Recognizing Famous Places on Android

Seminar
Practical Applications of Multimedia Retrieval
Winter Semester 2016/2017

Tim Oesterreich, Romain Granger

Supervisors:
Haojin Yang, Christian Bartz
Prof. Dr. Christoph Meinel

3. März 2017

Inhaltsverzeichnis

1 Introduction

As deep learning is becoming an increasingly big influence in everyday applications, more and more focus is put into increasing its distribution to different platforms. During the winter semester 2016/2017 a project seminar emerged, that laid focus on developing a mobile phone application for Google's Android operating system that is capable of recognizing famous sights in big cities from images taken with a smartphone camera.
Modern smartphones can in some cases outperform mid-range notebooks from a couple of years ago and, most interestingly, often have a dedicated GPU[1]. GPUs are usually used for deep learning because of their high parallelization capabilities.
Part of this seminar was the evaluation of a fairly recent deep learning framework called CNNDroid [1], which uses GPU acceleration for classification and promises substantial performance improvements compared to CPU classification.

[1] Graphics Processing Unit

2 CNNDroid

CNNDroid is an open source deep learning library for Android. It is able to execute convolutional neural networks, supporting most CNN layers used by existing desktop/-server deep learning frameworks, namely Caffe, Theano and Torch. Supported layers, as of March 2017 are:

- Convolutional Layer
- Pooling Layer
- Local Response Normalization Layer
- Fully-Connected Layer
- Rectified Linear Layer
- Softmax Layer
- Accuracy and Top-K Layer

Due to the library being open source, it is possible to add additional layers, such as batch normalization or sigmoid.
The library also supports a variety of customizations, like maximum memory usage, GPU or CPU acceleration and automatic performance tuning.

2.1 Setup and Integration into Android Project

CNNDroid is a source code library. That means that integration into an existing Android Project is fairly straight forward and doesn't require any third-party dependencies.
The only prerequisites are:

- A functional Android development environment (e.g. Android Studio[2])
- Android phone or emulator running at least Android SDK version 21.0 (Lollipop)

To integrate CNNDroid into the project, the project has to be cloned from its GitHub[3] page. Inside the '`CNNdroid Source Package`' folder, there are three folders. '`java`', '`rs`' and '`libs`'.
The '`java`' and '`rs`' folders need to be copied into the your '`app/src/main/`' directory, merging the '`java`' folders.
The '`libs`' folder has to be copied and merged into your '`app/`' directory.
For effective usage of CNNDroid, it needs read and write access to storage on the smartphone. Android policies require an application to request these permissions before the

[2] `https://developer.android.com/studio/index.html`
[3] https://github.com/ENCP/CNNdroid

app starts. These permission requests are specified inside the '`AndroidManifest.xml`' file. Inside this file, the following lines have to be added before the '`<application`' section:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

To cope with the high computational requirements, CNNDroid potentially needs a large amount of memory. Therefore, the heap has to be increased, which is done by adding

```
android:largeHeap="true"
```

inside the '`<application`' section.

Now, everything should be set up correctly and programming can commence.
After importing the package `network.CNNdroid`, the `CNNdroid` object is accesible, which can call the function `CNNDroid::classify`. This is the keyword to start execution of the trained model. The specification of the model is explained in the following section.

2.2 Structure Overview of necessary CNNDroid Files

- Layer Blob Files
- Definition File
- Labels

2.3 Convert Trained Models into CNNDroid-compatible Format

Short Introduction to MessagePack

- how to use convertion script
- compatible frameworks
- compatible layers

2.4 CPU vs GPU performance

Comparison of computation time of CPU (sequential) and GPU (parallel) mode on in-memory images using CIFAR10 (in-memory to minimize error using camera, CIFAR10 could be exchanged with CaffeNet if too fast)

3 Google Streetview Crawler

3.1 Setup of viewing parameters

Explain csv file

3.2 State of Automation (i.e. taking one image per viewing angle)

3.3 Current Limitations

Full automation not possible as of current street view API state; wrong latitude/longitude; Streetview Image API returns different images than JavaScript API (which is being used on google maps website)

3.4 Possible Improvements

Use Classifier to identify things in photsphere

3.5 Google Places API/shutterstock/Flickr

4 PlaceRecognizer Application

Overview: What does it do. Whom is it for. How does it achieve its task?

4.1 CNNDroid Integration/Image Classifier

Explain ImageClassifier Class; Including Variables that need adaption when changing Layers or DataSets
How to put msgpack on phone

4.2 Real-Time Frame Capture

How does the camera talk to the Image Classifier?

4.3 GPS Logger

How do we get GPS values and how can we integrate them?

4.4 Wikipedia Parser

How do we get the text for a classified image from Wikipedia?

4.5 How to setup project in Android Studio

5 Outlook

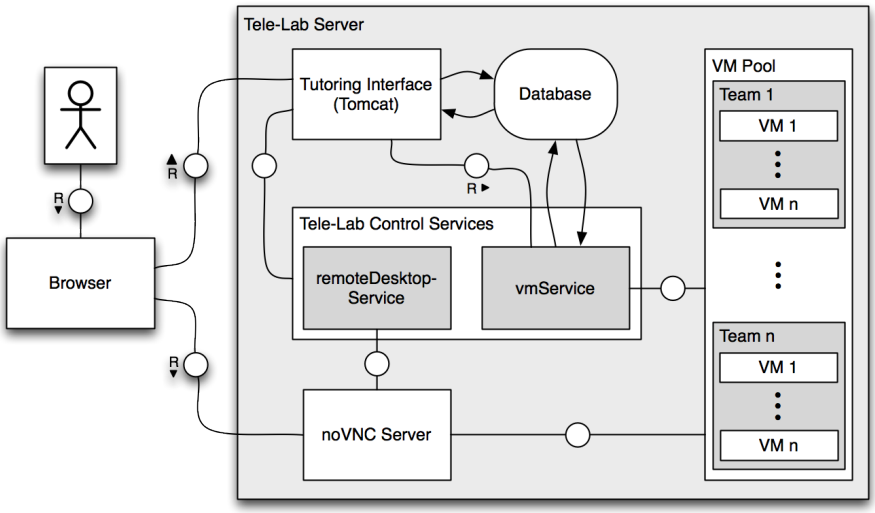

Abbildung 1: Eine Abbildung, Quelle: [1]

Literatur

[1] Seyyed Salar Latifi Oskouei, Hossein Golestani, Matin Hashemi, and Soheil Ghiasi. Cnndroid: Gpu-accelerated execution of trained deep convolutional neural networks on android. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 1201–1205, 2016.