آکادمی
یوتک

Machine Learning Course

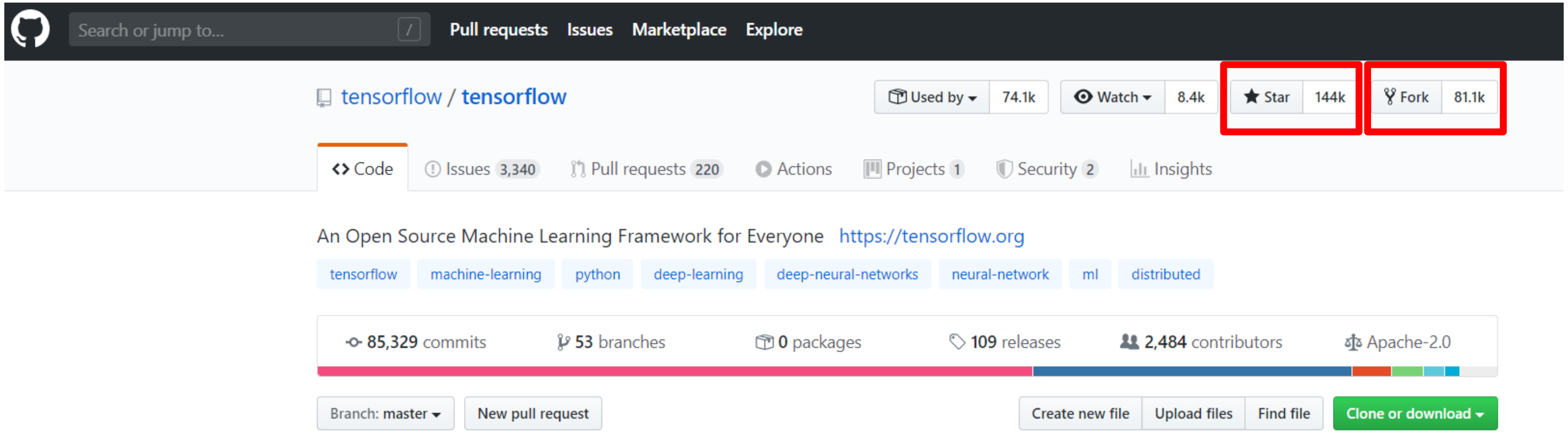
Vahid Reza Khazaie

K Keras

&

TensorFlow

TensorFlow



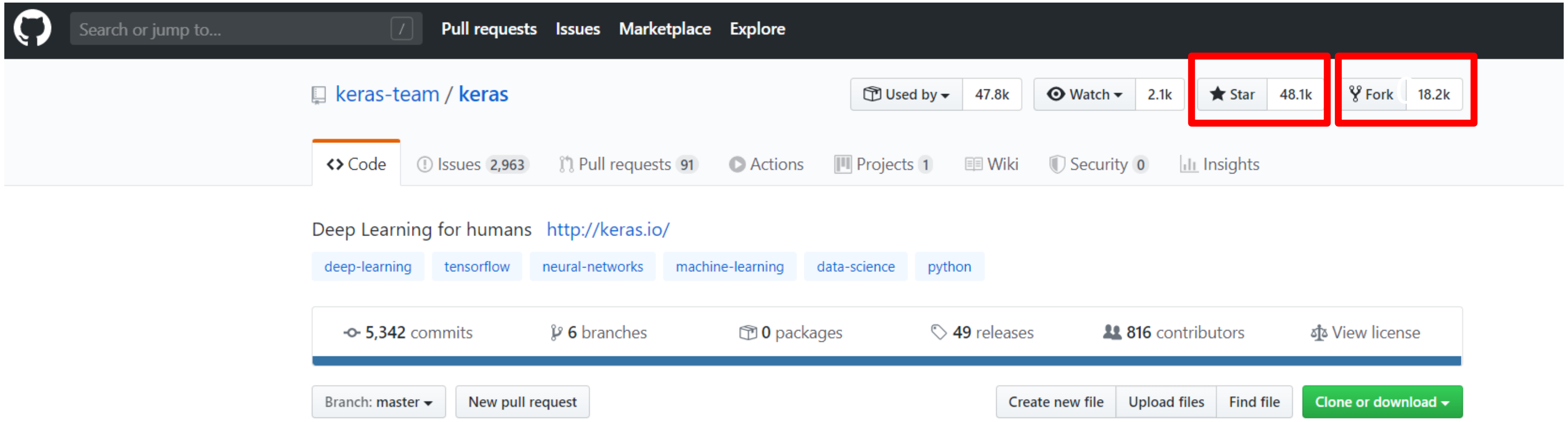
tensorflow / tensorflow

Used by 74.1k | Watch 8.4k | Star 144k | Fork 81.1k

An Open Source Machine Learning Framework for Everyone https://tensorflow.org

85,329 commits | 53 branches | 0 packages | 109 releases | 2,484 contributors | Apache-2.0

Keras



keras-team / keras

Deep Learning for humans http://keras.io/

deep-learning tensorflow neural-networks machine-learning data-science python

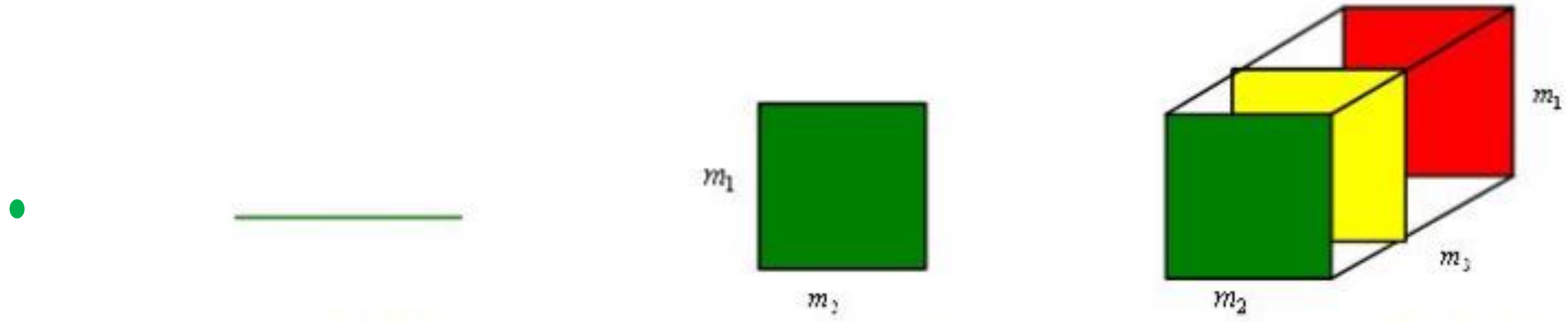
5,342 commits 6 branches 0 packages 49 releases 816 contributors View license

Deep Learning Library

Name	release	Star	Fork
Tensorflow	Nov 1, 2015	111k	68k
caffe	Sep 8, 2013	25k	15k
keras	Mar 22, 2015	34k	13k
mxnet	Apr 26, 2015	15k	5k
pytorch	Jan 22, 2012	19k	4.5k
theano	Jan 6, 2008	8.5k	2.5k

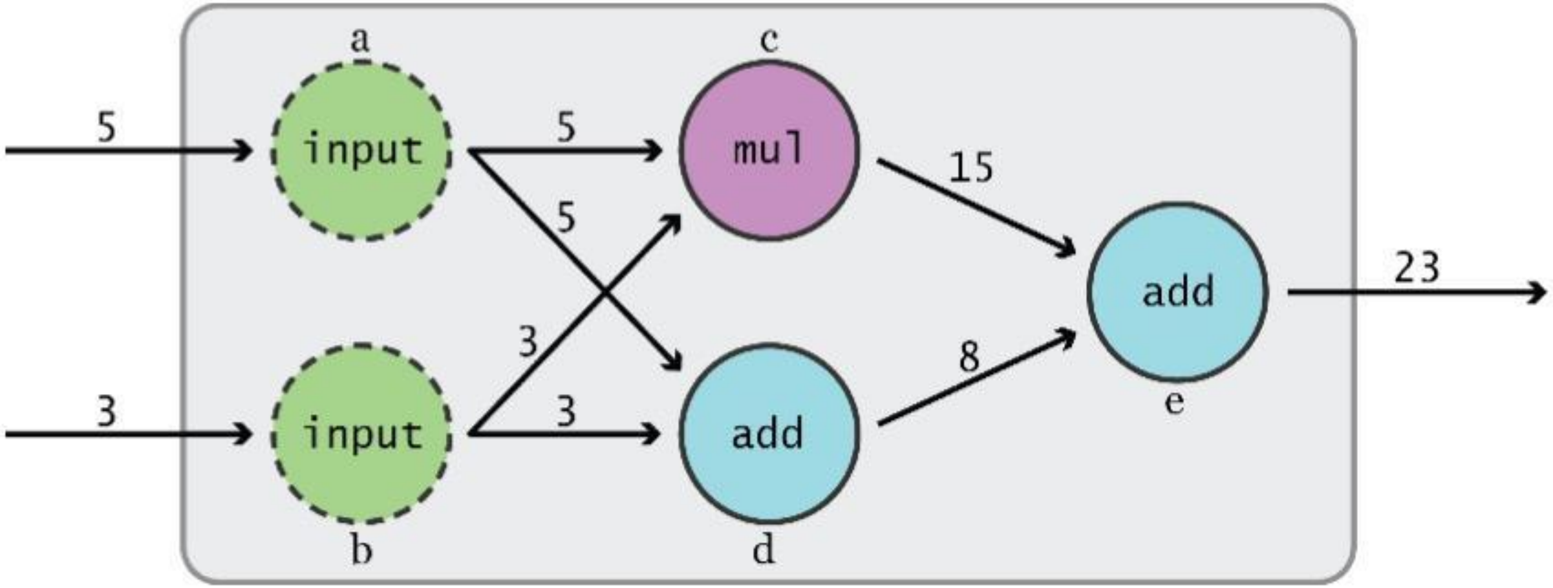
TensorFlow

- What is Tensor?



TensorFlow

- Graph Computation



TensorFlow

- An Example: Persian Handwritten Digit Classification

```
import tensorflow as tf

X = tf.placeholder(tf.float32, [None, 5, 5, 1])

W = tf.Variable(tf.zeros([25, 10]))

b = tf.Variable(tf.zeros([10]))

init = tf.initialize_all_variables()
```

TensorFlow

```
# model
Y = tf.nn.softmax(tf.matmul(tf.reshape(X, [-1, 25]),
W) + b)
# placeholder for correct answers
Y_ = tf.placeholder(tf.float32, [None, 10])
# loss function
cross_entropy = -tf.reduce_sum(Y_ * tf.log(Y))
```

TensorFlow

```
# % of correct answers found in batch
is_correct = tf.equal(tf.argmax(Y,1),
tf.argmax(Y_,1))
accuracy = tf.reduce_mean(tf.cast(is_correct,
tf.float32))
```

TensorFlow

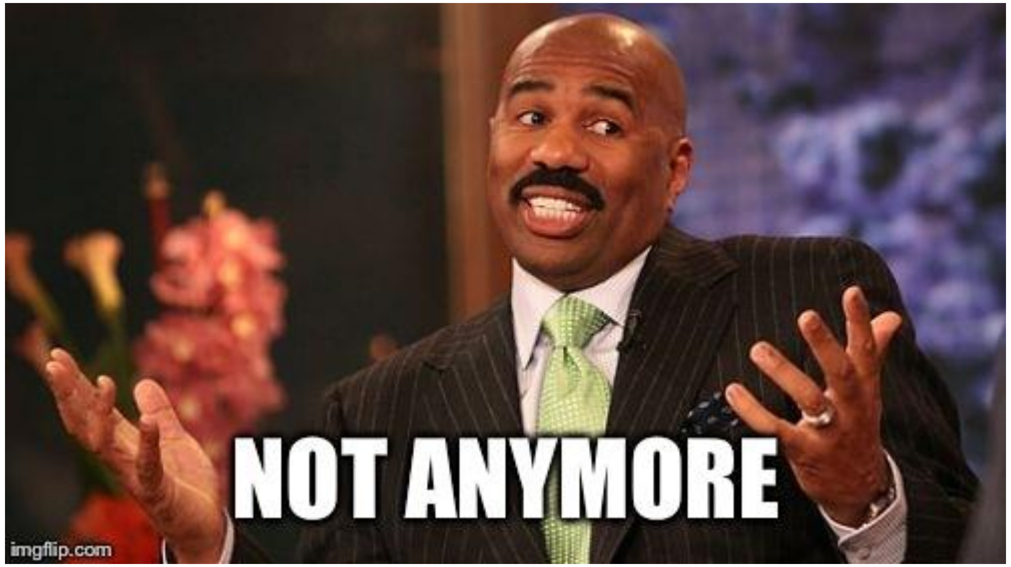
```
optimizer = tf.train.GradientDescentOptimizer(0.003)

train_step = optimizer.minimize(cross_entropy)
```

TensorFlow

```
sess = tf.Session()
sess.run(init)
for i in range(1000):
       # load batch of images and correct answers
       batch_X, batch_Y = mnist.train.next_batch(100)
       train_data={X: batch_X, Y_: batch_Y}
       # train
       sess.run(train_step, feed_dict=train_data)
       # success ?
       a,c = sess.run([accuracy, cross_entropy],
feed_dict=train_data)
       # success on test data ?
       test_data={X: mnist.test.images, Y_:
mnist.test.labels}
       a,c = sess.run([accuracy, cross_entropy, It],
feed=test_data)
```

TensorFlow

You don’t have to do this…



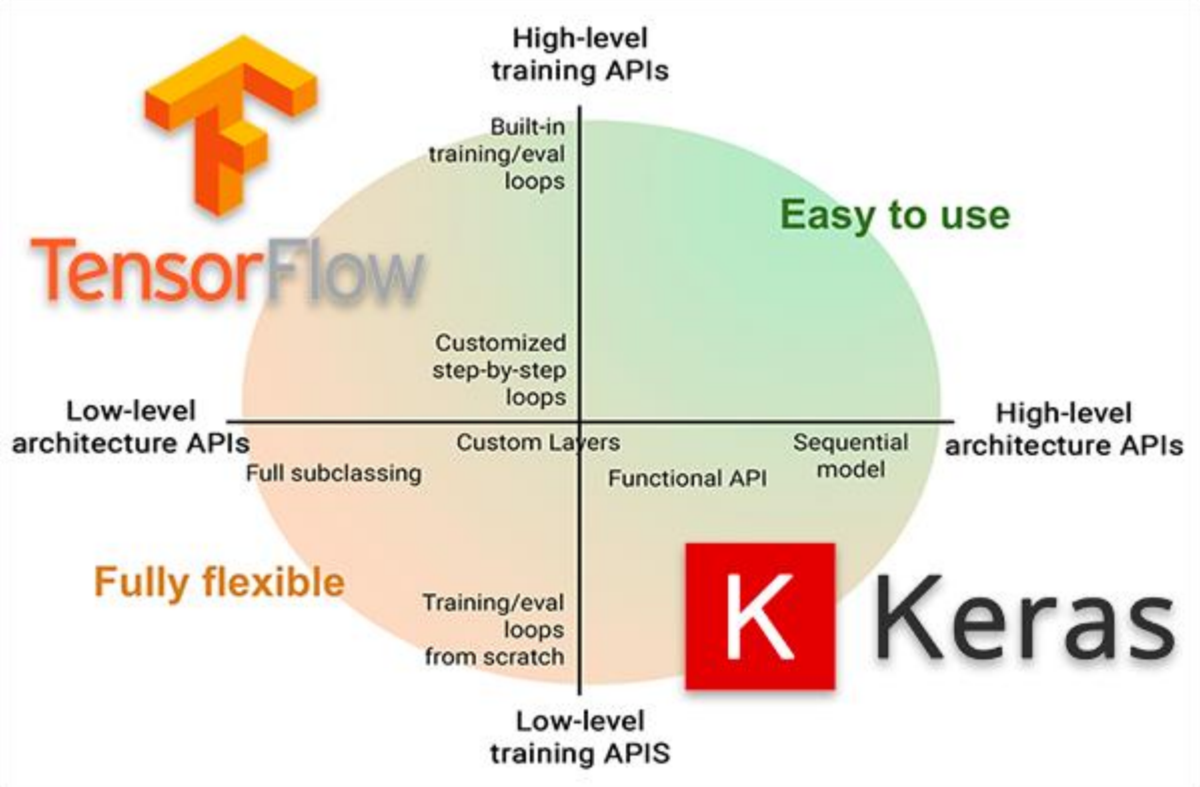
Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow], [CNTK], or [Theano]. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

TensorFlow 2.0



TensorFlow 2.0

- TensorFlow 2.0 and the tf.keras package.

from keras... import ...

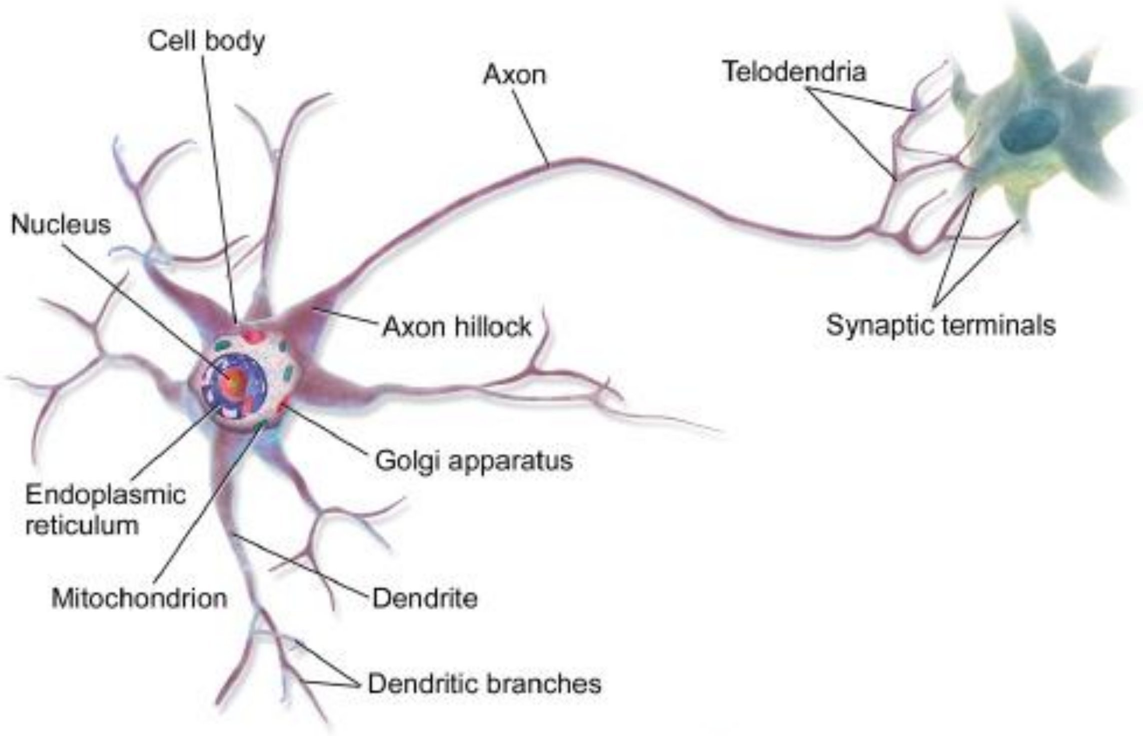
from tensorflow.keras... import ...

Review: MLP

Introduction to Artificial Neural Networks

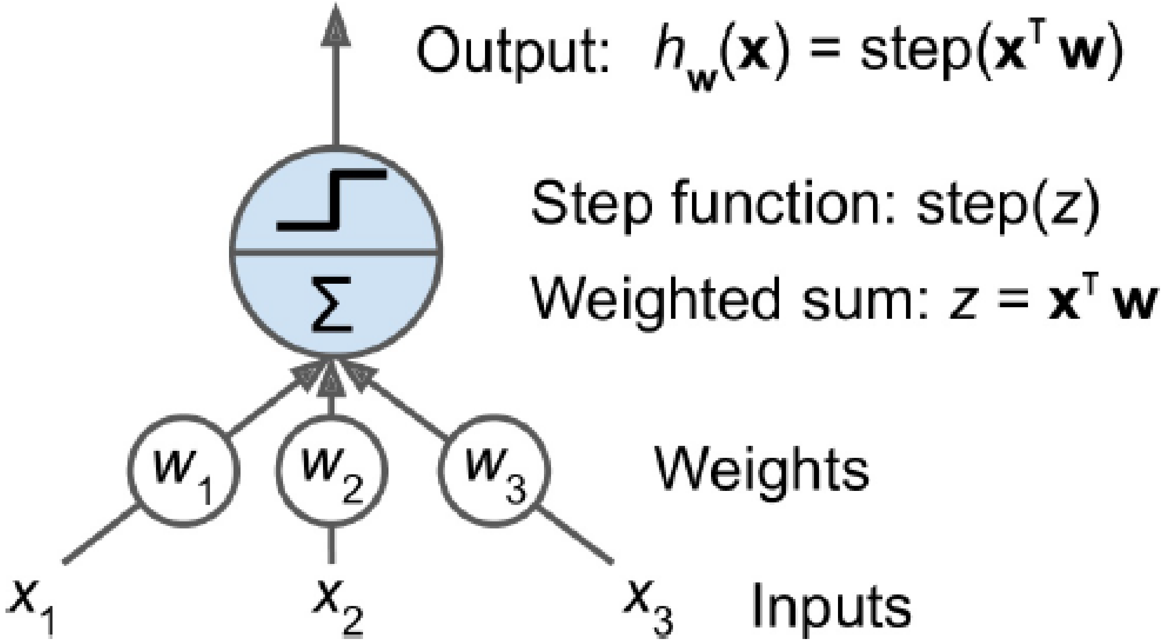
Review: MLP

Biological Neuron



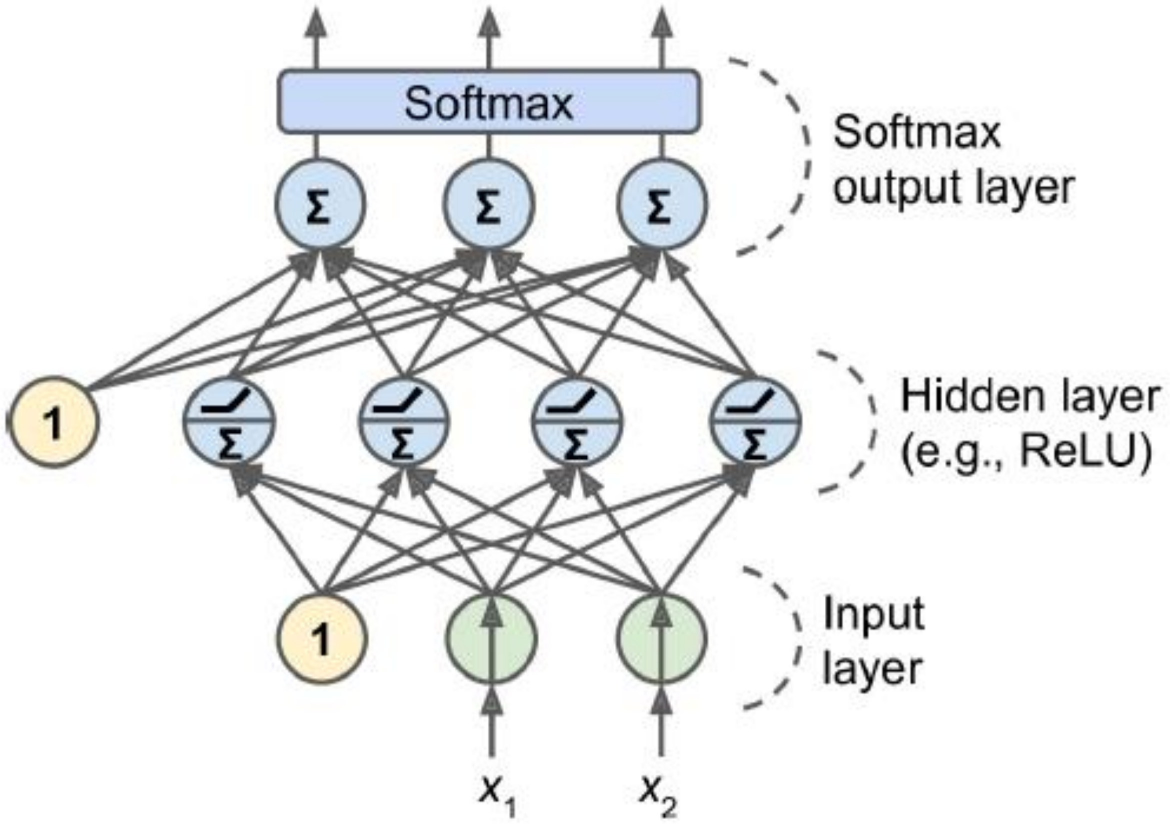
Review: MLP

Perceptron


Output: $h_{\mathbf{w}}(\mathbf{x}) = \text{step}(\mathbf{x}^T\mathbf{w})$
Step function: step(z)
Σ
Weighted sum: $z = \mathbf{x}^T\mathbf{w}$
w_1
w_2
w_3
Weights
x_1
x_2
x_3
Inputs

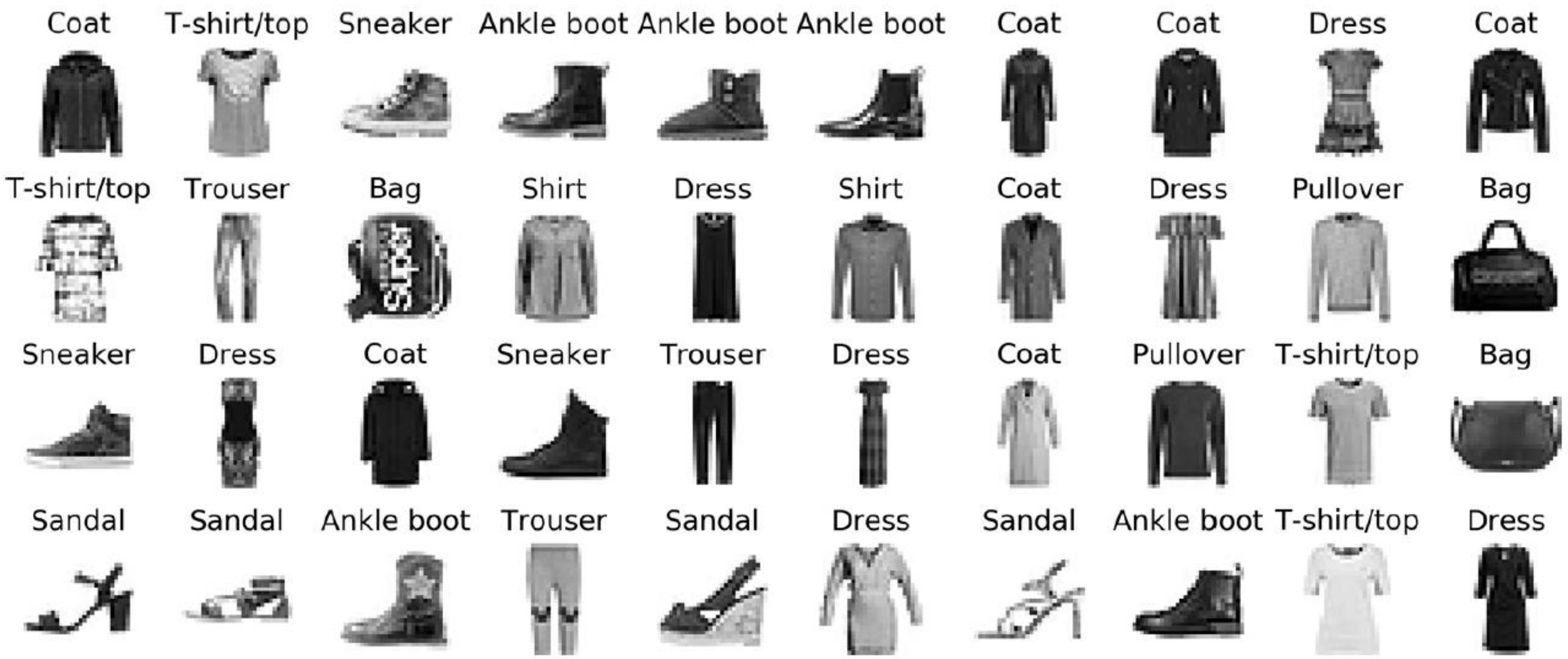
Review: MLP

Multi-layer Perceptron



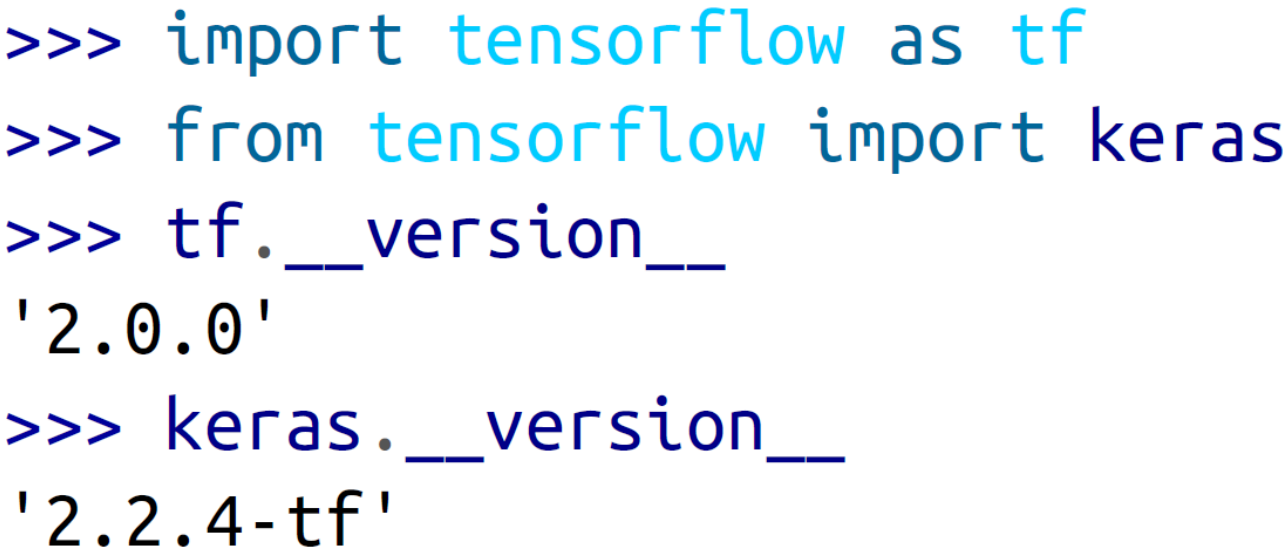
Review: MLP

Let’s classify these clothes…



Review: MLP

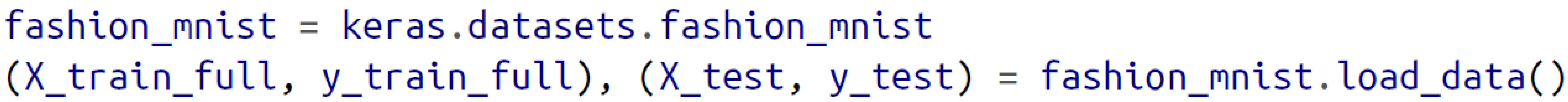
Do some Imports…



```
>>> import tensorflow as tf
>>> from tensorflow import keras
>>> tf.__version__
'2.0.0'
>>> keras.__version__
'2.2.4-tf'
```

Review: MLP

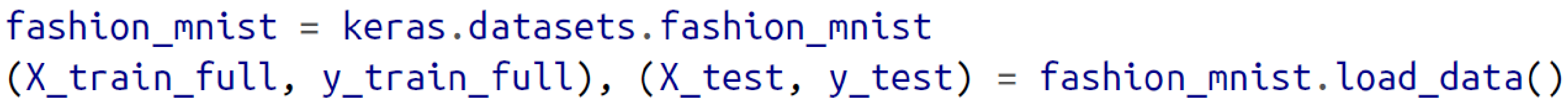
Load Data…



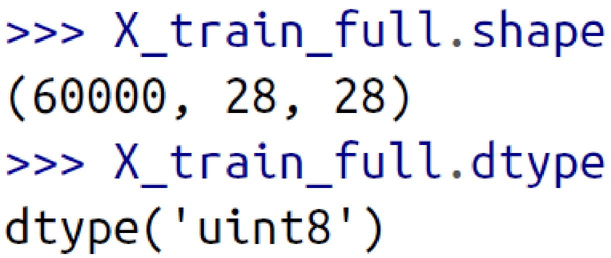
```
fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_data()
```

Review: MLP

See the shape and type of the data…



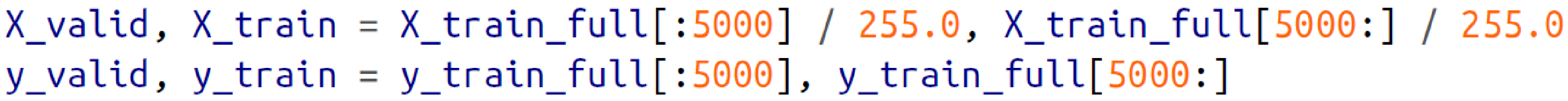
```
fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_data()
```



```
>>> X_train_full.shape
(60000, 28, 28)
>>> X_train_full.dtype
dtype('uint8')
```

Review: MLP

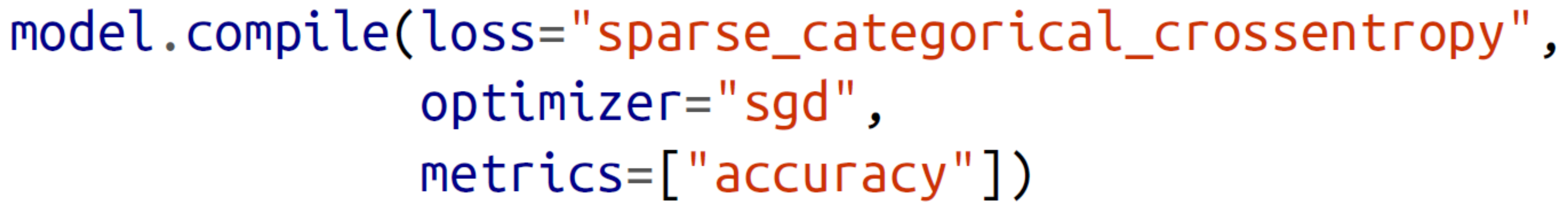
Split and normalize the data…



```
X_valid, X_train = X_train_full[:5000] / 255.0, X_train_full[5000:] / 255.0
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
```

Review: MLP

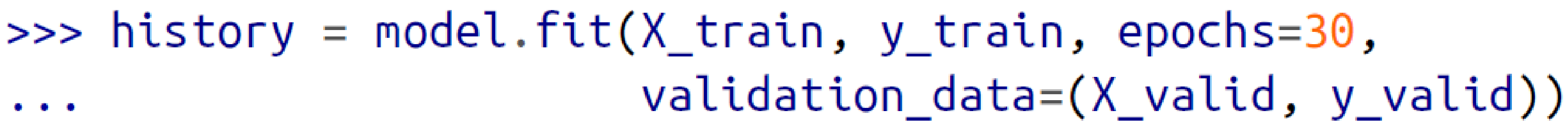
Set loss function, optimizer and metric…



```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

Review: MLP

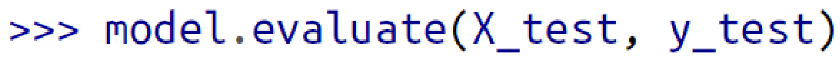
And finally train the model…



```
>>> history = model.fit(X_train, y_train, epochs=30,
...                     validation_data=(X_valid, y_valid))
```

Review: MLP

Mode evaluation…



```
>>> model.evaluate(X_test, y_test)
```