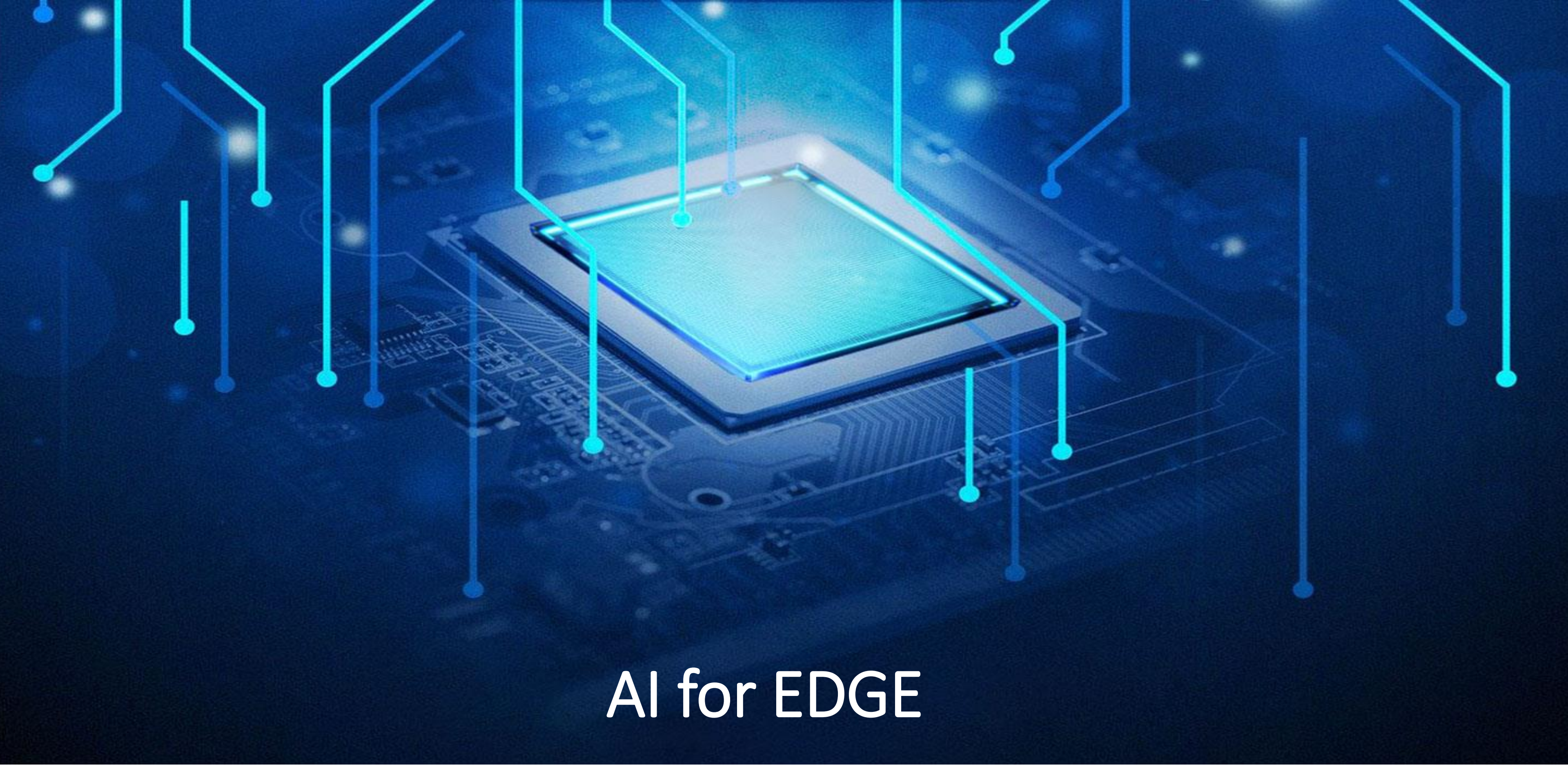


AI for EDGE

Machine learning frameworks

Outline

Machine learning Frameworks

- TensorFlow
- Keras
- scikit-learn
- PyTorch
- OpenVINO

TensorFlow

- Free and open-source software library for machine learning and AI
- Was developed by the Google Brain team and released under the Apache License 2.0 in 2015
- Updated version, named TensorFlow 2.0, released in 2019
- Can be used in a wide variety of programming languages (including Python, JavaScript, C++, and Java)

TensorFlow.js

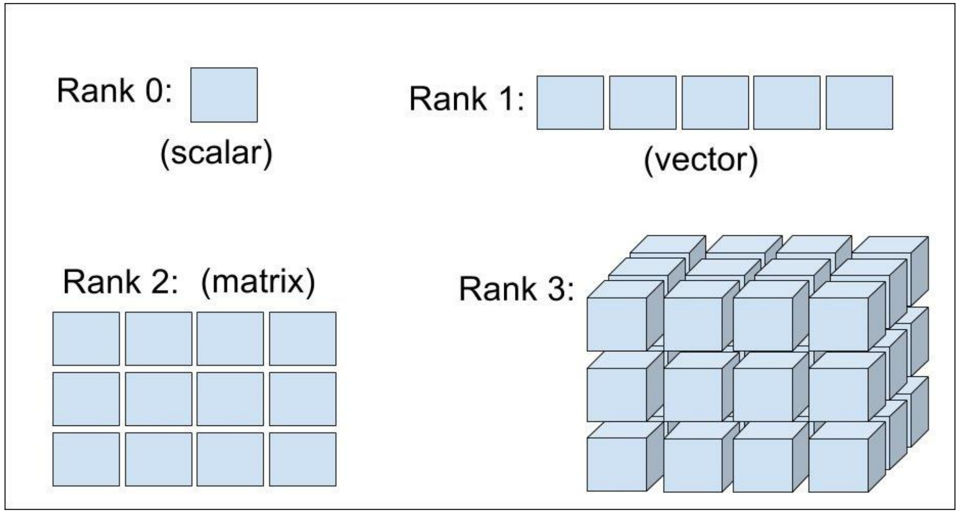
TensorFlow

K Keras

TensorFlow Lite

TensorFlow

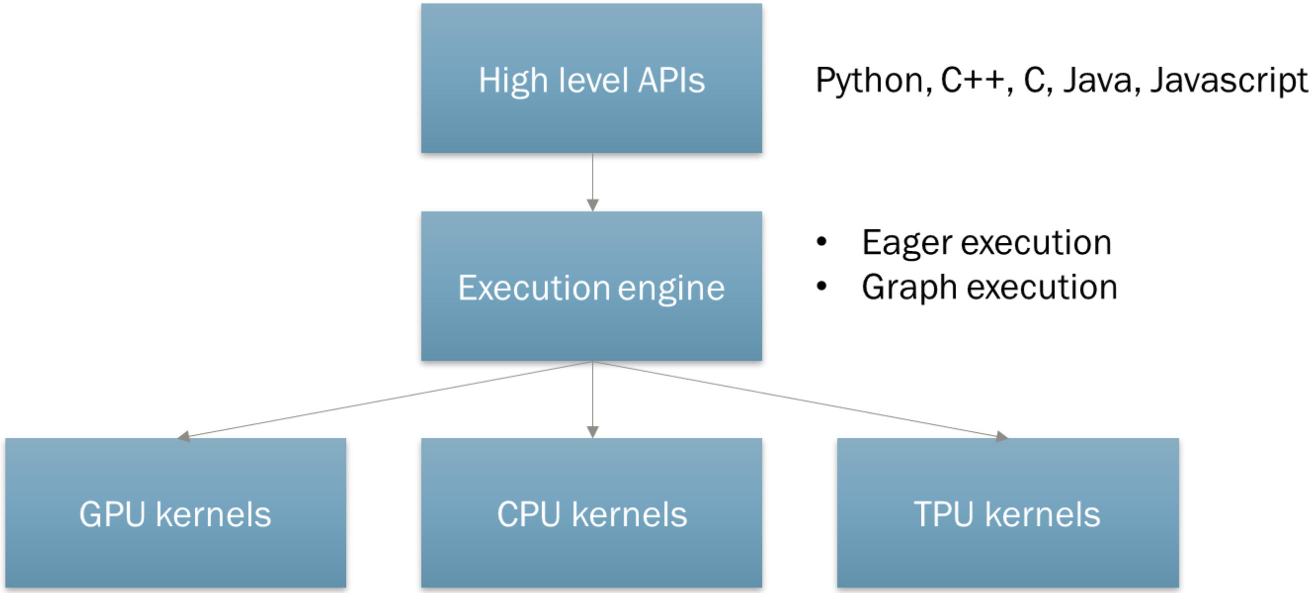
- A framework dedicated to machine learning, focusing mainly on deep neural networks
- The essence of Tensorflow are data structures, called tensors
- Each tensor has the following fields:
  - Rank
  - Shape
  - Size
  - Dtype
  - Data



TensorFlow

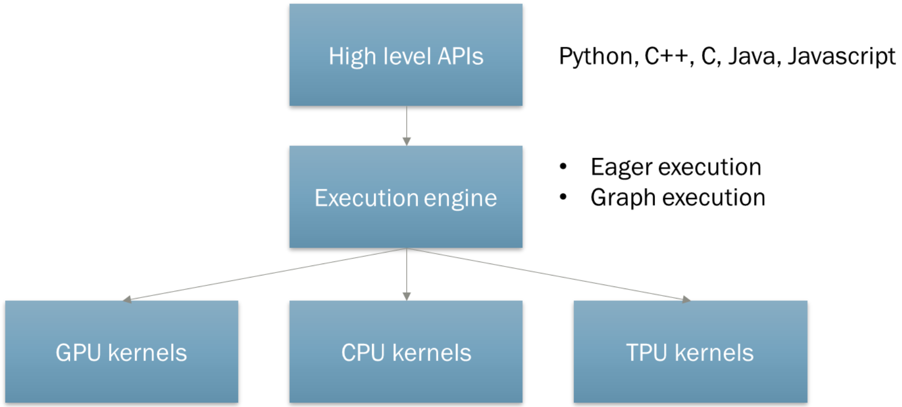
- A specific feature of tensors is that they cannot be edited. If we want to change the values of a tensor, we have to create a new one.
- The TensorFlow API gives us a wide range of operations on tensors
- These operations are defined so that they can be easily parallelized (from the perspective of the user code, they are performed on the entire array at once or on its part)
- Tensorflow does not automatically convert data for arithmetic operations

Framework architecture



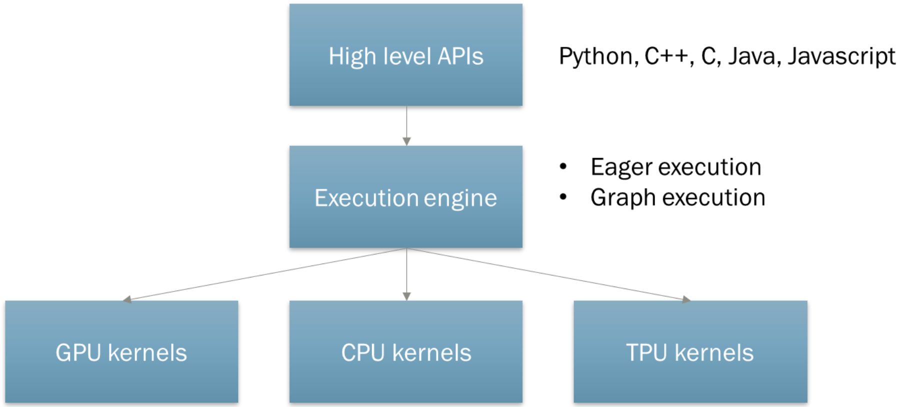
TensorFlow

- Eager execution
  - The default execution mode in TensorFlow 2.x
  - It allows for flexible code writing, thanks to which we can mix TensorFlow operations with pure Python code and other libraries (e.g. Numpy)
  - The code is executed synchronously, similar to pure Python code
  - Allows for easy debugging



TensorFlow

- Graph execution
  - This is the mode in which a set of tensor operations forms a computational graph. Such a computational graph is automatically optimized and the memory for its operations is preallocated. This results in a significant acceleration of calculations. This mode can be used in conjunction with eager mode, since a single function is designated as a graph.

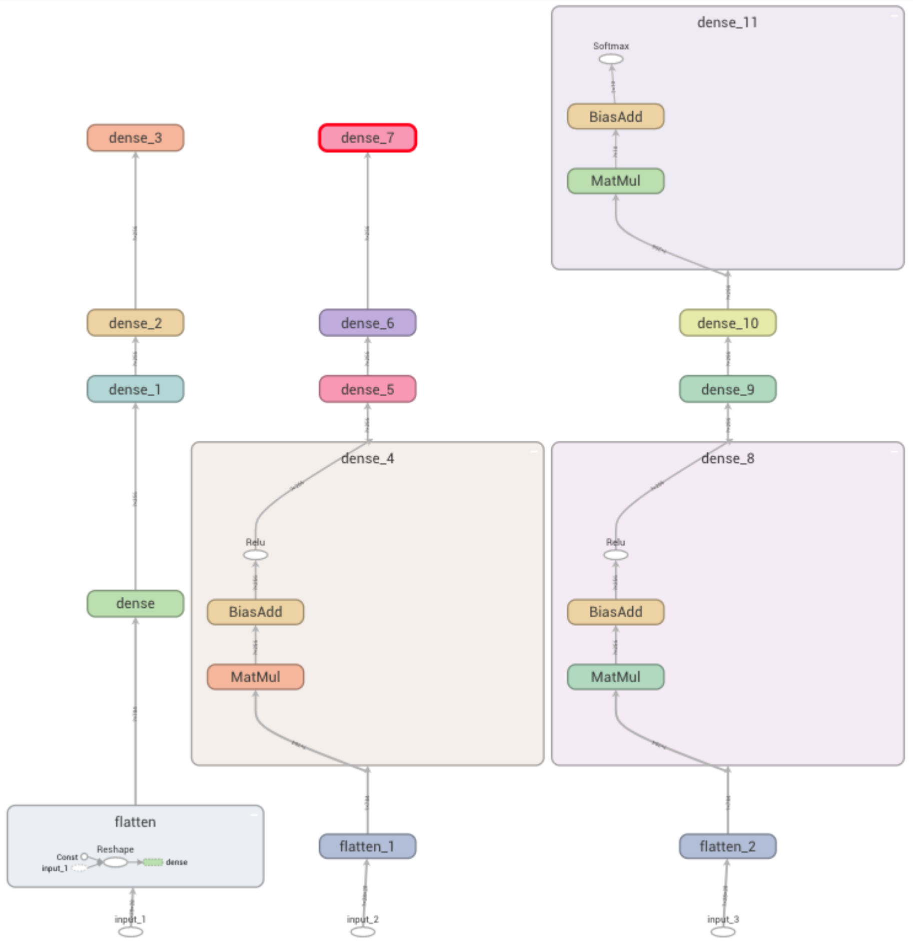


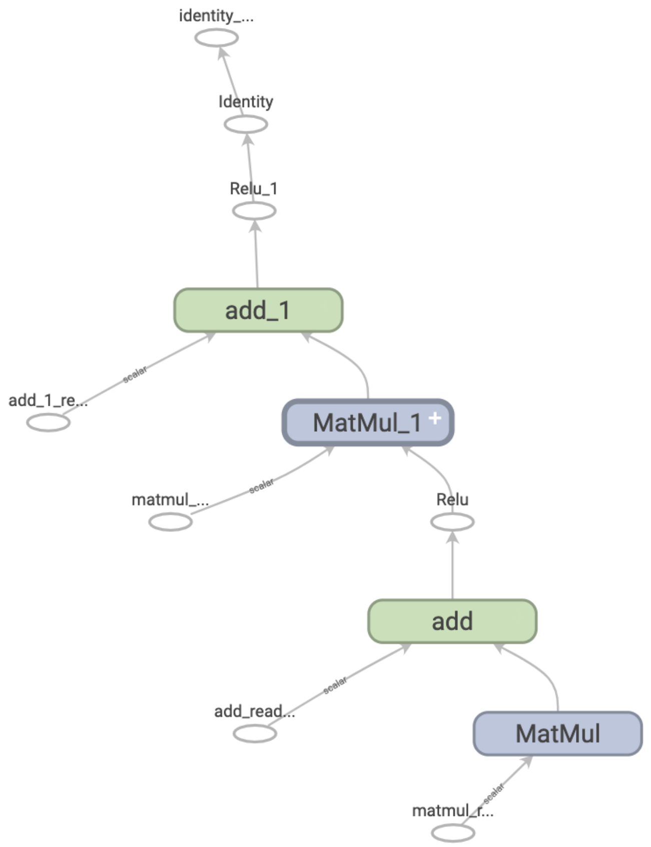
TensorFlow – Graph mode

Running a computational graph has two phases

- Tracing - involves running and tracing all tensor operations in the graph definition and creating a **tf.Graph** object from them. The **tf.Graph** object has preallocated memory for tensor operations in the graph and is optimized by the **AutoGraph** module built into TensorFlow. Due to memory preallocations in the graph, the tensor shapes must be known in advance. The more constraints we introduce on the shape of the tensors, the better optimization we get. Tensor sizes that are not known can be marked as dynamic by the "**None**" argument.
- Execution - execution of a computational graph on given inputs, which returns the result
- The idea of the graph mode is to perform tracing only once, at the first start of the graph, and then only perform the second phase. The first run of the graph is usually much longer than each subsequent run. They are colloquially referred to as graph warm up.

TensorFlow – Graph mode





Classroom materials

TensorFlow – Graph mode

- Limitations
  - Need to use operations defined by TensorFlow. Built-in Python functions like **list.append()** or object-oriented polymorphism will not work
  - It does not support recursion
  - Operations are performed asynchronously, which makes debugging difficult. This will also cause Python prints to not work (this can be circumvented by the **tf.print()** method)
  - In graph mode, graph outputs must be returned with **"return...".** It is not possible to modify an external Python variable from within a graph. Referring to such a variable will create a tensor based on it (if conversion is possible)

TensorFlow – Graph mode

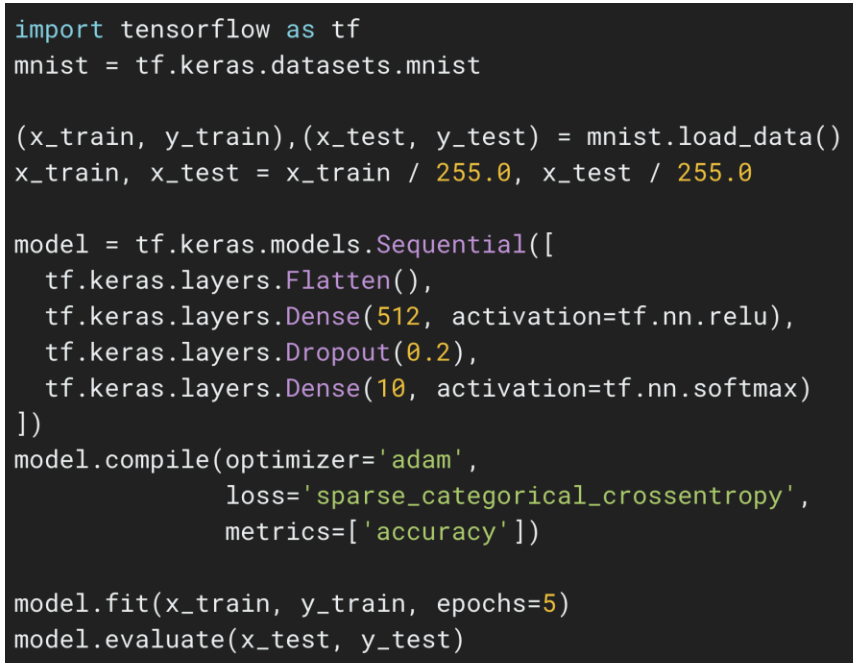
```
class Module:
        @tf.function
        def graph_function(x: tf.Tensor):
              y = x*x + 5
              return tf.math.reduce_mean(y)

mod = Module()

# Running eagerly
x = tf.constant([1, 2, 3])
x += 5

# Running in graph mode
result = mod.graph_function(x)

# Saving
tf.saved_model.save(mode, 'graph')
```

Usage

- To define a computational graph, we need to mark a function with the **@tf.function** decorator.
- We can also save the graph definition to a file. The main format for saving computational graphs in TensorFlow is the **saved model**.

Tensorflow Keras

- High-level API for designing neural networks
- The API is declarative, i.e. it allows you to declare the structure of the target computational graph (neural network)
- Components in a graph are objects that inherit from the **"tf.keras.layers.Layer"** class
- Components, training algorithms and metrics are available out of the box along with other useful things
- Graphs created by Keras API can run in eager or graph mode.
- Graph mode is enabled by default.

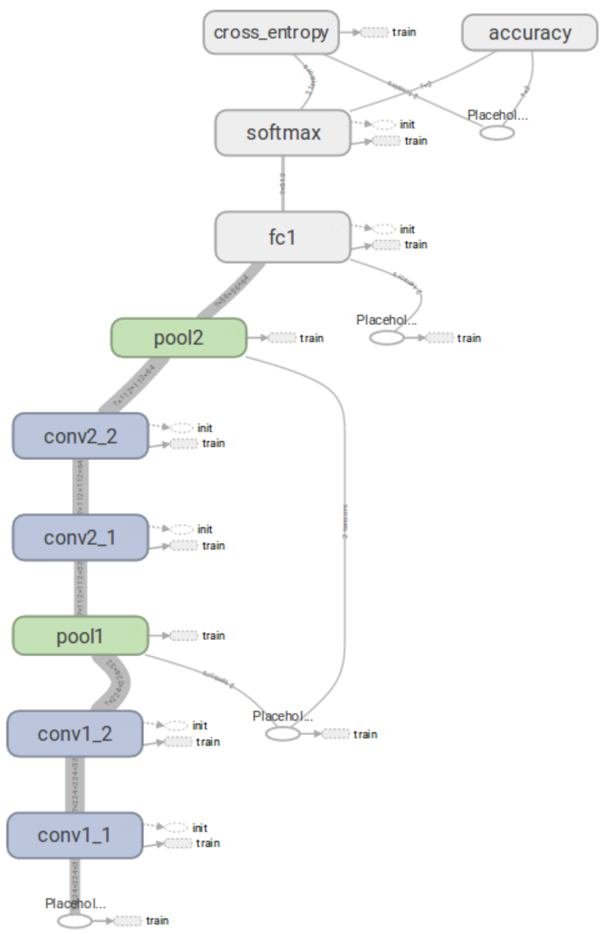


```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Tensorflow Keras - Sequential API

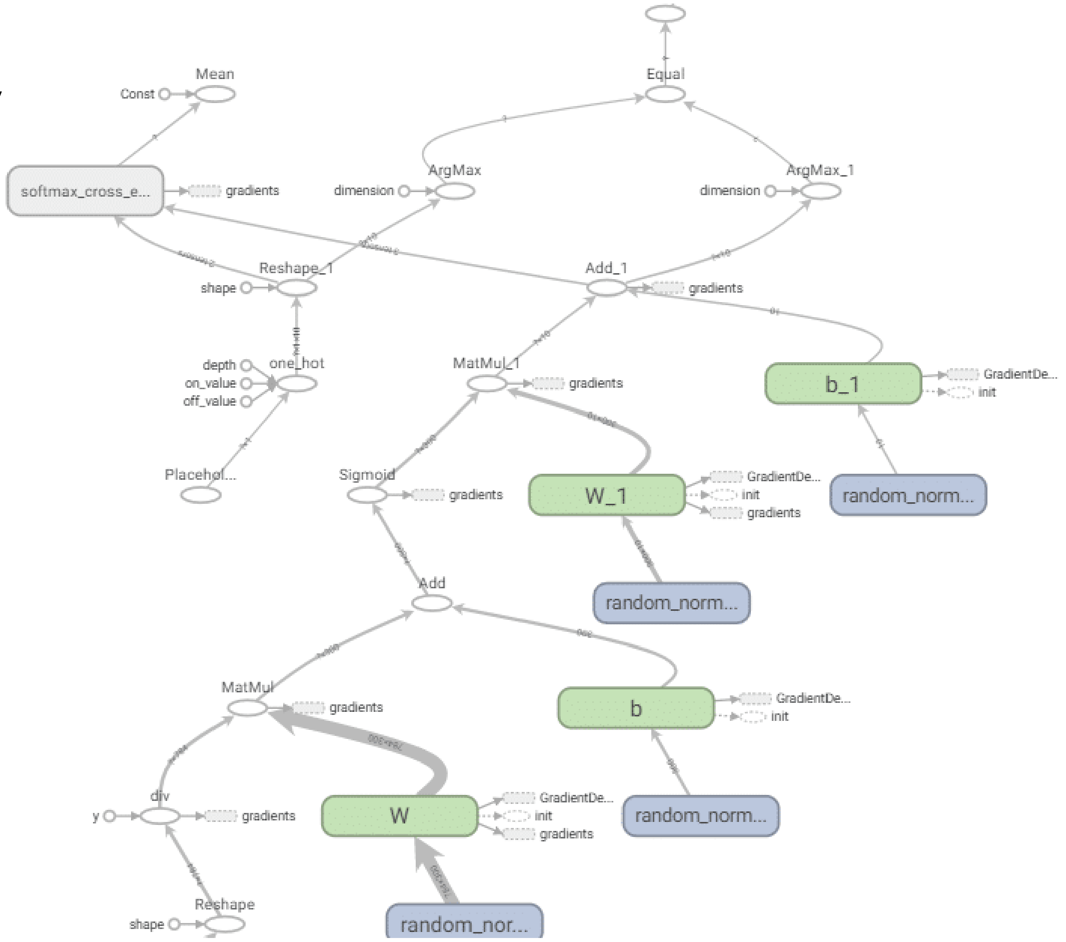
- The Sequential API in Keras has limited capabilities and is used to define stacked graph structures in which all operations are performed consecutively on one line.

Main Graph



Tensorflow Keras - Functional API

- Functional API in Keras has very wide possibilities in terms of declaring graph structures.
- We can use it to do the same as in the Sequential API, but in addition, the connections in the graph can be non-linear, and the graphs themselves can have many inputs and many outputs.

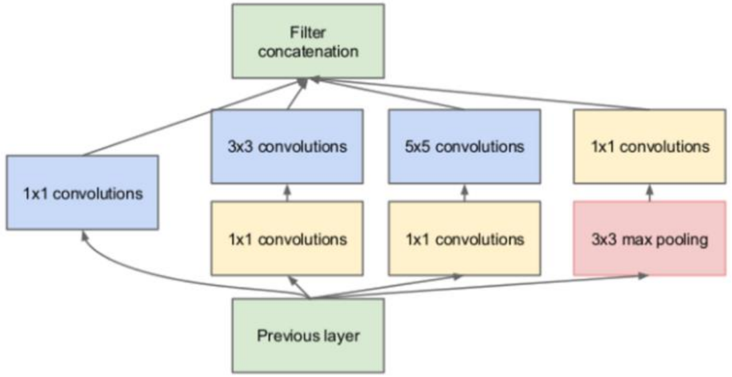


Tensorflow Keras - Functional API

- The **tf.keras.Model** object accepts **tf.keras.Input** objects and output tensors as input.
- **tf.keras.Model** automatically registers all operations which, based on inputs, will create appropriate outputs.

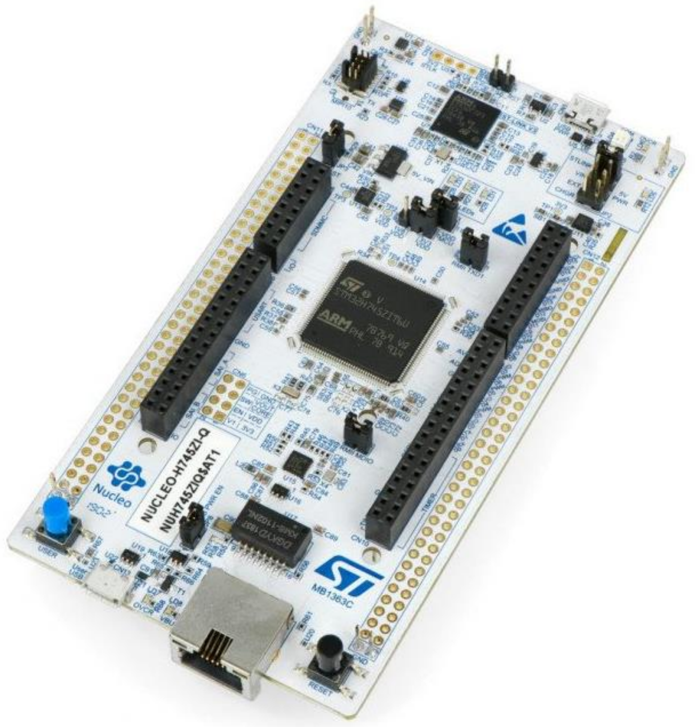
```
encoder_input = keras.Input(shape=(28, 28, 1), name="img")
x = layers.Conv2D(16, 3, activation="relu")(encoder_input)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.MaxPooling2D(3)(x)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.Conv2D(16, 3, activation="relu")(x)
encoder_output = layers.GlobalMaxPooling2D()(x)

encoder = keras.Model(encoder_input, encoder_output, name="encoder")
```

- **tf.keras.Input** is a so called symbolic tensor that has some information, such as a shape, but no value. It is used to declare constraints on input tensors.
- Symbolic tensors can have dynamic shapes, which we declare as **None**.
- As with **@tf.function**, tracing occurs the first time the graph (**tf.keras.Model** object) is queried. In the case of Keras, this step is referred to as a build step.



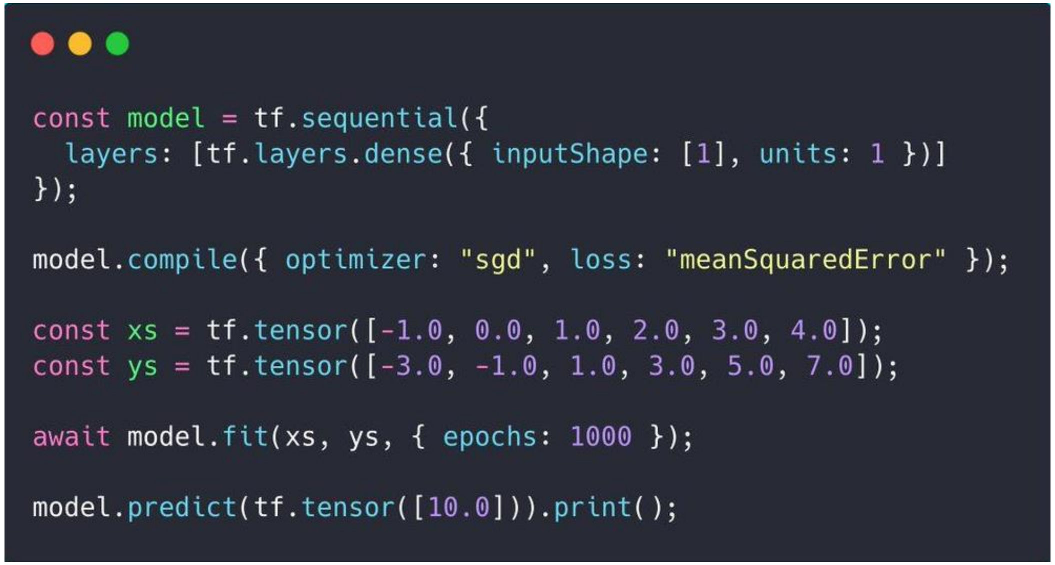
Tensorflow Lite

- A set of tools that allows you to deploy computational graphs to mobile, embedded and IoT devices
- Supports TF Lite special graph format
- Calculation graphs are converted to this format before deployment
- The number of operations that can be used in this type of graph is limited
- The minimum memory requirements for TF Lite are only a few hundred kB.
- Support for: Android, IOS, Embedded Linux and 32-bit microcontrollers



Tensorflow.js

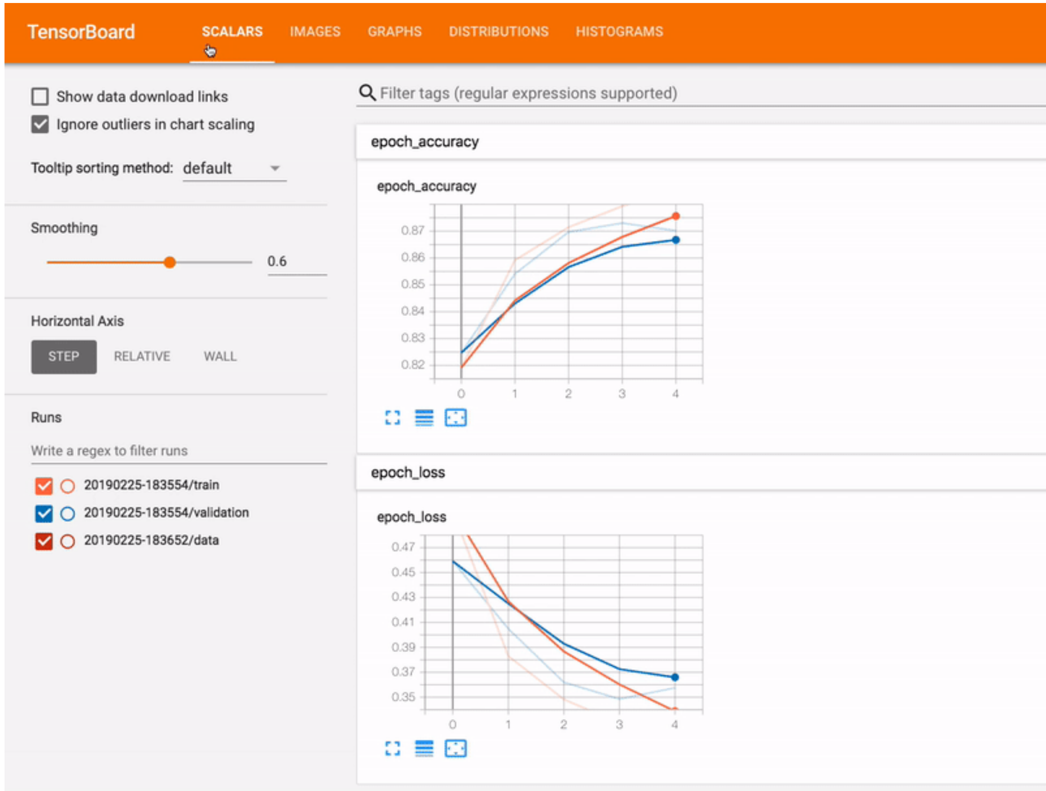
- Javascript library enabling the prototyping and training of neural networks in a browser
- Uses the same graphical hardware as the browser (GPU or integrated graphics)



```
const model = tf.sequential({
  layers: [tf.layers.dense({ inputShape: [1], units: 1 })]
});

model.compile({ optimizer: "sgd", loss: "meanSquaredError" });

const xs = tf.tensor([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0]);
const ys = tf.tensor([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0]);

await model.fit(xs, ys, { epochs: 1000 });

model.predict(tf.tensor([10.0])).print();
```

TensorflowBoard

- Browser tools for visualizing ML experiments
- Visualizations include e.g. training metrics, histograms of weights and biases in networks, computational graph diagrams and training examples
- Also allows profiling programs using Tensorflow



☐ Show data download links
☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

☑ 20190225-183554/train
☑ 20190225-183554/validation
☑ 20190225-183652/data

Related topics

https://www.tensorflow.org/guide

https://scikit-learn.org/stable/user_guide.html

Post-Training Quantization Best Practices
https://docs.openvino.ai/latest/pot_docs_BestPractices.html#doxid-pot-docs-best-practices

Movidius Myriad X VPU
https://www.aaeon.ai/eu/product/detail/ai-core-x