CC-MEI 2024: Introduction to Deep Learning Programming in the Cloud

Jordi Torres

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Our world has changed, and it has changed forever due to the impact of the recent, rapid growth of artificial intelligence (AI) technology. And what has caused this AI explosion? Undoubtedly, supercomputers provided through the Cloud are one of the critical pillars of AI's progress, speeding up the training of algorithms. The reason is that deep learning (DL) algorithms are highly parallel, making them not only conducive to taking advantage of GPU acceleration but also scalable to multiple GPUs and nodes. The following hands-on exercises will introduce you to using the Cloud for training a DL code and learn how to program a parallel supercomputer for training a real artificial intelligent problem.

Important note: Beware of "copy & paste" as symbols such as commas or dashes can be incorrectly copied from this document to the console. If a command/code doesn't work correctly, write it directly!

Hands-on 1-5

Hands-on Reports

```
Write a report for each exercise that includes all the tasks,
detailing the steps that were taken, the code used, and the results.
Once finished, generate a PDF version and submit it to the
corresponding inbox at the "racó" intranet.
```

1. IA in the Cloud

In this section, we will learn how to train a neural network using a Jupyter Notebook in the Google Cloud.

Google Colab

Google Colab is a Jupyter notebook based runtime environment that allows you to run code on the Google Cloud that can be accessed remotely through a browser. Google Colab supports accelerators, both GPU and TPU instances.

Task 1.1

Open a browser of your choice, go to colab.research.google.com, and sign in using your Google account. Click on a new notebook to create a new runtime instance or download a GitHub file. To create a GPU/TPU-enabled runtime, click on the "runtime" label in the toolbar menu, click "Change runtime type", and then select GPU or TPU.

Jupyter notebook

Training a DL model in Colab is very easy because it is not required to set up a custom runtime environment for the user (equivalent to *module load* in Marenustrum); it's all handled by him. For example, train a basic DL model to recognize handwritten digits (the model is basic, categorizes images as numbers, and recognizes them) trained on the MNIST dataset. The code uses Tensorflow library, and the data can be loaded from the standard Keras dataset archive. Execute by your own the following code to understand the details explained by the teacher in the class:

`https://github.com/jorditorresBCN/HPC-DL/blob/main/GettingStarted_DL.ipynb`

```
import tensorflow as tf
from tensorflow import keras

import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
```

```
import matplotlib.pyplot as plt
plt.imshow(x_train[8], cmap=plt.cm.binary)
```

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

x_train /= 255
x_test /= 255
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
```

```
from tensorflow.keras.utils import to_categorical
```

```
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)
```

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(10, activation='sigmoid', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
model.summary()
```

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

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

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
                range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Observación')
    plt.xlabel('Predicción')

from collections import Counter
from sklearn.metrics import confusion_matrix
import itertools

# Predict the values from the validation dataset
Y_pred = model.predict(x_test)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred, axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(y_test, axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
```

```
plot_confusion_matrix(confusion_mtx, classes = range(10))
```

```
x_test_old = x_test.reshape(10000, 28,28)
plt.imshow(x_test_old[11], cmap=plt.cm.binary)
predictions = model.predict(x_test)
np.argmax(predictions[11])
print(predictions[11])
```

Task 1.2

In your Colab, download the GitHub file `https://github.com/jorditorresBCN/HPC-DL/blob/main/GettingStarted_DL.ipynb` . Define, train, and evaluate a new model that improves the Accuracy obtained by the basic model (teacher's model). (a) Explain the improvement applied: More epochs? More layers? More neurons per layer? Print and analyze the confusion matrix. (b) Compare the optimizers SGD and Adam for your neural networks. Describe the results of this task in the lab report.

Convolutional Neural Network

Below, we describe the code that implements the solution for the previous image classification problem using convolutional neural networks. You can find this code in the same .ipynb file as the previous one. Feel free to use the same Colab notebook to analyze this code while executing it. The neural network-based solution is the one we will employ in future tasks.

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten

model = Sequential()
model.add(Conv2D(32, (5, 5), activation='relu', input_shape=(28, 28,
1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))

from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
```

```
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
from tensorflow.keras.utils import to_categorical

#mnist = tf.keras.datasets.mnist(train_images, train_labels),
(test_images, test_labels) =
mnist.load_data(path='/gpfs/projects/nct00/nct00008/basics-
utils/mnist.npz')

mnist = tf.keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()

print (train_images.shape)
print (train_labels.shape)
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
model.fit(train_images, train_labels, batch_size=100, epochs=5,
verbose=1)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

Task 1.3

Define, train, and evaluate a new model that improves the Accuracy obtained by the basic convolutional model (teacher's model). Explain the improvement applied. Describe the results of doing this task in the lab report.