IntroToNeuralNetworks - IntroToTensorflow - 1

One should look for what is and not what he thinks should be. (Albert Einstein)

Intro to TensorFlow: Topic introduction

In this part of the course, we will cover the following concepts:

- Overview of TensorFlow / Keras building blocks
- Implement and fit a neural network model using Tensorflow on train data
- Evaluate neural network model on test data

Module completion checklist

Objective	Complete
Introduce TensorFlow and Keras	
Define a neural network model using Tensorflow	

Warm up: What is TensorFlow

- Have you heard of TensorFlow, an open source platform for machine learning?
- Watch the video ***TensorFlow in 100 seconds*** to get started and know more about TensorFlow
- After watching the video, share your thoughts on the following questions in the chat:
  - What feature was the most interesting?
  - What application was the most surprising?

TensorFlow

- **TensorFlow** is a base production-grade, **deep learning framework** for distributed processing and data flow graphs modeled on tensor data objects (originally from Google)
- Its core is in C++ with a layer of Python around it, abstracting processes into graphs and giving the core instructions
- ***Click here*** to get more information about TensorFlow

Keras

- Keras is a more high-level wrapper around TensorFlow, making it easier to build models that resemble other Python & scikit-learn functions you are used to
- It is a **deep learning library** explicitly built for Python
- It's a few layers up the abstraction chain from the core in C++, but it allows us to train models efficiently
- ***Click here*** to get more information about Keras
- **Note:** Since TensorFlow 2.0 fully integrated Keras, there is no need to install Keras separately

TensorFlow and Keras

- **TensorFlow** and **Keras** do **NOT** have to go together!
- Keras can be run on top of other systems like CNTK
- You can place your TensorFlow code directly into the Keras training pipeline or model
- If you want to develop your own algorithms, you can do so with TensorFlow

TensorFlow vs. Keras

- Although TensorFlow native modules alone can do everything (and then some) that Keras modules are built for, there are a few advantages of using Keras based on the four guiding principles used by Francois Chollet (the author of Keras):
  - **Modularity**: a model can be understood as a sequence or a graph alone; all the concerns of a deep learning model are discrete components that can be combined in arbitrary ways
  - **Minimalism**: the library provides just enough to achieve an outcome, no frills, and maximizing readability
  - **Extensibility**: new components are easy to add and use within the framework, intended for researchers to try and explore new ideas
  - **Python**: no separate model files with custom file formats as everything is native Python

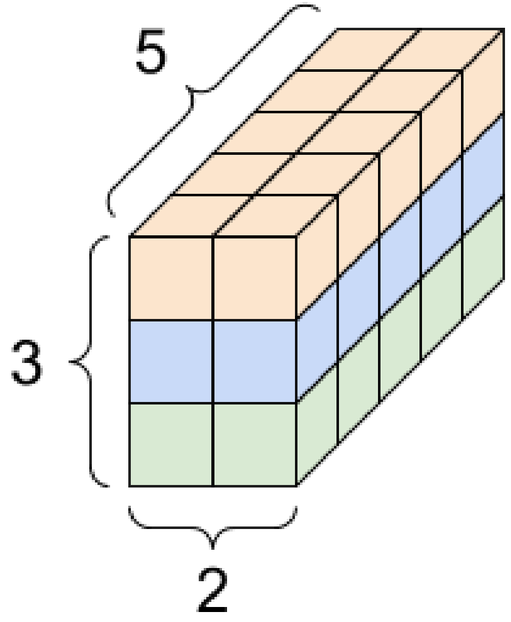
TensorFlow’s APIs

- TensorFlow has APIs that support several languages including:
  - Python
  - JavaScript
  - C++
  - Java
- Additional community-supported languages include C#, Julia, Ruby, Rust, and Scala
- There are several available modules such as Lite and TensorFlow Extended (TFX)
- ***Click here*** to get more information about TensorFlow’s APIs

Tensors

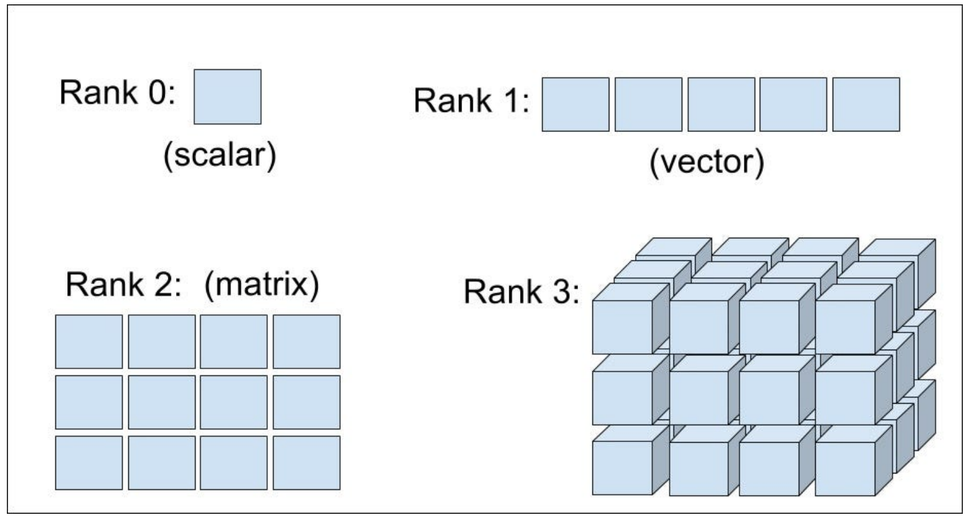
What are tensors?

- **Tensors** are multi-dimensional arrays with a uniform type (called a dtype)
- All tensors are immutable like Python numbers and strings: you can never update the contents of a tensor, only create a new one
- When working with TensorFlow, there is a high probability that you will end up working with tensors at some point



Tensor vocabulary

- Let’s discuss important concepts associated to tensors:
  - **Shape**: The length (number of elements) of each of the dimensions of a tensor
  - **Rank**: Number of tensor dimensions
  - **Axis or Dimension**: A particular dimension of a tensor
  - **Size**: The total number of items in the tensor, the product shape vector



Tensor shape

- You may be familiar with scalars, vectors, matrices, and tensors, but let's discuss how these translate into TensorFlow

Tensor shape	Math type
0	scalar (magnitude)
1	vector (magnitude and direction)
2	matrix
3	3D-tensor (cube)
4+	multi-dimensional tensor (hypercube)

- ***Click here*** to get more information on creating basic tensors

Load TensorFlow for Python

- Loading TF is very simple, just run the following code:

```
# Import tensorflow.
import tensorflow as tf
```

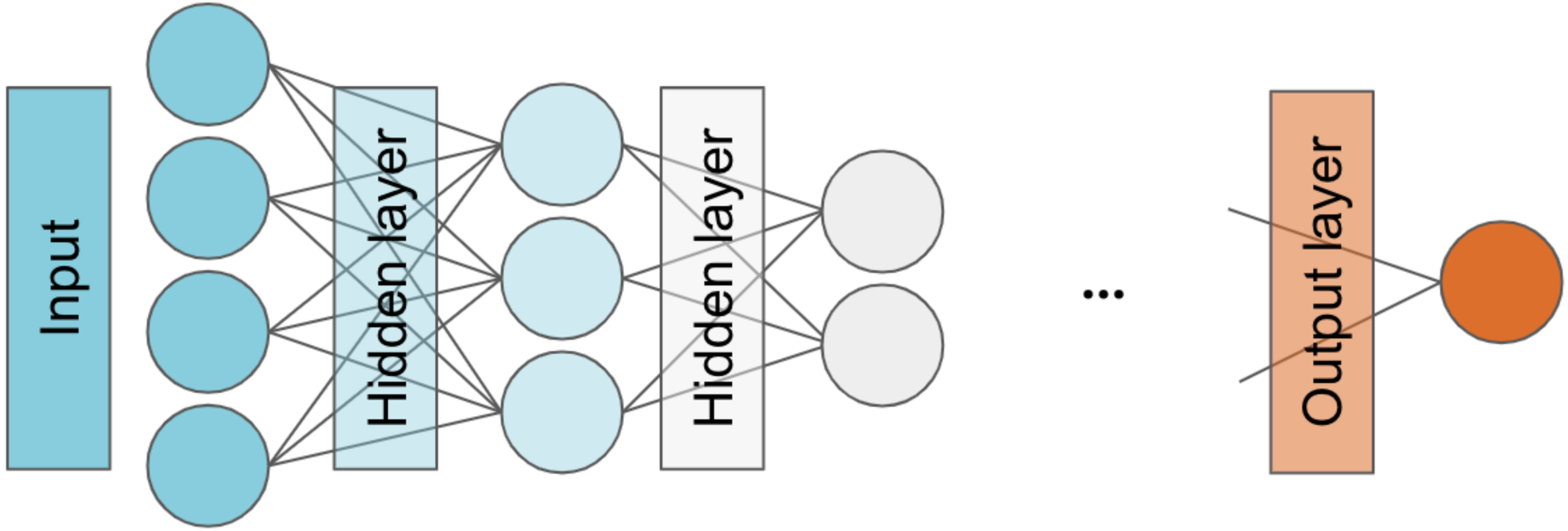
- The `tf` library is the main TensorFlow package that is loaded into the environment

- ***Click here*** to get the list of all modules available through `tf`

Module completion checklist

Objective	Complete
Introduce TensorFlow and Keras	✔
Define a neural network model using Tensorflow	

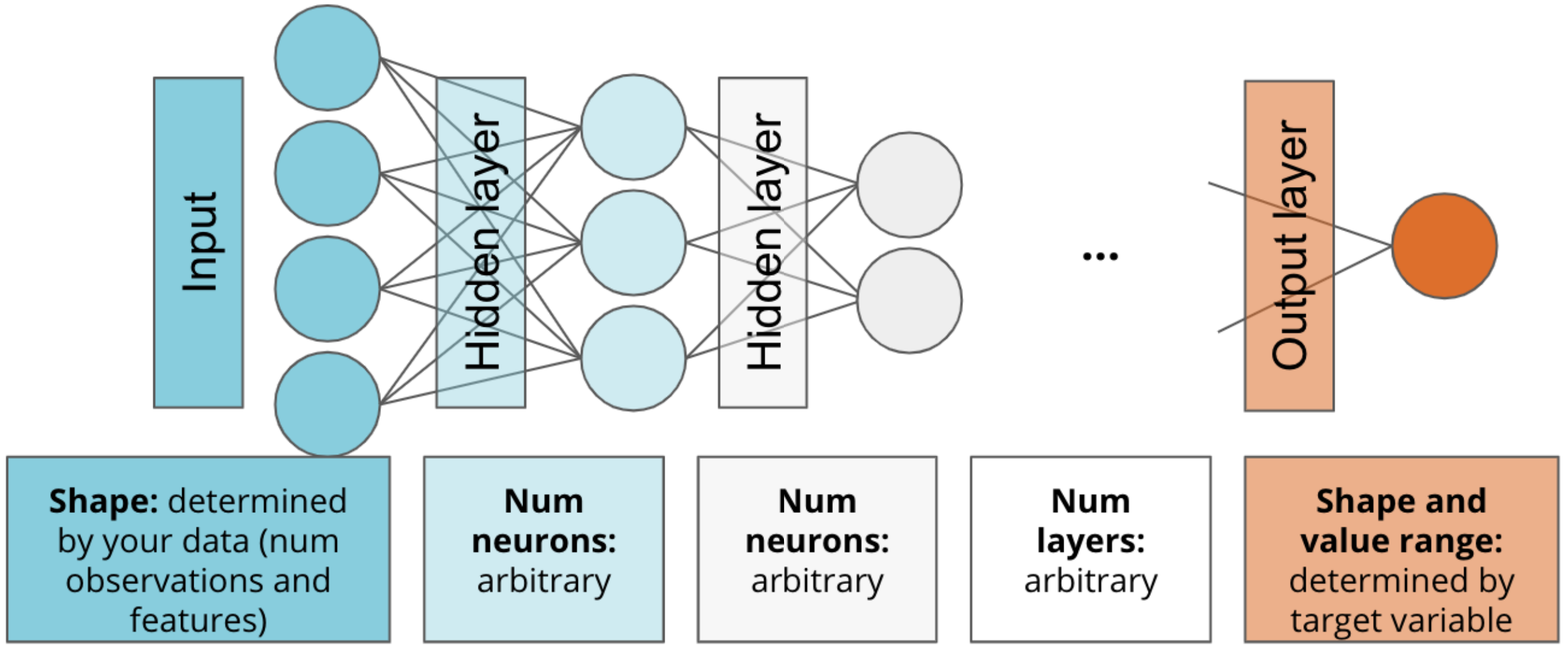
Neural network model: layers



- Any neural network modeling starts with the architecture
- We need 1 **input** and 1 **output** layer with at least 1 **hidden** layer in between
- The number of **hidden layers** determines the **depth of the network**
- It's a trial-and-error process; you must adjust it for every model you build

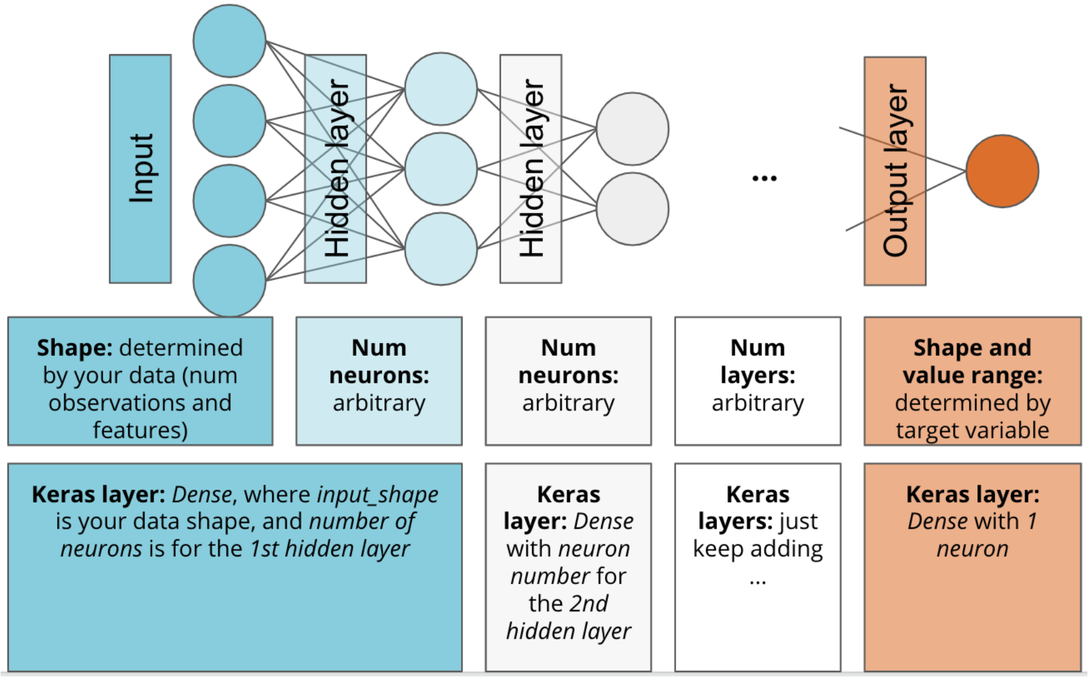
Neural network model: layers and neurons

- The **output** for a binary classification problem will have a **single neuron with a sigmoid** activation function

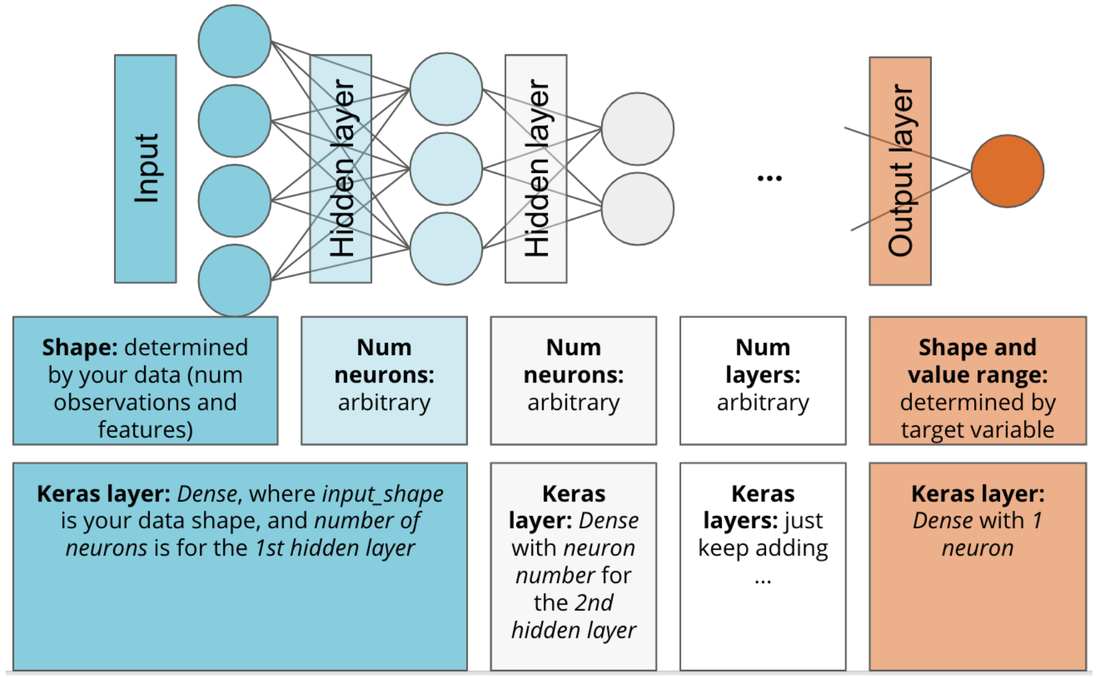


Neural network model: implemented in tf.keras

- Keras model architecture follows this logic:
  - choose model type
  - add your layers one-by-one and make sure to pick your parameters properly

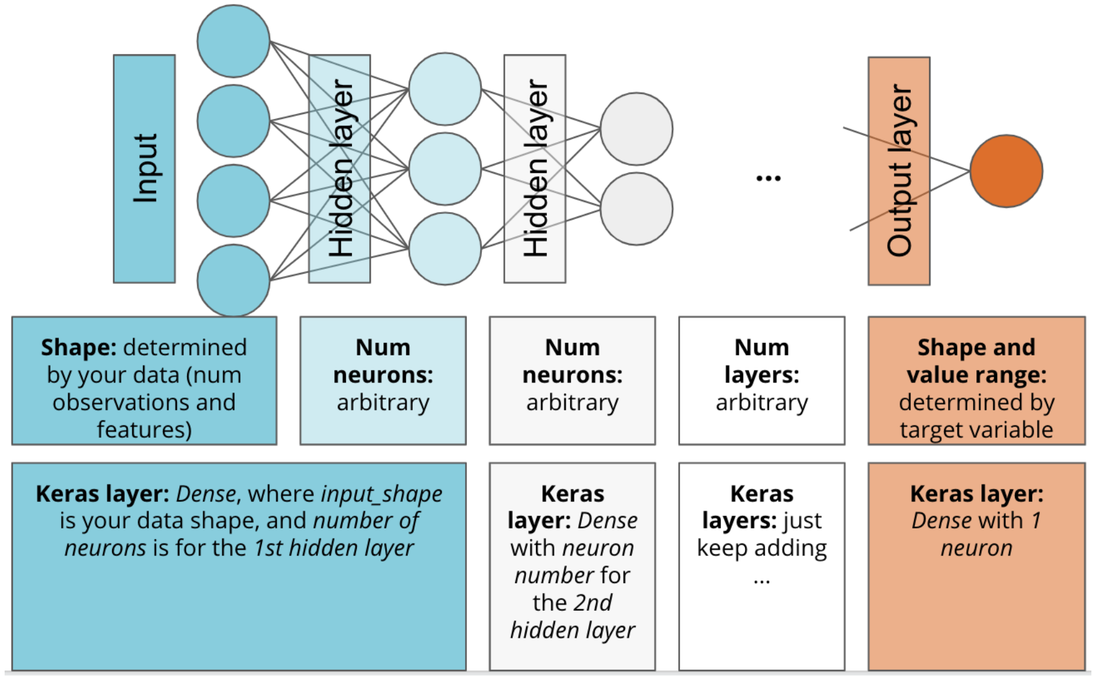


Neural network model: implemented in tf.keras (cont'd)



- There are two main types of models available in Keras:
  - **Sequential model:** A linear stack of layers with each layer having exactly one input and one output tensor
  - **Functional API:** Uses the same layers as the sequential model but provides more flexibility and can handle models with non-linear topology, shared layers, and multiple inputs or outputs

Neural network model: implemented in tf.keras (cont'd)



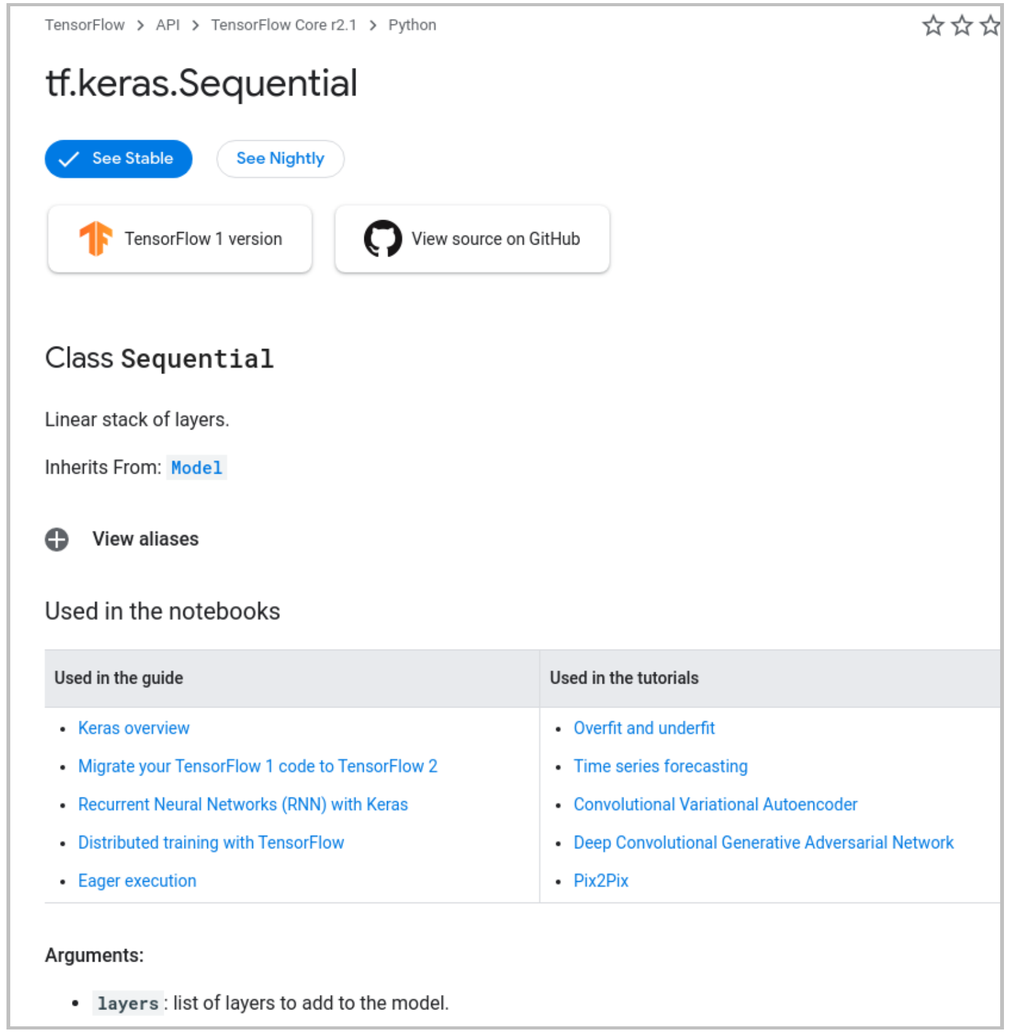
- We will discuss the `Sequential` model first, as we will start by working with one input tensor and one output tensor at each layer
  - Each layer will be `Dense`, i.e., densely connected to the preceding and following layers
  - For each layer we just need to pick appropriate parameters

Keras Sequential model with dense layers

- To create a `Sequential` model, you must pass a **list of layers** to the model constructor

```
Sequential([
  Dense()
  Dense()
  ...
])
```

- ***Click here*** for the package documentation on the TensorFlow website

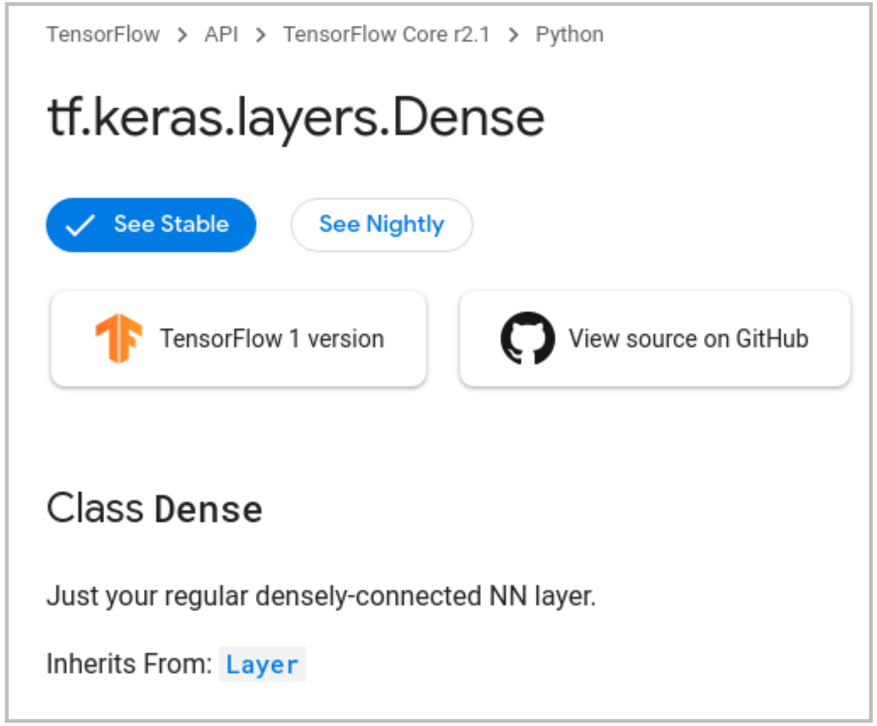


Keras Sequential model with dense layers (cont'd)

- Each `Dense` layer in the list will have the following form

```
Dense(units,                    #<- num neurons
      activation = None,        #<- activation
function
      ...
      )
```

- ***Click here*** for the package documentation on the TensorFlow website



Knowledge check

Module completion checklist

Objective	Complete
Introduce TensorFlow and Keras	✔
Define a neural network model using Tensorflow	✔

Congratulations on completing this module!