Introduction to Keras

Joe Yeh, M.D.

What is Keras?

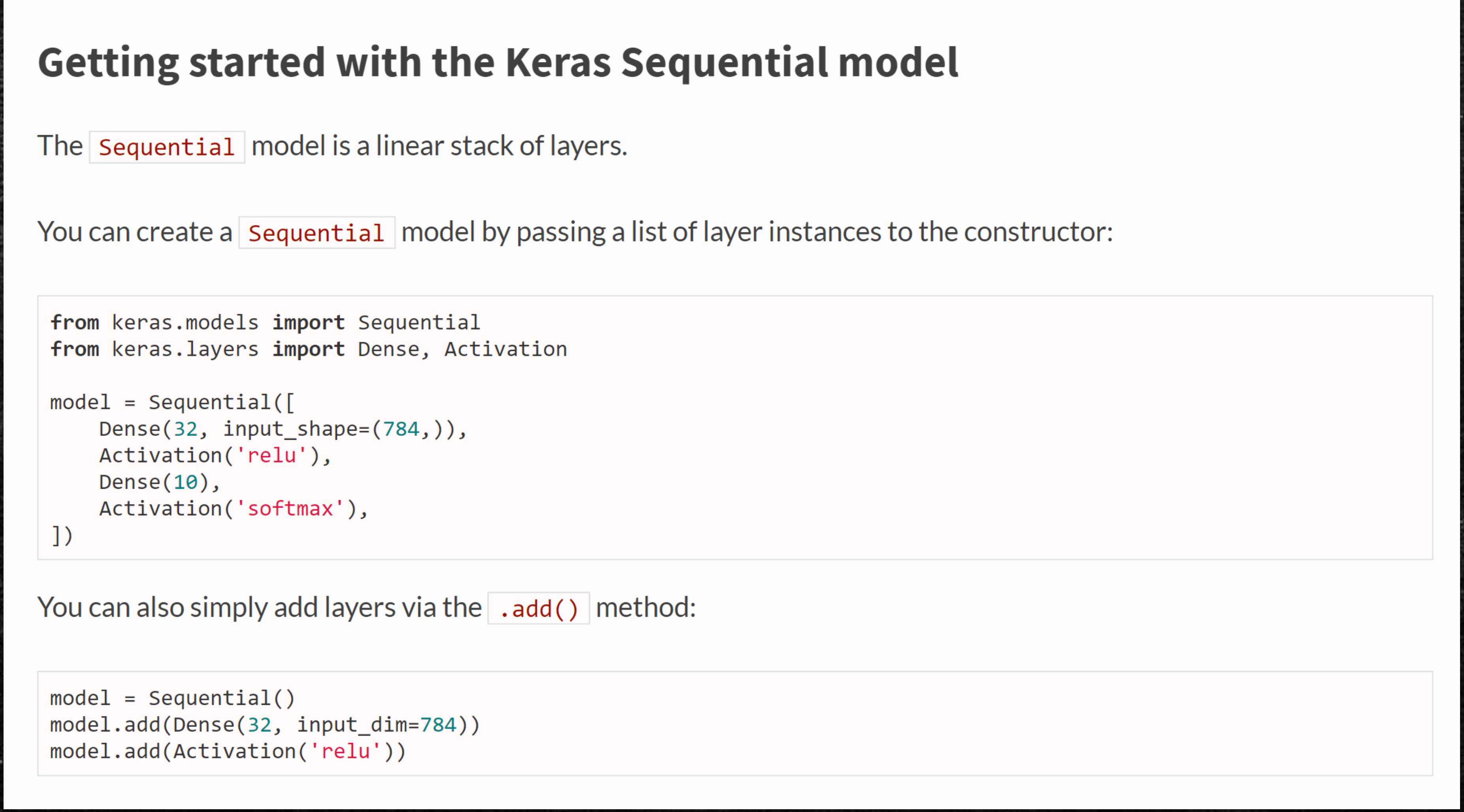
- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

- 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.

Two Modes of Keras

- Sequential
  - Straightforward, for a linear stack of layer
- Functional
  - For more complex, arbitrary computation graph

Sequential Model



Getting started with the Keras Sequential model

The `Sequential` model is a linear stack of layers.

You can create a `Sequential` model by passing a list of layer instances to the constructor:

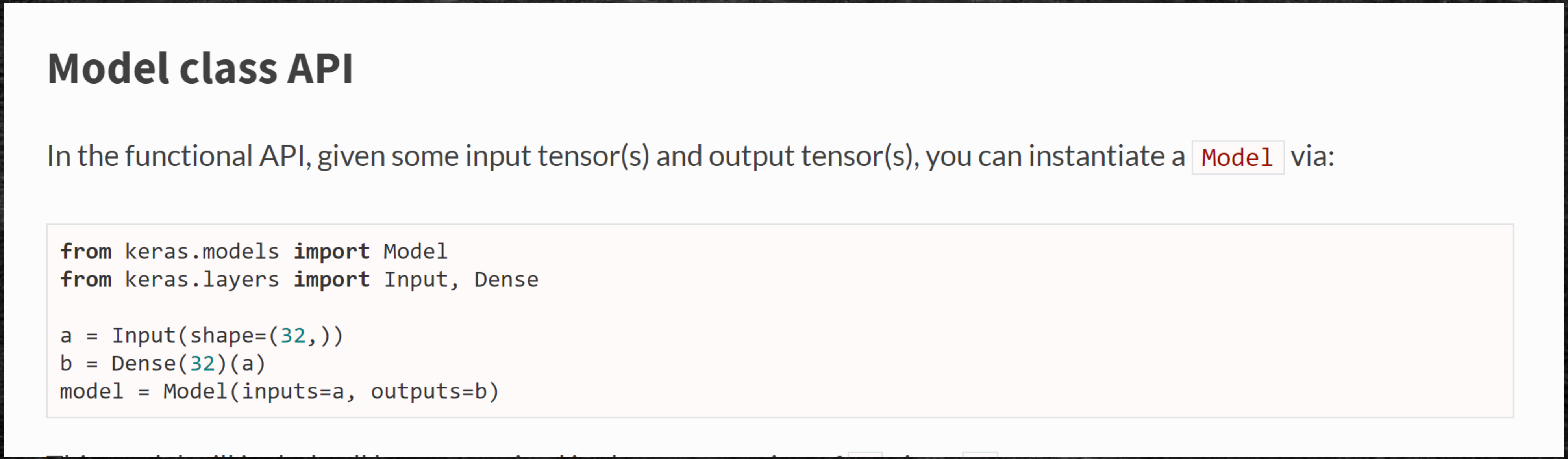
```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

You can also simply add layers via the `.add()` method:

```
model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('relu'))
```

Functional API for Model Construction

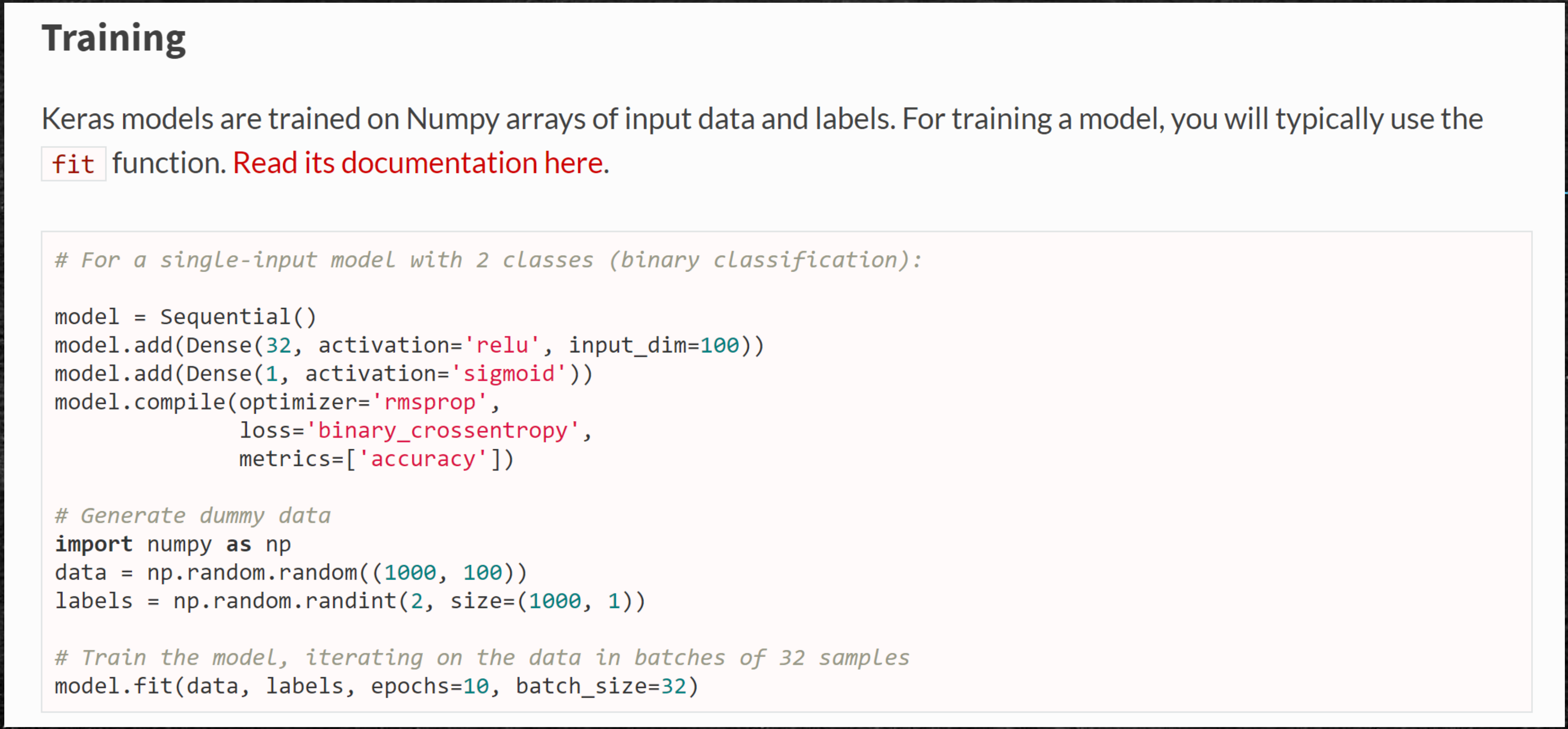


Model class API

In the functional API, given some input tensor(s) and output tensor(s), you can instantiate a `Model` via:

```
from keras.models import Model
from keras.layers import Input, Dense

a = Input(shape=(32,))
b = Dense(32)(a)
model = Model(inputs=a, outputs=b)
```



Training

Keras models are trained on Numpy arrays of input data and labels. For training a model, you will typically use the `fit` function. Read its documentation here.

```
# For a single-input model with 2 classes (binary classification):

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Generate dummy data
import numpy as np
data = np.random.random((1000, 100))
labels = np.random.randint(2, size=(1000, 1))

# Train the model, iterating on the data in batches of 32 samples
model.fit(data, labels, epochs=10, batch_size=32)
```

- After a model is constructed either using sequential model or functional API, the compile, fit and evaluate methods are the same.

Methods of a Model

- Compile : Configure model for training

  compile(self, optimizer, loss=None, metrics=None, loss_weights=None, sample_weight_mode=None, weighted_metrics=None, target_tensors=None)

- Fit : Trains the model for a fixed number of epochs

  fit(self, x=**None**, y=**None**, batch_size=**None**, epochs=1, verbose=1, callbacks=**None**, validation_split=0.0, validation_data=**None**, shuffle=**True**, class_weight=**None**, sample_weight=**None**, initial_epoch=0, steps_per_epoch=**None**, validation_steps=**None**)

- Evaluate : Returns the loss value & metrics values

  evaluate(self, x=**None**, y=**None**, batch_size=**None**, verbose=1, sample_weight=**None**, steps=**None**)