Keras

In this module we will introduce <u>Keras (https://keras.io/)</u>, a high level API for Neural Networks.

To be specific

- we will mostly restrict ourselves to the Keras Sequential model
- this will greatly simplify your learning and coding
- it will restrict the type of Deep Learning programs that you can write
 - but not a meaningful restriction for the simple programs that you will write in this course

After we introduce the high level Keras API

- we will review the history of Deep Learning programming to see how we got here
- this will give you greater insight into what Keras does "under the covers"
 - appreciate history
 - aid your diagnostics

Note:

The code snippets in this notebook are *fragments* of a larger <u>notebook</u> (<u>DNN TensorFlow example.ipynb</u>)

 are illustrative: will not actually execute in this notebook but will in the complete notebook

Confusion warning:

- There are two similar but different packages that implement Keras
 - one built into TensorFlow (the one we will use)
 - a separate project

Later in this module we will explain the difference and why it's important to distinguish between them.

The Keras Sequential Model

Reference: Getting started with the Keras Sequential Model (https://keras.io/getting-started/sequential-model-guide/)

Keras has two programming models

- Sequential
- Functional

We will start with the Sequential model

The Sequential model allows you to build Neural Networks (NN) that are composed of a sequence of layers

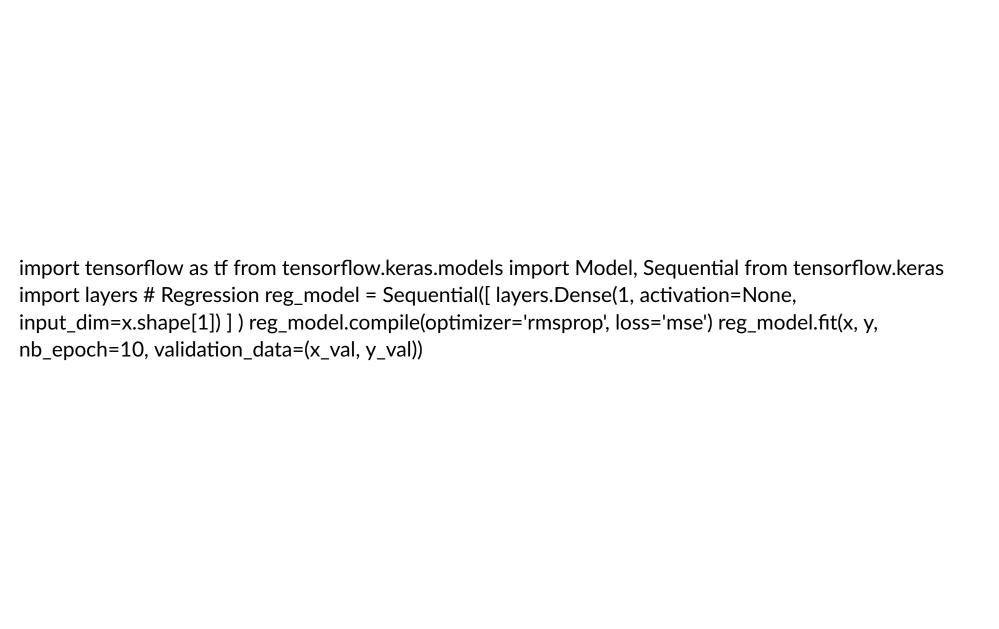
- just like our cartoon
- a very prevalent paradigm

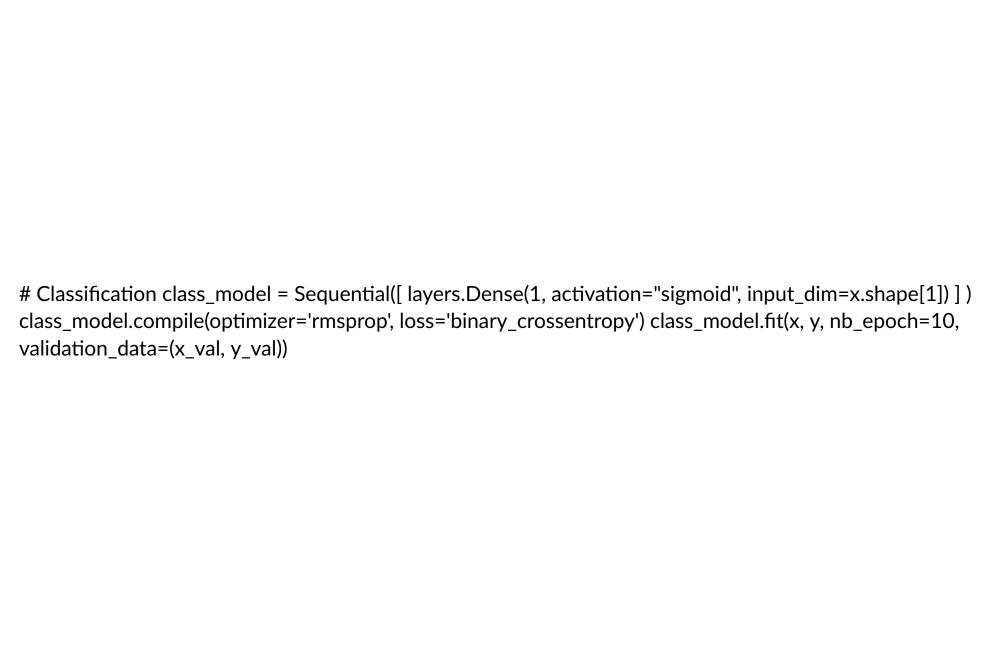
This will likely be sufficient in your initial studies

- but it restricts the architecture of the Neural Networks that you can build
- use the Functional API for full generality
 - but it might appear more complicated

Let's jump into some code.

Some old friends, in new clothing:



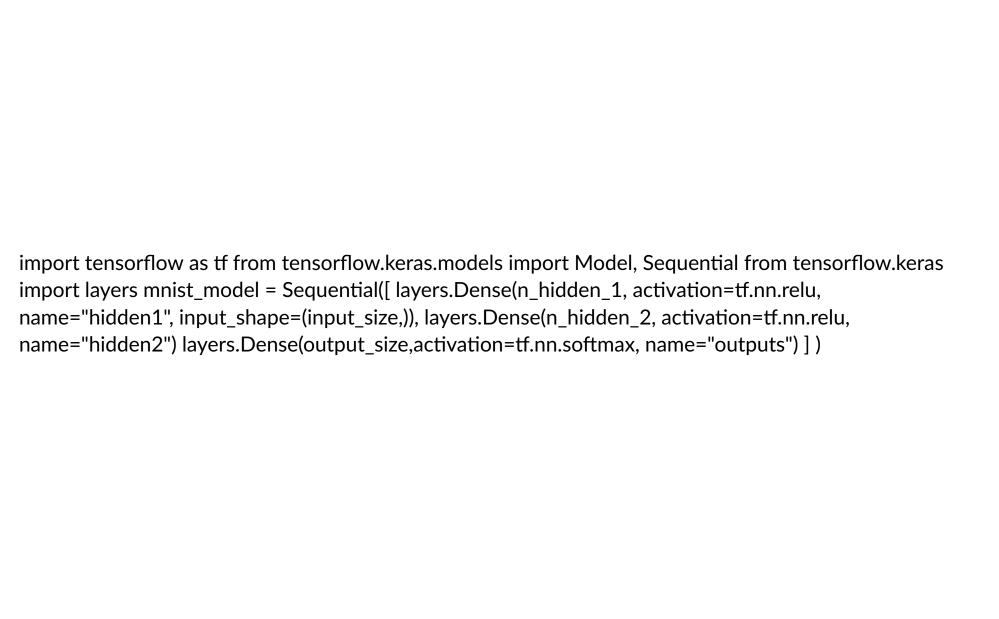


TL;DR

- Both examples are a single layer
 - Dense, with 1 unit ("neuron")
- Regression example
 - No activation
 - MSE cost
- Binary classification example
 - Sigmoid activation
 - Binary cross entropy cost

Hopefully you get the idea.

Let's explore a slightly more complicated model.



This defines a NN with three layers

• we will explain the layers in detail later

To use the model, you first need to "compile" it

metrics = ["acc"] mnist_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=metrics)

"Compiling" is quite significant as we will demonstrate later

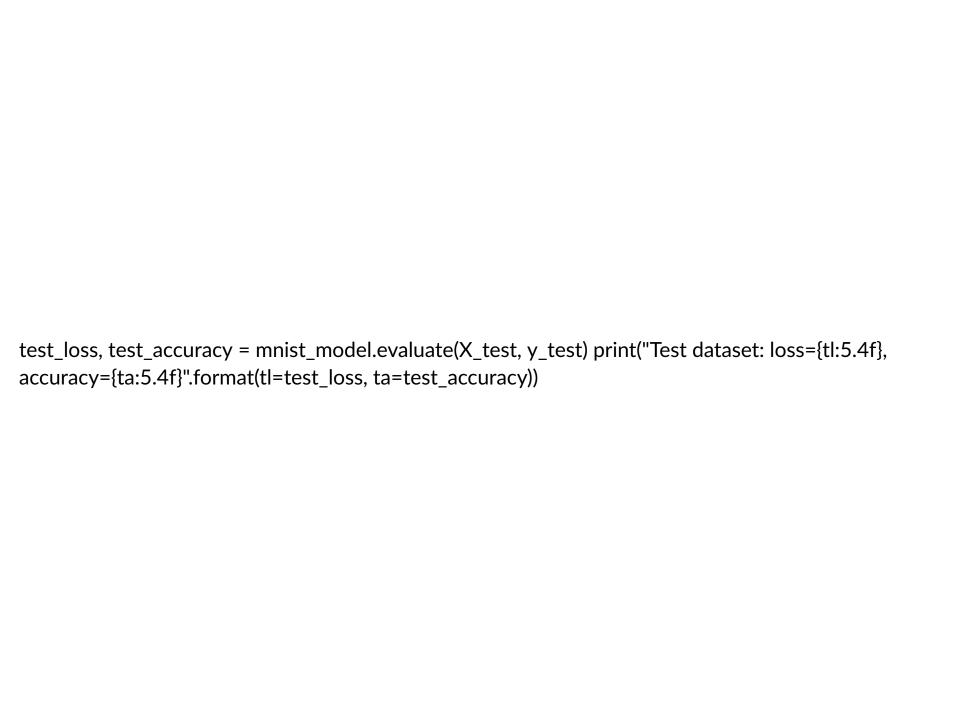
• For now: it is where you define the Cost/Loss function

Next, just as in sklearn: you "fit" the model to the training data.

history = mnist_model.fit(X_train, y_train, epochs=n_epochs, batch_size=batch_size, validation_data= (X_valid, y_valid), shuffle=True)

Once the model is fit, you can predict, just like sklearn.

Here we evaluate the model on the Test dataset.



The idea is quite simple

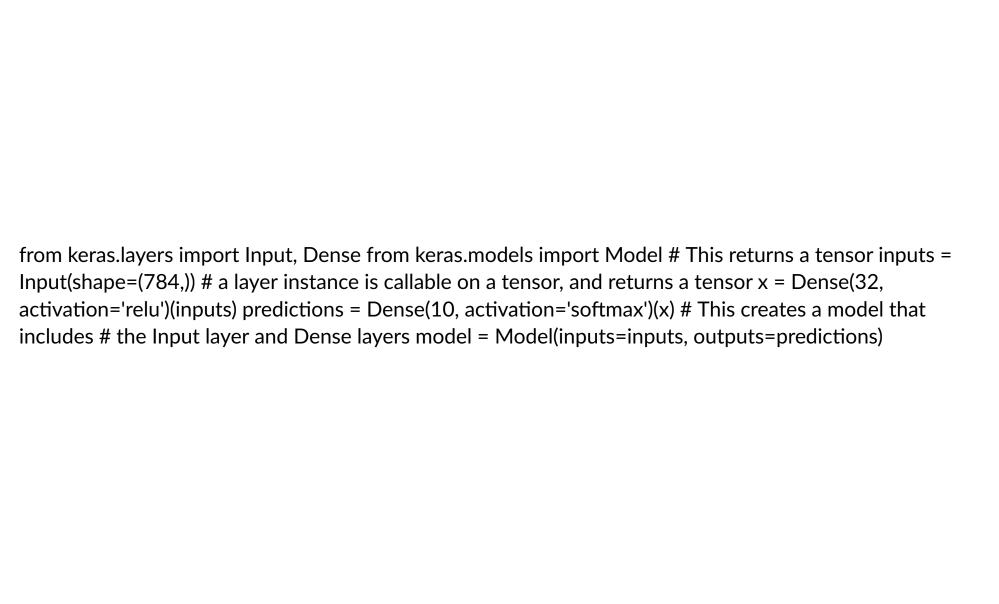
- Keras Sequential implements an sklearn-like API
 - define a model
 - fit the model
 - predict

We have glossed over a lot of details

- What does each layer do?
- Why do we need to "compile"?
 - and why does it need an optimizer?

The Keras Functional Model

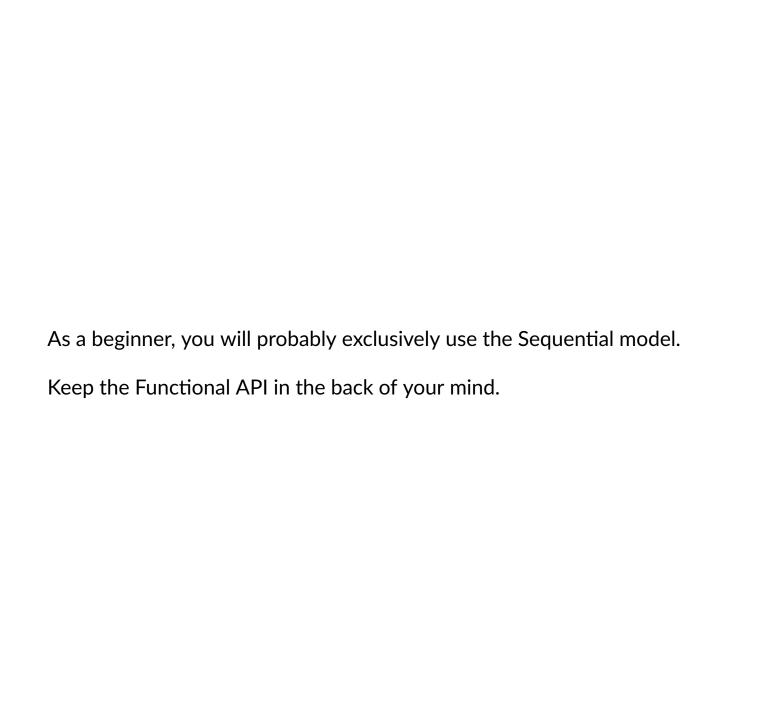
- More verbose than Sequential
- Also more flexible
 - you can define more complex computation graphs (multiple inputs/outputs, shared layers)



Highlights:

- Manually invoke a single layer at a time
 - Passing as input the output of the prior layer.
- You must define an Input layer (placeholder for the input/define its shape)
 - Sequential uses the input_shape= parameter to the first layer
- You "wrap" the graph into a "model" by a Model statement
 - looks like a function definition
 - names the input and output formal parameters
 - a Model acts just like a layer (but with internals that you create)





Let's code!

Lets see a working notebook.

Two options

- Run on your local machine: <u>DNN Tensorflow example Notebook local (DNN Tenso</u>
 - Tensorflow version 2+ only!
- Run on Google Colab: <u>DNN Tensorflow example Notebook from github</u>
 (https://colab.research.google.com/github/nyutandononline/MachineLearning and (Colab)

```
In [1]: print("Done")
```

Done