

Artificial Neural Networks with Keras

Prof. Gheith Abandah

References:

- *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow* by Aurélien Géron (O'Reilly). 2019, 978-1-492-03264-9.
- *Deep Learning with Python*, by François Chollet, Manning Pub. 2018
- *Introduction to Keras* by Francois Chollet, March 9th, 2018 ([slides](#))

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

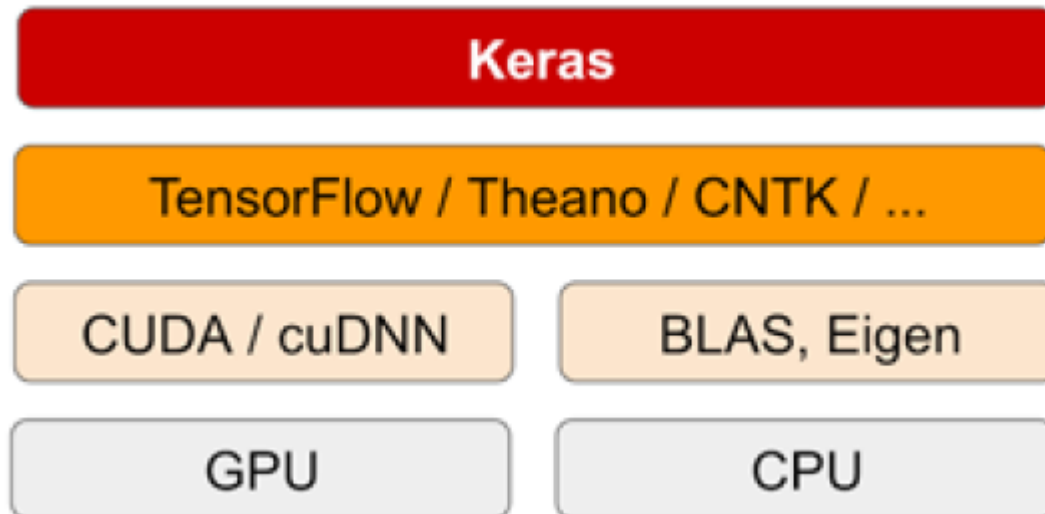
Introduction

- YouTube Video: *Keras Explained* from Siraj Raval

https://youtu.be/j_pJmXJwMLA

1. Introduction

- **Keras** is a high-level API to build and train deep learning models.



1. Introduction – Advantages

- **User friendly**: Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.
- **Modular and composable**: Keras models are made by connecting configurable building blocks together, with few restrictions.
- **Easy to extend**: Write custom building blocks to express new ideas for research. Create new layers, loss functions, and develop state-of-the-art models.

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

2. Keras API Styles

1. The Sequential Model

- Dead simple
- Only for single-input, single-output, sequential layer stacks
- Good for 70+% of use cases

2. The functional API

- Like playing with Lego bricks
- Multi-input, multi-output, arbitrary static graph topologies
- Good for 95% of use cases

3. Model subclassing

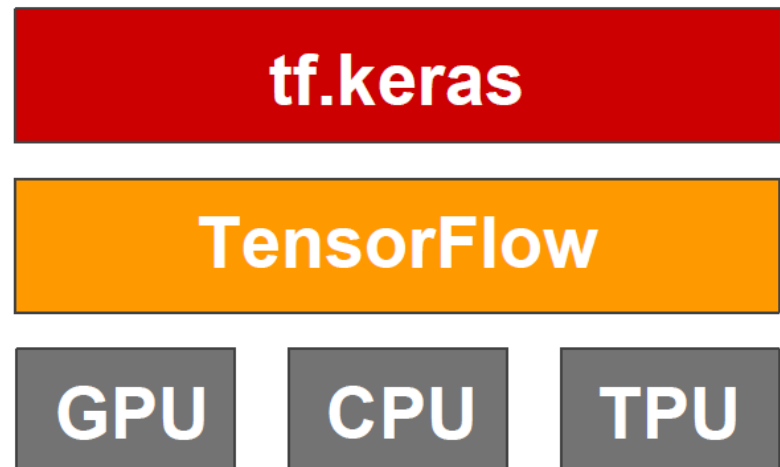
- Maximum flexibility
- Larger potential error surface

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

3. TensorFlow Keras

- Keras is the official high-level API of TensorFlow
- tensorflow.keras (tf.keras) module
- Part of core TensorFlow since v1.4
- Full Keras API
- With useful extra features such as **tf.data**



3. TensorFlow Keras

- To install TensorFlow

```
$ pip install --upgrade tensorflow
```

- To import Keras from TensorFlow

```
>>> import tensorflow as tf
```

```
>>> from tensorflow.keras import layers
```

```
>>> from tensorflow import keras
```

```
>>> tf.__version__  
'2.1.0'
```

```
>>> keras.__version__  
'2.2.4-tf'
```

- Dense
- Activations
- Dropout
- Conv1D, 2D, 3D
- Polling
- RNN, LSTM, GRU
- ...

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

4. Image Classifier Using the Sequential Model

- **Fashion MNIST** is similar to MNIST (70,000 grayscale images of 28×28 pixels each, with 10 classes).



4.1 Get and Prepare the Dataset

```
import tensorflow as tf
from tensorflow import keras

# Get the Fashion MNIST
fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) =
    fashion_mnist.load_data()

# Prepare the data train (55000), val (5000), test (10000)
X_valid = X_train_full[:5000] / 255.
X_train = X_train_full[5000:] / 255.
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
X_test = X_test / 255.
```

4.2 Build the Model

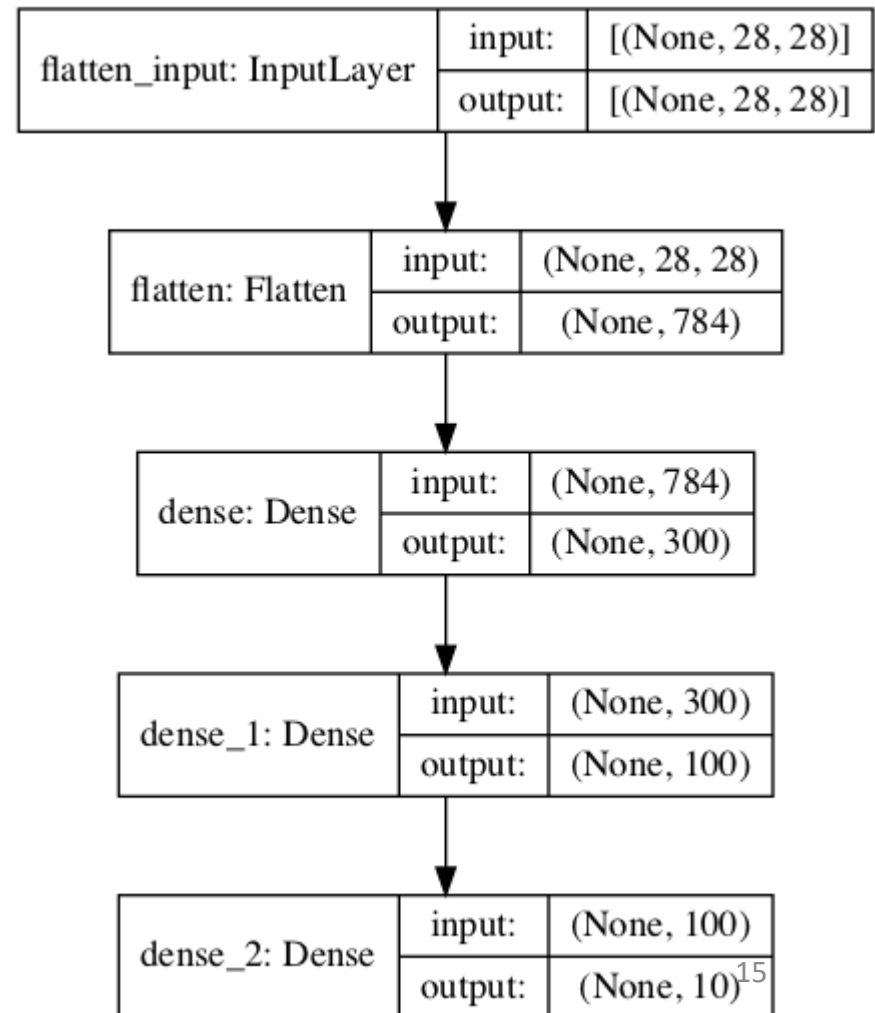
The default is no activation function, i.e., linear layer.

```
model = keras.models.Sequential()  
model.add(keras.layers.Flatten(input_shape=[28, 28]))  
model.add(keras.layers.Dense(300, activation="relu"))  
model.add(keras.layers.Dense(100, activation="relu"))  
model.add(keras.layers.Dense(10, activation="softmax"))
```

```
>>> model.summary()  
-----  
Layer (type)                Output Shape                Param #  
-----  
flatten_1 (Flatten)         (None, 784)                 0  
-----  
dense_3 (Dense)             (None, 300)                235500  
-----  
dense_4 (Dense)             (None, 100)                30100  
-----  
dense_5 (Dense)             (None, 10)                 1010  
-----  
Total params: 266,610  
Trainable params: 266,610  
Non-trainable params: 0
```

4.2 Build the Model

```
# Plot the model
keras.utils.plot_model(
    model,
    "my_model.png",
    show_shapes=True)
```



4.3 Compile the Model

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

Stochastic Gradient
Descent

For sparse labels (0-9):

```
loss = "sparse_categorical_crossentropy"
```

For one-hot labels:

```
loss = "categorical_crossentropy"
```

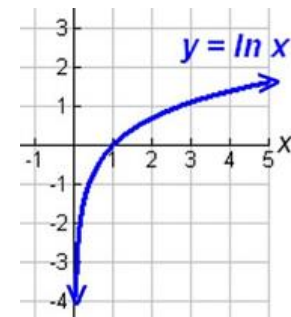
For binary labels:

```
loss = "binary_crossentropy"
```

For regression:

```
loss = "mean_squared_error"
```

$$-\sum_{c=1}^M y_{o,c} \log(p_{o,c})$$



4.4 Train the Model

```
# Train the model
```

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

Train on 55000 samples, validate on 5000 samples

Epoch 1/30

55000/55000 [=====] - 2s 44us/sample - loss:
0.7226 - accuracy: 0.7641 - val_loss: 0.5073 - val_accuracy: 0.8320

Epoch 2/30

55000/55000 [=====] - 2s 39us/sample - loss:
0.4844 - accuracy: 0.8321 - val_loss: 0.4541 - val_accuracy: 0.8478

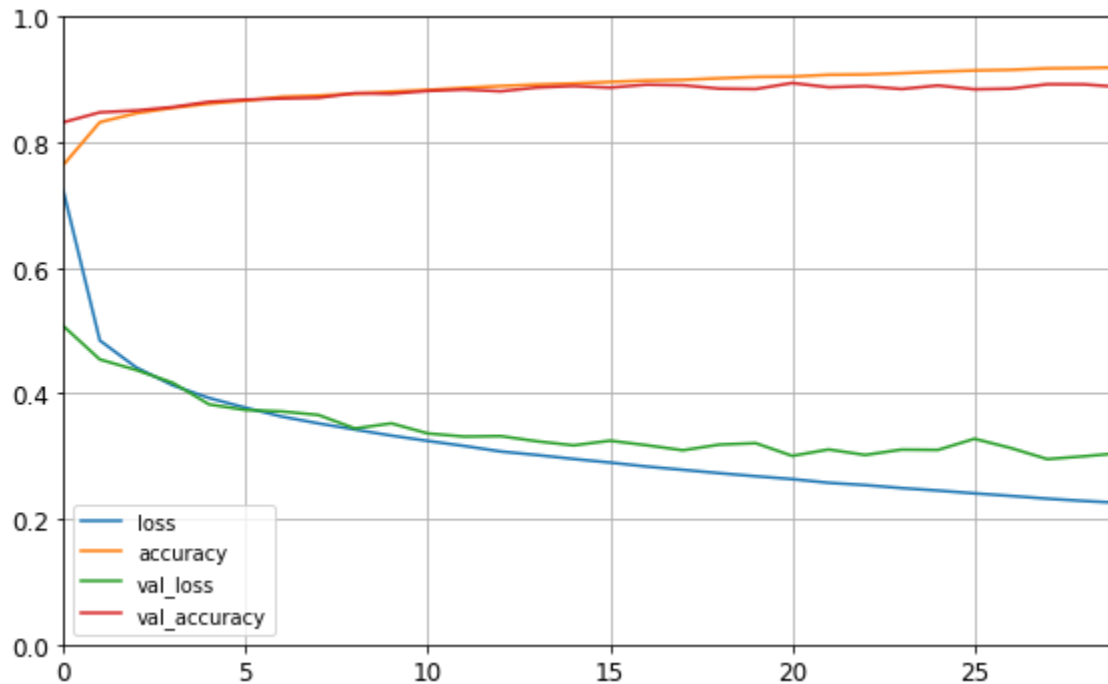
...

Epoch 30/30

55000/55000 [=====] - 2s 39us/sample - loss:
0.2256 - accuracy: 0.9195 - val_loss: 0.3049 - val_accuracy: 0.8882

4.4 Train the Model

```
import pandas as pd
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
save_fig("keras_learning_curves_plot")
plt.show()
```



4.5 Evaluate and Use the Model

```
model.evaluate(X_test, y_test)
10000/10000 [=====] - 0s 21us/sample - loss:
0.3378 - accuracy: 0.8781
[0.33780701770782473, 0.8781]
```

```
X_new = X_test[:3]
y_proba = model.predict(X_new)
y_proba.round(2)
array([[0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.01, 0. , 0.99],
       [0. , 0. , 0.99, 0. , 0.01, 0. , 0. , 0. , 0. , 0. ],
       [0. , 1. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ]],
      dtype=float32)
```

```
model.predict_classes(X_new)
array([9, 2, 1])
```

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

5. Example - MNIST

1. Define your training data: input tensors and target tensors.
2. Define a network of layers (or **model**) that maps your inputs to your targets.
3. Configure the learning process by choosing a loss function, an optimizer, and some metrics to monitor.
4. Iterate on your training data by calling the **fit()** method of your model.

5. Example – Prepare the data

```
from keras.datasets import mnist
(train_images, train_labels), (test_images,
                               test_labels) = mnist.load_data()
#(60000, 28, 28), (60000), #(10000, 28, 28), (10000)
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

from keras.utils import to_categorical    #one hot
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

5. Example – Define and configure the network

```
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu',
                        input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))

network.compile(optimizer='rmsprop',
               loss='categorical_crossentropy',
               metrics=['accuracy'])
```

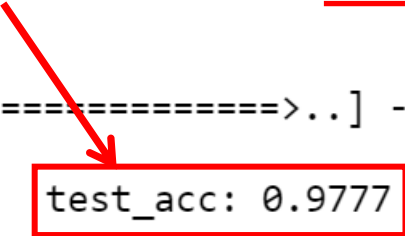
5. Example – Training and evaluation

```
network.fit(train_images, train_labels, epochs=5,  
           batch_size=128)
```

```
Epoch 1/5  
60000/60000 [=====] - 2s - loss: 0.2577 - acc: 0.9245  
Epoch 2/5  
60000/60000 [=====] - 1s - loss: 0.1042 - acc: 0.9690  
Epoch 3/5  
60000/60000 [=====] - 1s - loss: 0.0687 - acc: 0.9793  
Epoch 4/5  
60000/60000 [=====] - 1s - loss: 0.0508 - acc: 0.9848  
Epoch 5/5  
60000/60000 [=====] - 1s - loss: 0.0382 - acc: 0.9890
```

```
test_loss, test_acc = network.evaluate(test_images,  
                                       test_labels)
```

```
9536/10000 [=====>..] - ETA: 0s
```



```
test_acc: 0.9777
```


Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

6. Regression Using the Sequential Model

- Solve the **California housing** problem using a regression neural network.
- Scikit-Learn has `fetch_california_housing()` function to load the data
- This dataset contains only numerical features and there is no missing values.

6.1 Get and Prepare the Dataset

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
housing = fetch_california_housing()
```

The default is 75% : 25%



```
X_train_full, X_test, y_train_full, y_test =
    train_test_split(housing.data, housing.target,
                    random_state=42)
```

```
X_train, X_valid, y_train, y_valid = train_test_split(
    X_train_full, y_train_full, random_state=42)
```

```
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
```

```
X_valid = scaler.transform(X_valid)
```

```
X_test = scaler.transform(X_test)
```

6.2 Build and Compile the Model

```
# Building by passing a list of layers when creating  
# the Sequential model
```

```
model = keras.models.Sequential(  
    keras.layers.Dense(30, activation="relu",  
        input_shape=X_train.shape[1:]),  
    keras.layers.Dense(1)  
])
```

The default is 0.01

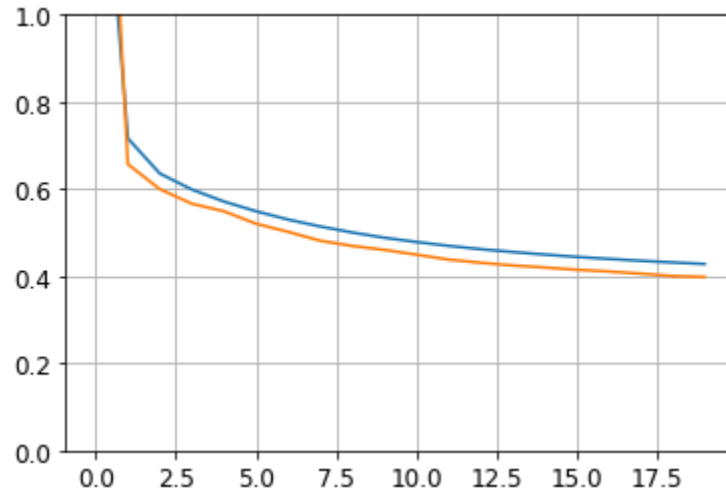


```
# Compile with creating an optimizer object
```

```
model.compile(loss="mean_squared_error",  
    optimizer=keras.optimizers.SGD(lr=1e-3))
```

6.3 Train and Evaluate the Model

```
history = model.fit(X_train, y_train, epochs=20,  
                    validation_data=(X_valid, y_valid))
```



```
mse_test = model.evaluate(X_test, y_test)
```

```
5160/5160 [=====] - 0s
```

```
15us/sample - loss: 0.421
```

6.4 Save and Restore the Model

- After training a model save it to a file.

```
model.save("my_keras_model.h5")
```

- In the production program, load the trained model.

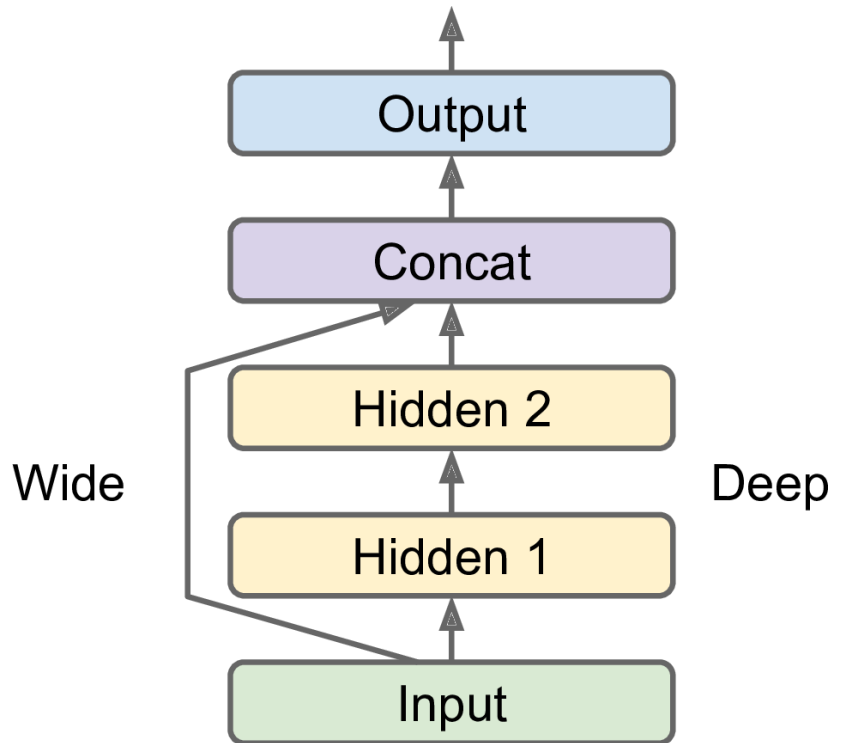
```
model = keras.models.load_model("my_keras_model.h5")
```

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

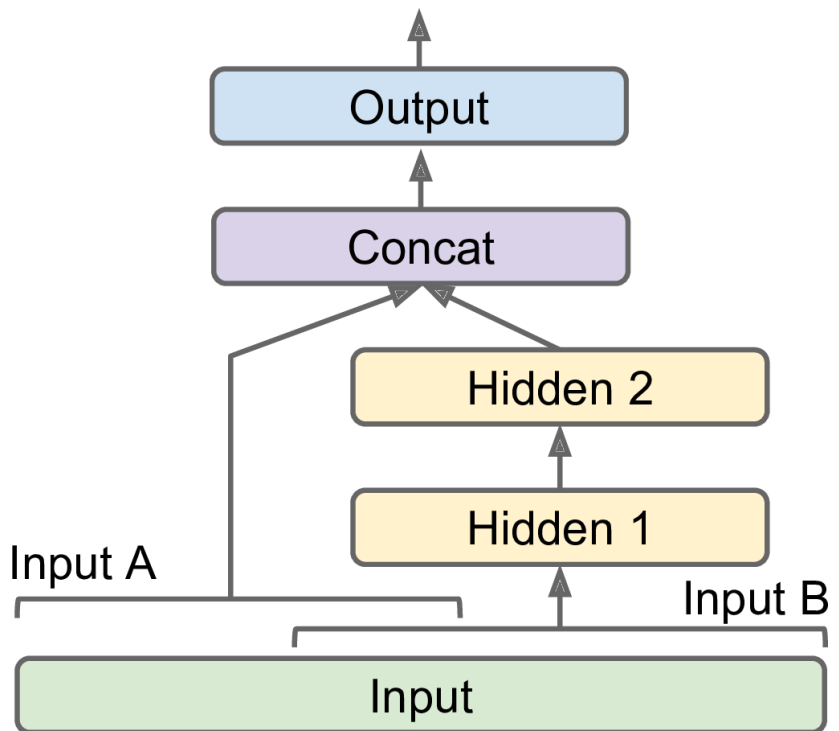
7. Using the Functional API

- Keras functional API can be used to build arbitrary static graph topologies.
- Create a layer and as soon as it is created, **call it like a function**, passing it the input.
- Example 1: the **wide and deep** network that learns both deep patterns (using the deep path) and simple rules (through the short path).



7. Using the Functional API

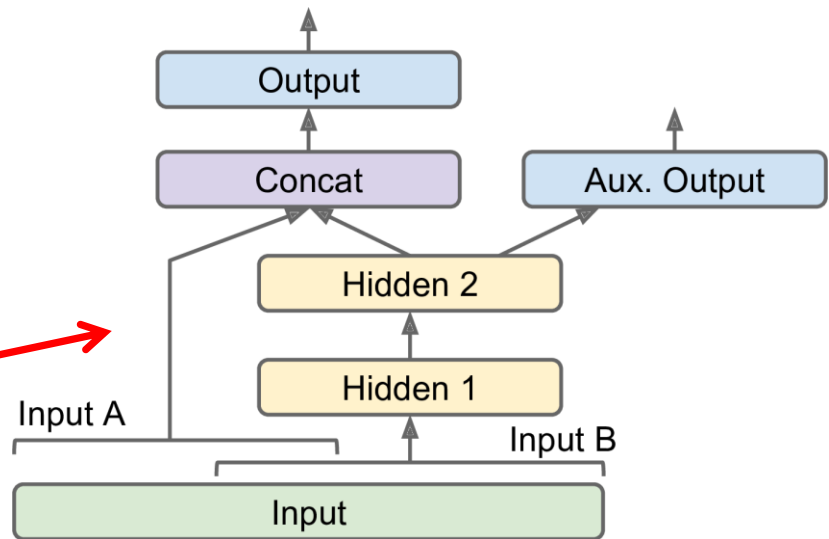
2. Multi-input: You can send a subset of the features through the wide path, and a different subset (possibly overlapping) through the deep path.



7. Using the Functional API

3. Multiple Outputs

- To **locate and classify** the main object in a picture.
- **Multiple independent tasks** to perform based on the same data.
- **Regularization technique** (to ensure that the deep network learns something useful on its own).



7.1 Auxiliary Output for Regularization

```
# Build the model
```

```
input_A = keras.layers.Input(shape=[5], name="wide_input")
input_B = keras.layers.Input(shape=[6], name="deep_input")
hidden1 = keras.layers.Dense(30,
           activation="relu")(input_B)
hidden2 = keras.layers.Dense(30,
                               activation="relu")(hidden1)
concat = keras.layers.concatenate([input_A, hidden2])
output = keras.layers.Dense(1, name="main_output")(concat)
aux_output = keras.layers.Dense(1,
                                  name="aux_output")(hidden2)
```

```
model = keras.models.Model(inputs=[input_A, input_B],
                            outputs=[output, aux_output])
```

7.1 Auxiliary Output for Regularization

```
# Split the input
```

```
X_train_A, X_train_B = X_train[:, :5], X_train[:, 2:]
```

```
X_valid_A, X_valid_B = X_valid[:, :5], X_valid[:, 2:]
```

```
X_test_A, X_test_B = X_test[:, :5], X_test[:, 2:]
```

```
# Take some test samples
```

```
X_new_A, X_new_B = X_test_A[:3], X_test_B[:3]
```

7.1 Auxiliary Output for Regularization

```
# Compile, train, evaluate, and predict
```

```
model.compile(loss=["mse", "mse"], loss_weights=[0.9, 0.1],  
              optimizer=keras.optimizers.SGD(lr=1e-3))
```

```
history = model.fit([X_train_A, X_train_B],  
                   [y_train, y_train], epochs=20,  
                   validation_data=( [X_valid_A, X_valid_B],  
                                     [y_valid, y_valid]))
```

```
total_loss, main_loss, aux_loss = model.evaluate(  
    [X_test_A, X_test_B], [y_test, y_test])
```

```
y_pred_main, y_pred_aux = model.predict([X_new_A, X_new_B])
```

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

8. Using Callbacks

- The `fit()` method accepts a `callbacks` argument that lets you specify a list of objects that Keras will call during training
 - at the start and end of **training**
 - at the start and end of each **epoch**
 - before and after processing each **batch**
- There are many callbacks available in the `keras.callbacks` package. See

<https://keras.io/callbacks/>

8.1 Saving Best Model

- Save your model when its performance on the validation set is the best so far.

```
checkpoint_cb = keras.callbacks.ModelCheckpoint(  
    "my_keras_model.h5", save_best_only=True)  
history = model.fit(X_train, y_train, epochs=10,  
    validation_data=(X_valid, y_valid),  
    callbacks=[checkpoint_cb])
```

```
model = keras.models.load_model("my_keras_model.h5")  
# rollback to best model  
mse_test = model.evaluate(X_test, y_test)
```


8.2 Early Stopping

- Interrupt training when there is no progress on the validation set for a number of epochs (defined by the `patience` argument)
- Optionally roll back to the best model.

```
early_stopping_cb = keras.callbacks.EarlyStopping(  
    patience=10, restore_best_weights=True)
```

```
history = model.fit(X_train, y_train, epochs=100,  
    validation_data=(X_valid, y_valid),  
    callbacks=[checkpoint_cb, early_stopping_cb])
```

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise

9. Visualization Using TensorBoard

- TensorBoard is a great interactive visualization tool that comes with TensorFlow.
- Use it using its callback

```
tensorboard_cb =
```

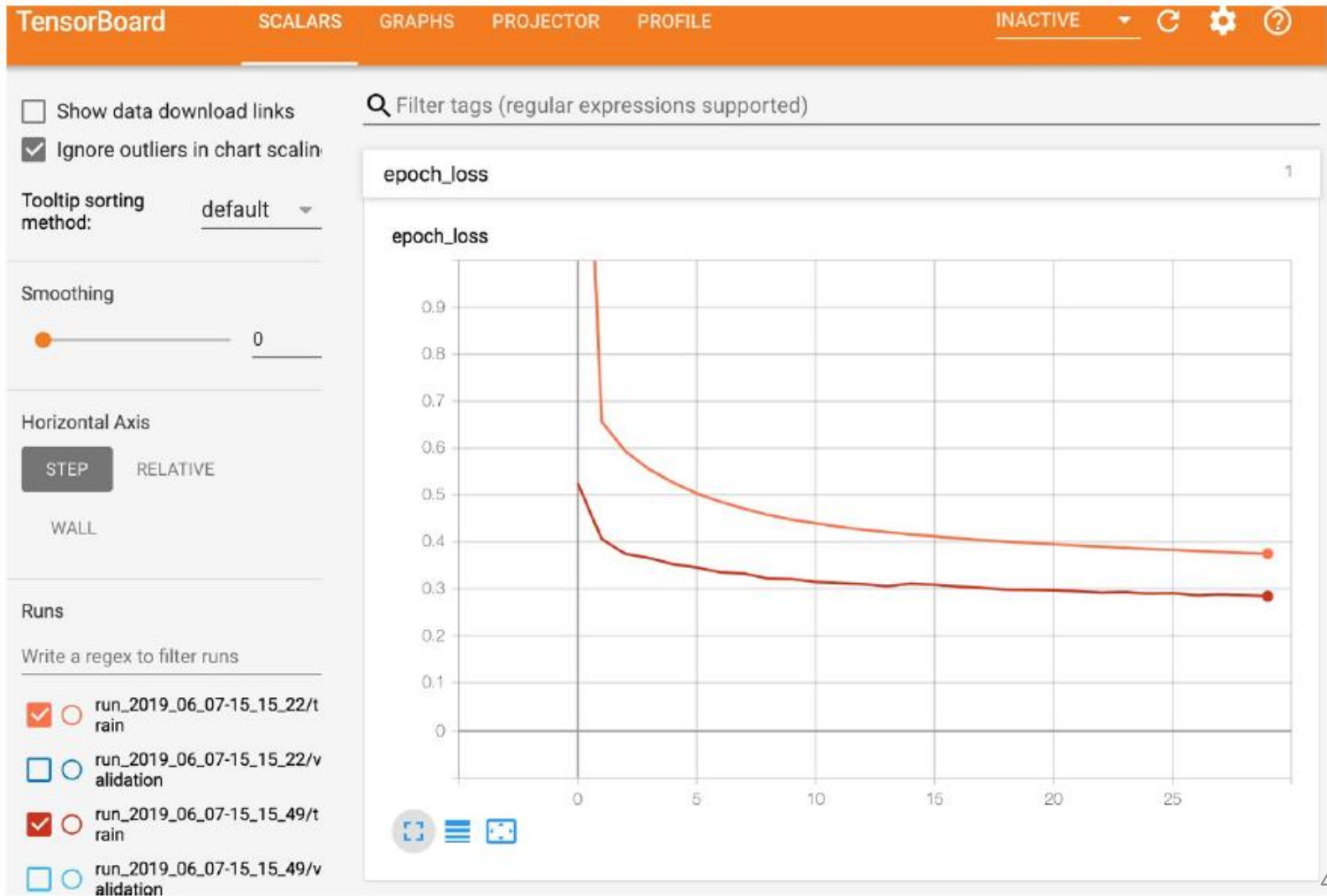
```
    keras.callbacks.TensorBoard(run_logdir)
```

```
history = model.fit(X_train, y_train, epochs=30,  
                    validation_data=(X_valid, y_valid),  
                    callbacks=[tensorboard_cb])
```

- Start TensorBoard server

```
$ tensorboard --logdir=./my_logs --port=6006
```

9. Open <http://localhost:6006>



Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
- 10. Fine-Tuning Neural Network Hyperparameters**
- 11. Tutorials**
- 12. Exercise**

10. Fine-Tuning Neural Network Hyperparameters

- **Number of Hidden Layers**
 - One hidden layer can theoretically model even the most complex functions, provided it has enough neurons.
 - But for complex problems, deep networks have a much higher parameter efficiency than shallow ones.
- **Number of Neurons per Hidden Layer**
 - **Pyramid** across layers or **same** size
 - **Stretch pants**: pick a model with more layers and neurons than you actually need, then use early stopping and other regularization techniques to prevent it from overfitting.
- Better to increase the number of layers instead of the number of neurons per layer.

10. Fine-Tuning Neural Network Hyperparameters

- **Learning Rate**: the optimal LR is about half of the maximum LR.
- **Optimizer**: There are other than the Mini-batch Gradient Descent optimizer.
- **Batch Size**
 - Larger gives better speed up with hardware accelerators.
 - Smaller makes the models more general.
- **Activation Functions**

11. Tutorials

- <https://keras.io/>
- <https://www.tensorflow.org/guide/keras>
- Keras Tutorial: Deep Learning in Python from DataCamp,
<https://www.datacamp.com/community/tutorials/deep-learning-python>
- Keras Tutorial: The Ultimate Beginner's Guide to Deep Learning in Python, from EliteDataScience,
<https://elitedatascience.com/keras-tutorial-deep-learning-in-python>

12. Exercise

From Chapter 10, solve exercise:

- 10

Summary

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. Image Classifier Using the Sequential Model
5. Example - MNIST
6. Regression Using the Sequential Model
7. Using the Functional API
8. Using Callbacks
9. Visualization Using TensorBoard
10. Fine-Tuning Neural Network Hyperparameters
11. Tutorials
12. Exercise