6. - Deep Learning APIs

6.2 - Deep Learning using TensorFlow

Rodolfo Sabino

10 Setembro, 2020

Departamento de de Computação

Disclaimer

The following content is heavily based on HCIA-AI Course material by Huawei Technologies Co., Ltd., authored by Zhang Luxiao and Yuan Meng. Distribution is not allowed.

Index

- TensorFlow 2.x Environment Setup
- TensorFlow 2.x Development Overview
- Study Case: MNIST Digit Recognition

TensorFlow 2.x Environment Setup

TensorFlow 2.x Installation

First, install Python3 and pip, maybe using a manager such as Miniconda

By using **pip** package manager, the installation is simple from command line:

```
pip install tensorflow
```

For using the stable release of TensorFlow for CPU (recommended), or

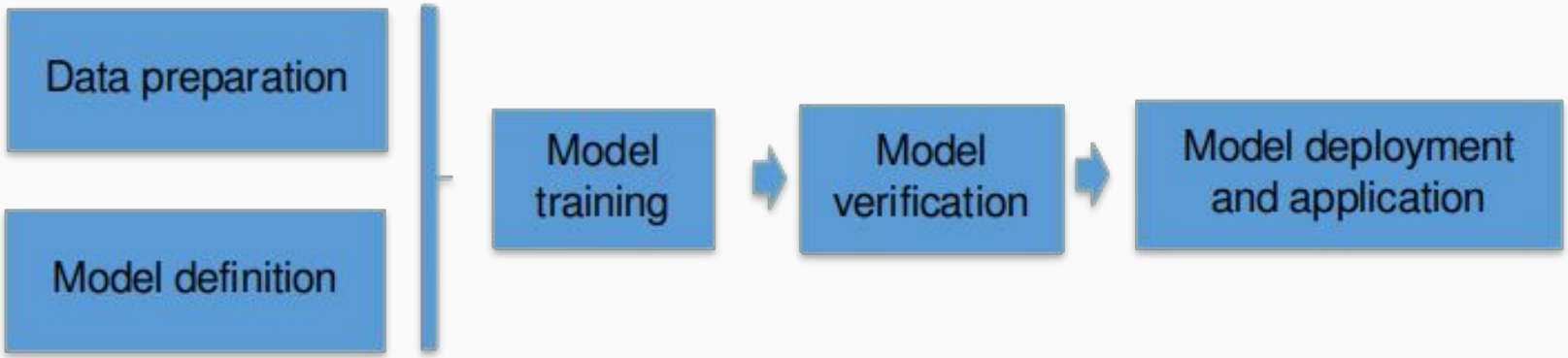
```
pip install tensorflow-gpu
```

for GPU-acceleration support (avoid this unless you know what you are doing).

Or simply create a Colaboratory Notebook in Google Drive

TensorFlow 2.x Development Overview

TensorFlow Development Process



Defining the Model

Models in TensorFlow can be either sequential or functional.

Sequential models provided by tf.keras.Sequential are simple to use and cover a wide range of common scenarios.

Functional models provided by tf.keras.Model are more complex allow to create more versatile applications, with non-sequential connections and multiple input / outputs.

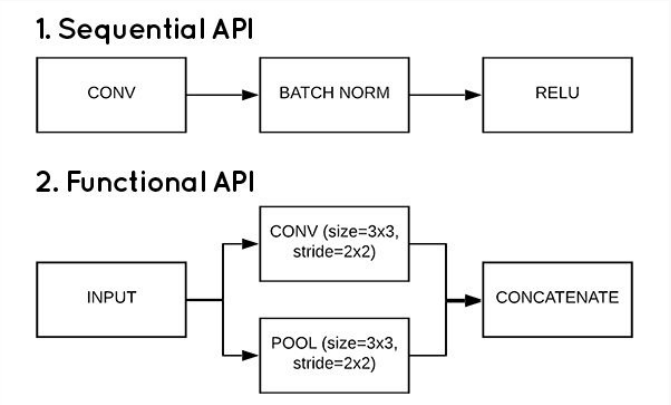


Image source

Defining the Model

Building and configuring layers of a neural network. tf.keras.layers offers different types of layers based on different algorithms and optimized for different applications, such as image convolution layers and RRN layers.

Training the Model

After defining the model, use **`tf.keras.Model.compile()`** to compile it, i.e., to configure how the underlying learning process is carried out by defining a loss function, an optimizer, evaluation metrics, and other setups. Common options can be found in tf.keras.losses, tf.keras.optimizers, and tf.keras.metrics modules.

Then **`tf.keras.Model.fit()`** to train the model.

Training the model may take a long time for complex models and big datasets. Callback functions from tf.keras.callback should be employed to add breakpoints and give opportunity of saving and further loading the model training set.

Saving and Loading the Model

The model can be saved during the training as a checkpoint.

A model can be saved directly by invoking tf.keras.Model.save() / tf.keras.Model.save_weights() once the training is over.

Finally, a model can be loaded by calling tf.keras.models.load_model() or tf.keras.Model.load_weights(), for example.

Testing the Model

After training, tf.keras.Model.evaluate() can be used to test the model. It returns useful metrics such as loss and accuracy values.

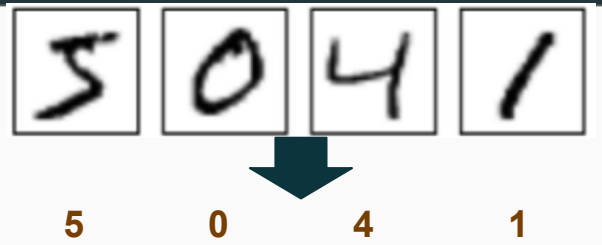
Using the model

Finally, using the model for inference is relatively easy. Use tf.keras.Model.predict() with an input tensor acting as samples or batches of samples and this function will generate outputs predictions for the input samples.

Further reading on these steps:
https://www.tensorflow.org/guide/keras/train_and_evaluate

Study Case: MNIST Digit Recognition

MNIST Dataset



Can be downloaded from http://yann.lecun.com/exdb/mnist/

Different from printed fonts, handwriting of different people has different sizes and styles, making it difficult for computers to recognize handwriting.

The MNIST datasets consist of a training set and a test set

- Training set: 60,000 handwriting images and corresponding labels
- Test set: 10,000 handwriting images and corresponding labels

MNIST Dataset

Examples

Corresponding labels

[0,0,0,0,0, 1,0,0,0,0]	[0,0,0,0,0, 0,0,0,0,1]	[0,0,0,0,0, 0,0,1,0,0]	[0,0,0,1,0, 0,0,0,0,0]	[0,0,0,0,1, 0,0,0,0,0]

TensorFlow 2.x and MNIST: Quick and Easy

```
import tensorflow as tf
import tensorflow.keras as keras

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

model = keras.models.Sequential([
                                    keras.layers.Flatten(input_shape=(28, 28)),
                                    keras.layers.Dense(128, activation='relu'),
                                    keras.layers.Dropout(0.2),
                                    keras.layers.Dense(10)
                                  ])

model.compile(optimizer='adam',
              loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=32, epochs=5)
model.evaluate(x_test,  y_test, verbose=1)
```

Thank You!

Next: Lab Guides 04 and 05