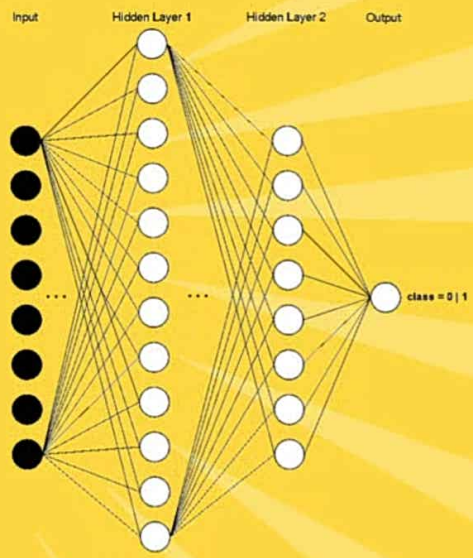
- **Compile Model**

```
model.compile(loss='binary_crossentropy',
              optimizer='adam', metrics=['accuracy'])
```

- **Adding Layers**

```
model.add(Dense(12, input_dim=8, init='uniform', activation='relu'))

model.add(Dense(8, activation='relu'))
```



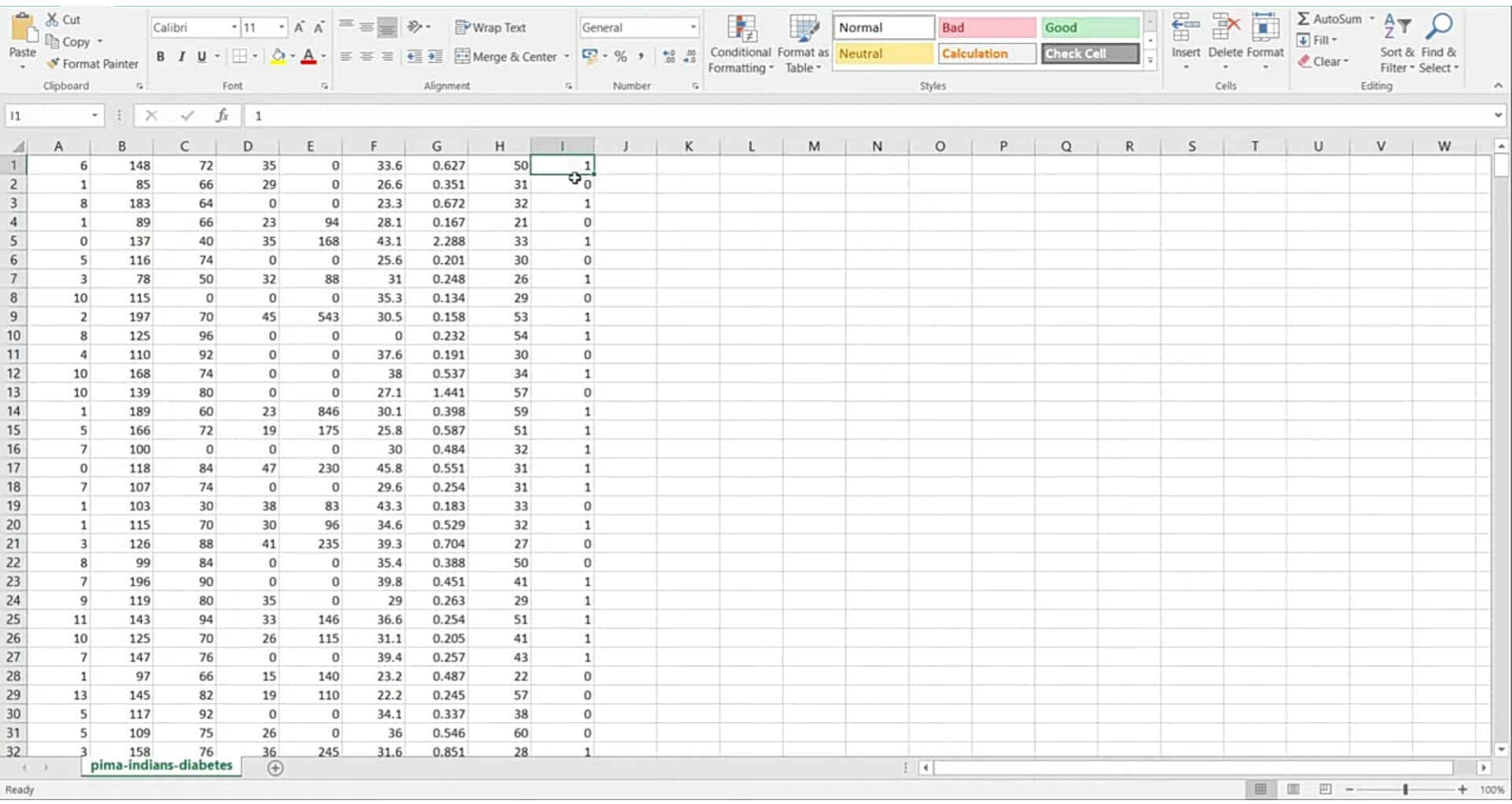
Save & Load Model

Save Model

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("model.h5")
print("Saved model to disk")
```

Load Model

```
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
loaded_model.load_weights("model.h5")
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	6	148	72	35	0	33.6	0.627	50	1														
2	1	85	66	29	0	26.6	0.351	31	0														
3	8	183	64	0	0	23.3	0.672	32	1														
4	1	89	66	23	94	28.1	0.167	21	0														
5	0	137	40	35	168	43.1	2.288	33	1														
6	5	116	74	0	0	25.6	0.201	30	0														
7	3	78	50	32	88	31	0.248	26	1														
8	10	115	0	0	0	35.3	0.134	29	0														
9	2	197	70	45	543	30.5	0.158	53	1														
10	8	125	96	0	0	0	0.232	54	1														
11	4	110	92	0	0	37.6	0.191	30	0														
12	10	168	74	0	0	38	0.537	34	1														
13	10	139	80	0	0	27.1	1.441	57	0														
14	1	189	60	23	846	30.1	0.398	59	1														
15	5	166	72	19	175	25.8	0.587	51	1														
16	7	100	0	0	0	30	0.484	32	1														
17	0	118	84	47	230	45.8	0.551	31	1														
18	7	107	74	0	0	29.6	0.254	31	1														
19	1	103	30	38	83	43.3	0.183	33	0														
20	1	115	70	30	96	34.6	0.529	32	1														
21	3	126	88	41	235	39.3	0.704	27	0														
22	8	99	84	0	0	35.4	0.388	50	0														
23	7	196	90	0	0	39.8	0.451	41	1														
24	9	119	80	35	0	29	0.263	29	1														
25	11	143	94	33	146	36.6	0.254	51	1														
26	10	125	70	26	115	31.1	0.205	41	1														
27	7	147	76	0	0	39.4	0.257	43	1														
28	1	97	66	15	140	23.2	0.487	22	0														
29	13	145	82	19	110	22.2	0.245	57	0														
30	5	117	92	0	0	34.1	0.337	38	0														
31	5	109	75	26	0	36	0.546	60	0														
32	3	158	76	36	245	31.6	0.851	28	1														

train.py - E:\DeepLearning\8\8\train.py (3.7.8)

File Edit Format Run Options Window Help

```
'''
   1. Number of times pregnant
   2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
   3. Diastolic blood pressure (mm Hg)
   4. Triceps skin fold thickness (mm)
   5. 2-Hour serum insulin (mu U/ml)
   6. Body mass index (weight in kg/(height in m)^2)
   7. Diabetes pedigree function
   8. Age (years)
   9. Class variable (0 or 1)
'''

from numpy import loadtxt #handle/load dataset

from keras.models import Sequential #Empty working area
from keras.layers import Dense #Dense layer

dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
x = dataset[:,0:8]
y = dataset[:,8]
print(x)


model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

train.py - E:\DeepLearning\8\8\train.py (3.7.8)

File Edit Format Run Options Window Help

```
from numpy import loadtxt #handle/load dataset

from keras.models import Sequential #Empty working area
from keras.layers import Dense #Dense layer

dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
x = dataset[:,0:8]
y = dataset[:,8]
print(x)


model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x, y, epochs=5, batch_size=10)

_, accuracy = model.evaluate(x, y)
print('Accuracy: %.2f' % (accuracy*100))

model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("model.h5")
print("Saved model to disk")
```

```
WARNING:tensorflow:From C:\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:3376: The name tf.log is dep
recated. Please use tf.math.log instead.

WARNING:tensorflow:From C:\Python37\lib\site-packages\tensorflow\python\ops\nn_impl.py:180: add_dispatch_support.<loca
ls>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add
is deprecated. Please use tf.compat.v1.assign_add instead.

Epoch 1/5
2021-11-25 20:01:07.279672: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that t
his TensorFlow binary was not compiled to use: AVX2
768/768 [==============================] - 0s 431us/step - loss: 1.0111 - acc: 0.6211
Epoch 2/5
768/768 [==============================] - 0s 97us/step - loss: 0.7402 - acc: 0.6419
Epoch 3/5
768/768 [==============================] - 0s 110us/step - loss: 0.7579 - acc: 0.6003
Epoch 4/5
768/768 [==============================] - 0s 103us/step - loss: 0.7002 - acc: 0.6289
Epoch 5/5
768/768 [==============================] - 0s 108us/step - loss: 0.6912 - acc: 0.6393
768/768 [==============================] - 0s 79us/step
Accuracy: 66.80
Saved model to disk

E:\DeepLearning\8\8>
```

test.py - E:\DeepLearning\8\8\test.py (3.7.8)

File Edit Format Run Options Window Help

```
from numpy import loadtxt
from keras.models import model_from_json

dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
x = dataset[:,0:8]
y = dataset[:,8]

json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()

model = model_from_json(loaded_model_json)
model.load_weights("model.h5")
print("Loaded model from disk")

predictions = model.predict_classes(x)

for i in range(5,10):
        print('%s => %d (Original Class: %d)' % (x[i].tolist(), predictions[i], y[i]))
```

```
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)t
ype'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
WARNING:tensorflow:From C:\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder
 is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From C:\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uni
form is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From C:\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:174: The name tf.get_default
_session is deprecated. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From C:\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:181: The name tf.ConfigProto
 is deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From C:\Python37\lib\site-packages\keras\backend\tensorflow_backend.py:186: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

2021-11-25 20:05:12.993055: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that t
his TensorFlow binary was not compiled to use: AVX2
Loaded model from disk
[5.0, 116.0, 74.0, 0.0, 0.0, 25.6, 0.201, 30.0] => 0 (Original Class: 0)
[3.0, 78.0, 50.0, 32.0, 88.0, 31.0, 0.248, 26.0] => 0 (Original Class: 1)
[10.0, 115.0, 0.0, 0.0, 0.0, 35.3, 0.134, 29.0] => 1 (Original Class: 0)
[2.0, 197.0, 70.0, 45.0, 543.0, 30.5, 0.158, 53.0] => 0 (Original Class: 1)
[8.0, 125.0, 96.0, 0.0, 0.0, 0.0, 0.232, 54.0] => 1 (Original Class: 1)

E:\DeepLearning\8\8>
```

Deep Learning Terminology - 1

Multilayer Perceptron

- An entry point towards complex neural nets where input data travels through various layers of artificial neurons
- Every single node is connected to all neurons in the next layer which makes it a fully connected neural network
- Input and output layers are present having multiple hidden Layers
- It has a bi-directional propagation i.e. forward propagation and backward propagation

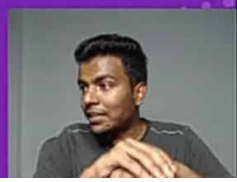
- Used for deep learning [due to the presence of dense fully connected layers and back propagation]

Deep Learning Terminology - 2

Different Optimizers in Neural Network

- Gradient Descent
- Stochastic Gradient Descent
- Mini-Batch Gradient Descent
- Momentum
- Nesterov Accelerated Gradient
- Adagrad
- AdaDelta
- Adam

Adam is the best optimizers. If one wants to train the neural network in less time and more efficiently than Adam is the optimize



Deep Learning Terminology - 3

Vanishing & Exploding Gradient

- It is very common problem in every Neural Network, which is associated with Backpropagation.
- Weights of network are updated through backpropagation by finding gradients.
- When the number of hidden layer is high, then the gradient vanishes or explodes as it propagates backward. It leads instability in network, unable to learn from training
- The explosion occurs through exponential growth by repeatedly multiplying gradients through the network layers that have values larger than 1.0