Огляд Розвитку Технології Tensorflow Lite під Платформу Android

Степан Титаренко
кафедра Програмної інженерії
Харківський національний університет радіоелектроніки
Харків, Україна
stepan.tytarenko@nure.ua

Ірина Кириченко
кафедра Програмної інженерії
Харківський національний університет радіоелектроніки
Харків, Україна
iryna.kyrychenko@nure.ua

Overview of Tensorflow Lite Development On Android

Stepan Tytarenko
Department of Software engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
stepan.tytarenko@nure.ua

Iryna Kyrychenko
Department of Software engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
iryna.kyrychenko@nure.ua

***Анотація*—У цій статті автори хочуть зупинитися на наступних моментах: що таке сам Tensorflow, що таке Tensorflow Lite, які варіанти використання цієї технології, яка найкраща практика для пристроїв Android, обчислення хмарного машинного навчання за допомогою ML Kit, або нативного на пристрої з TFLite.**

***Abstract*—In this article the authori want to set on the following points: what is Tensorflow itself, what is Tensorflow Lite, which are the use cases for this technology, what is the best practice for android device, cloud Machine Learning computations with ML Kit, or native running on device with TFLite.**

Ключові слова—Штучний інтелект; Машинне навчання; Android; Tensorflow; TPU; Google; Хмарні технології

Keywords—Artificial Intelligence; Machine Learning; Android; Tensorflow; TPU; Google; Cloud technologies

I. Introduction

Since the supremacy of AI is coming, Google as the so-called AI-first company in the World, continues on developing Machine Learning solutions for mobile architectures. In 2017 they introduced their first version of Tensorflow Lite, as a native solution, towards running Machine Learning models on device. This idea has already lived into the minds of most of the developers for quite some time. TFLite software stack is designed specifically for mobile development. TensorFlow Lite "Micro", on the other hand, is a version specifically for Microcontrollers, which recently merged with ARM's uTensor.

A year after TFLite was announced in 2018, Google introduced Firebase ML Kit, which became the part of the FIrebase services, and provided an interface to various Machine Learning models, either custom or pre-defined running in the powerfull Cloud, so that the user device is less involved. But still, the code in the cloud runs on Tensorflow, boosted by the powerful TPUs (Tensor Processing Units) special type of processors, delivered by Google for the needs of fast Tensorflow models.

II. What is TensorFlow?

In order to understand what Tensorflow is, the best idea would be to refer to the first source, to the creators themselves.

"An entire ecosystem to help you solve challenging, real-world problems with machine learning… TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML

training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition." [1]

As it is stated from the developers, it is the ecosystem, with the build-in solutions for different ML pipeline stages, such as modelling, fine-tuning, deployment, etc. Generally tensorflow is created using C/C++, however it is mostly used in Python. Since the release of version 2, previous year, the technology has changed a lot, becoming more high-level and user-friendly, providing even simpler interface towards ML solutions. Mostly this technology is used on cope with Keras. Below the basic example of a piece of code for the Tensorflow on Java is provided [1].

```
import org.tensorflow.ConcreteFunction;
import org.tensorflow.Signature;
import org.tensorflow.Tensor;
import org.tensorflow.TensorFlow;
import org.tensorflow.op.Ops;
import org.tensorflow.op.core.Placeholder;
import org.tensorflow.op.math.Add;
import org.tensorflow.types.TInt32;
public class HelloTensorFlow {
  public static void main(String[] args) throws Exception {
    System.out.println("Hello TensorFlow " + TensorFlow.version());
    try (ConcreteFunction dbl = ConcreteFunction.create(HelloTensorFlow::dbl);
        Tensor<TInt32> x = TInt32.scalarOf(10);
        Tensor<TInt32> dblX = dbl.call(x).expect(TInt32.DTYPE)) {
      System.out.println(x.data().getInt() + " doubled is " + dblX.data().getInt());
    }
  }
  private static Signature dbl(Ops tf) {
    Placeholder<TInt32> x = tf.placeholder(TInt32.DTYPE);
    Add<TInt32> dblX = tf.math.add(x, x);
    return Signature.builder().input("x", x).output("dbl", dblX).build();
  }
}
```

Diving deeper into the review of the Tensorflow and its implementation is out of the scope of this paper, so we will only limit it with the basic notion and an example, of "Hello, World" program. Further analysis, and free tutorials with examples can be found on the Tensorflow official web-site [1].

III. What is Tensorflow Lite?

The best thing to start with defining some notion, is basically refer to the creators, and find out how they define it.

"TensorFlow Lite is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size.

TensorFlow Lite consists of two main components:

The TensorFlow Lite interpreter, which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.

The TensorFlow Lite converter, which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance." [2]

TensorFlow Lite is available on Android and iOS via a C++ API and a Java wrapper for Android developers. On devices that support it, the library can also take advantage of the Android Neural Networks API for hardware acceleration.

According to Google, Tensorflow is designed to perform Machine Learning difficult computations ondevice, without the need of sending data back and forth from a server. And that is actually a significant point, to which we will refer later in this work, discussing best practises and when it is better to use Firebase ML Kit instead of device hardware.

For developers the use of native ML tool can support:

- Latency, as far as there is no data flow to the server and back;
- Privacy, because no data leaves the device;
- Independence, as far there is no need for internet connection;
- Power consumption, internet connectivity usually quite power-demanding;

But there are some limitations on the use of Tensorflow Lite, related to running on mobile devices. TensorFlow Lite plans to provide high performance on-device inference for any TensorFlow model. However, the TensorFlow Lite interpreter currently supports a limited subset of TensorFlow operators that have been optimized for on-device use. This means that some models require additional steps to work with TensorFlow Lite.

On the fig.1 there is a Tensorflow Lite architecture. It is clear from the diagram, that custom architectures are easily applied, and natively supported via the provided TFLite API. Even though it is quite beneficial to use such way of Machine

Learning pipeline, just as always, there are some trade-offs, and here are a few of them [3]:

System utilization: Performing neural networks computations involves a lot of cpu capacity, which could increase battery power usage. The one should take into account and monitor the battery health, especially for long-running computations.

Application size: The one should also pay attention to the size of the models. Models may take up multiple megabytes of space. If bundling large models in APK would truly impact users, the one may want to consider downloading the models after app installation, using smaller models, or running computations in the cloud, as will be proposed in the further chapters. NNAPI does not provide functionality for running models in the cloud.

Below a part of code with an example of computer vision classification usage is provided, from the official Tensorflow team [1].

```
/** Initializes an {@code ImageClassifier}. */
ImageClassifier(Activity activity) throws IOException {
// TODO(b/169965231): Add support for delegates.
tflite = new Interpreter(loadModelFile(activity));
labelList = loadLabelList(activity);
imgData =
ByteBuffer.allocateDirect(
DIM_BATCH_SIZE
* getImageSizeX()
* getImageSizeY()
* DIM_PIXEL_SIZE
* getNumBytesPerChannel());
imgData.order(ByteOrder.nativeOrder());
filterLabelProbArray = new float[FILTER_STAGES][getNumLabels()];
Log.d(TAG, "Created a Tensorflow Lite Image Classifier.");
}
```

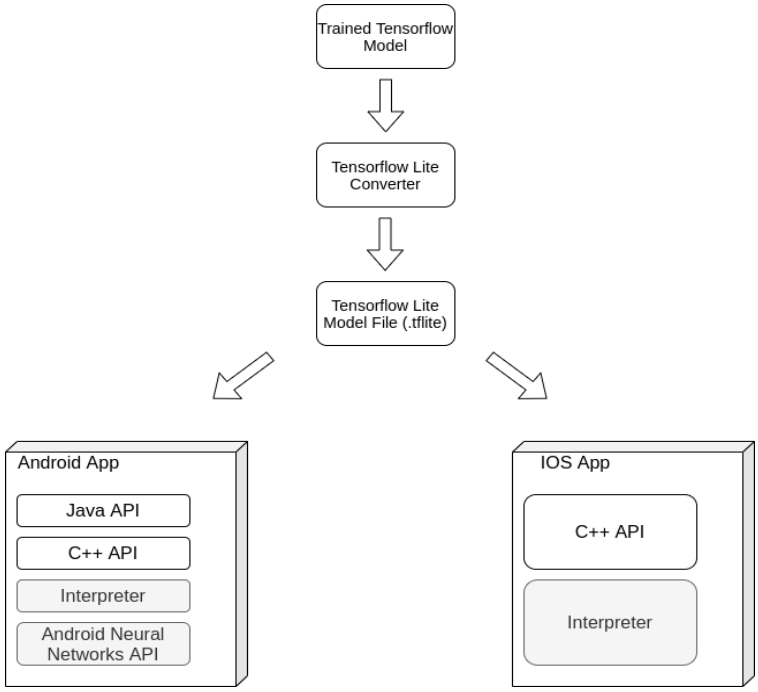

Fig. 1. TensorFlow Lite architecture

IV. TensorFlow Lite Use Cases

A typical example of a Machine Learning application is computer vision. It provides computers with the ability to recognize objects on an image or a live camera feed. To accomplish such tasks, the program must first be “trained” by the process of tuning its parameters such as weights, biases etc, and being shown quite a huge amount of pictures of the object to be taught to recognize. The system never understands what it is, it tries to recognize, it simply learns the patterns (changes in contrast, particular angles or curves) that are likely to match the object. With the passing time, the program becomes more precise at recognizing and spotting that object, depending on the task being solved.

As an Android developer, computer vision creates many possibilities: it is quite common practice today, to have facial recognition technology on Android devices, which is just about Computer vision, and is built with the help of Tensorflow tools in most cases. It serves as a security feature, in the similar way it is possible to create an augmented reality program that can highlight elements in the environment, or build the next "Reface" app. Such companies as Google are pretty concerned about AR today.

Creating and implementing these types of models from scratch would be an extremely arduous task for a single developer, which is why it's so useful to have access to ready-made libraries.

V. Comparing TensorFlow Lite and FireBase ML Kit

There are advantages and drawbacks of both, and some of them were already touched in this article, but still let us compare these two approaches on some general basis.

The TensorFlow version can be used without an Internet connection. While, when taking a snapshot, there is a noticeable delay when using the MLKit cloud API, it is related to the data transfer process, which is inevitable when we speak about cloud solutions, and the bigger data is being transferred, the bigger delay will be experienced.

The ML Kit app is smaller, since the model files are not needed, while they sometimes may reach the size of several megabytes.

Accuracy:

It is highly dependent on the implementation. Sometimes the model which is more flexible, and easily fine-tuned, where the author knows all its aspects and features can run better. But in that case it also needs more data, more time, and better expertise. While on the ML Kit solutions, there is usually "plug-and-go", as soon as the one is intended to use something, it can be used, without much effort. It definitely speeds up the development process, moreover, the models there are created and tuned by the best Google engineers as well as data gathered by the best data analysts, so in general case these models are still at least no worse than self-made ones.

Pricing:

TensorFlow, as used in an app, is free of charge. So are the ML Kit on-device APIs. But the cloud API is subject to Google Cloud pricing. The first 1000 calls each month are free. After that, it will cost around one dollar per 1000 calls.

Flexibility:

It is usually quite easy to modify the ML Kit solution to recognize one set of things instead of the other. It is just about substituting the data or changing the labels, or the models. With TensorFlow, it is required to download thousands of samples of new data from the Internet and do a new transfer learning, with possibly new models and fine-tuning, which may take several days sometimes.

But with ML Kit, the one is limited by the undocumented on-device and cloud labels supported by ML Kit. Or use ML Kit's functions for training and deploying custom TensorFlow Lite models.

So as it is clear from the comparison, there is no dominance of one solution over another. They are quite equal, and each of them is preferred for different kinds of tasks, and different development processes. It is for the developer to decide which approach to use.

VI. Conclusions

As for the conclusion, it is clear that with the development of processing power of mobile devices, more and more complicated Machine Learning tools appear on them, and become applied. There are various approaches towards applying such solutions, either involving cloud computations, or ondevice processing. It is more of a specific project question which approach is better. While the purpose of this work is to show the rapidly developing and highly demanded technology Tensorflow, with the reference to Android development on it, with Java.

References

[1] TensorFlow Official Web Site [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.tensorflow.org/about.

[2] TensorFlow Lite Guide [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.tensorflow.org/lite/guide

[3] Sharma S. TensorFlow on Mobile [Електронний ресурс] / Sagar Sharma. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/tensorflow-on-mobile-tutorial-1-744703297267.

[4] Hellgern B. ML Kit or TensorFlow: Which is best? [Електронний ресурс] / B. Hellgern. – 2020. – Режим доступу до ресурсу: https://medium.com/flutter-community/ml-kit-or-tensorflow-which-is-best-6c965e74366