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

March 5, 2017

Contents

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 Unit Layer (ReLU)
- 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

Of course, CNNDroid needs to know how to classify an input. For that it either uses converted binary layer files, created by a desktop deep learning framework or internal conversion functions (e.g. for pooling or ReLU).
The layer blobs are generally used for more complicated layers, such as convolutional or fully connected layers, which have a high dimensional and variable amount of variables. These files have to be converted into the MessagePack[4] format and put onto the smartphones storage.
The order in which the layers are executed, additional configurations and layers which don't need a blob file are defined inside the definition file. This file is used as a settings file for the `CNNdroid` classifier.
On creation of the `CNNdroid` object, the absolute path to this file has to be specified.
Configurations, other than the layer definitions are:

- the absolute path to the layer blob files
  (`root_directory: "/absolute/path/to/layer/files"`)
- the maximum amount of RAM the classifier should use on the smartphone
  (`allocated_ram: Amount_in_MB`)
- whether or not automatic performance tuning should be used
  (`auto_tuning: "on|off"`)
- whether or not classification should be GPU accelerated (if possible)
  (`execution_mode: "sequential|parallel"`)

[4] http://msgpack.org/

Layers are defined as shown in figure 1.

```
root_directory: "/sdcard/Download/berlin_sights_data/"
allocated_ram: 100
auto_tuning: "off"
execution_mode: "parallel"

layer {
  type: "Convolution"
  name: "conv1"
  parameters_file: "model_param_conv1.msg"
  pad: 2
  group: 1
  stride: 1
}
```

Figure 1: Excerpt of a CNNDroid definition file

Not essential for the execution of CNNDroid, but very helpful for processing the final results, is a labels file. This file should specify human-readable names for the extractable classes. The order inside this file should be mapped to the output order of the last layer (probably a fully-connected layer), i.e. the first value of the output array should correspond to the first class name inside the labels file.

2.3 Convert Trained Models into CNNDroid-compatible Format

As mentioned before, CNNDroid uses a format called MessagePack for the layer definitions. MessagePack is a binary serialization format for exchanging data, comparable to JSON[5], but other than JSON it is not human-readable. The omission of this constraint makes it possible to reduce size and increase parsing speed of the data stored inside. This is especially important due to the limited storage that most smartphones posess, especially compared to big servers, where deep learning is usually executed.
Figure 2 illustrates how MessagePack saves storage space by using types (JSON encodes everything as strings, so for example, 'true' uses four Bytes, while it only uses one Byte using MessagePack) and dropping control symbols, like the curly braces {}.

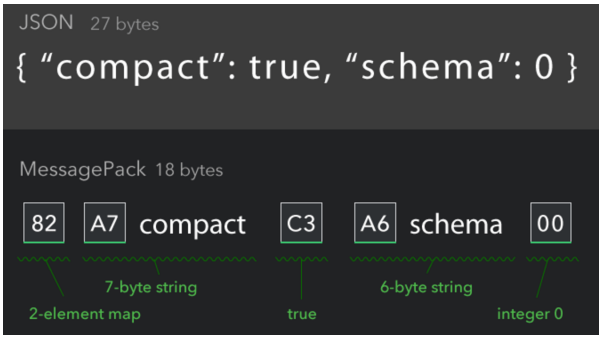

Figure 2: Comparison JSON and MsgPack

[5]JavaScript Object Notation

In order to convert models from the aforementioned desktop/server deep learning frameworks, CNNDroid provides conversion scripts. The script for Caffe is written in Python and requires the packages `numpy` and `msgpack` to be installed.
Three variables need to be defined. The absolute path to the trained model, the deployment prototxt file, which specifies the parameters for the layers, and the saving path.
The output are a couple of files in MessagePack format. The definition and labels file has to be created manually, still. For the definition file, most of the content of the prototxt deployment file can be used. Only some parts of the syntax has to be adapted to a CNNDroid parsable format.

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

Classification of real-world images requires a very big dataset for the learning process beforehand. Google Streetview provides 360 degree images from many places in all German urban areas. Using sophisticated crawling techniques, a huge amount of image data can be extracted from these, so called, 'Photospheres'.
This chapter describes how we used the Google Streetview and other image APIs to crawl images from famous sights in Berlin.

3.1 Image Crawler

We provide a Google Streetview image crawling python script inside the `streetviewcrawler` folder. This script takes a csv file as input, which specifies the parameters the crawler needs, such as position and viewing angle. Chapter 3.2 explains the contents of this file. All requirements for the crawler are specified in the `requirements.txt` file and can be installed with the command `$pip install -r requirements.txt`.
Given a location in longitude and latitude, the image crawler will automatically create jpeg-images from different viewing angles. These angles can either be specified or automatically generated by calculating the angle between two geo-coordinates. In the latter case, the script will generate images in five degree steps from 30 degrees to the left to 30 degrees to the right of the calculated viewing angle. If specified by hand, we use one degree steps from the starting to the end angle.
Because photospheres can change or get removed, the API uses the closest photosphere to the location provided. This might not be the location specified in the csv file, which is why we also check the metainformation of each photosphere for the coordinates connected to the image and the status code. In case there is no image in a certain area, the status code tells us that and we don't need to go through all the requests.
To prevent Denial-of-Service attacks, the API uses a time- and request-based blocking system. That means, that an IP might be denied further downloads if it either makes request too fast or it reaches the maximum amount. To counter the first blockage, our script uses an exponential wait mechanism. If requests were made too quickly, it waits for a short while before retrying. If it still gets no result, the wait time is increased. This is repeated until the blockage is lifted.
For the second part, we provide the usage of a Google API key[6], which can be passed to the crawler script with the `-key {key}` command line switch. The API key enables the crawler to request 25000 requests per day for free, with the option to increase the volume against money.

3.2 Setup of viewing parameters

Explain csv file

[6] https://developers.google.com/maps/documentation/streetview/

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

3.4 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)

3.5 Possible Improvements

Use Classifier to identify things in photsphere

3.6 Google Places API

For our project, we use several API from google which help us get mainly two things. Complementary informations about a particular place such as: Name of the place, adresse of the place, phone Number, Web site, rating from google reviews.

- PLACE DETECTION API

The place detection api will allow us to discover the place near where the device is located. By places, it means all places registered on google including local businesses, points of interest, and geographic locations. This api will return a maximum of 10 probable places. We can retrieve informations like adresse, phone number or web site. Another very interesting use is that it can help us determinate if our prediction is relevant, by comparing our classification with the results from this functions.

- GEO DATA API

The Google Places API provides more informations about a place, including the place's reviews and address, the geographical location specified as latitude/longitude coordinates, the type of place (such as night club, pet store, museum) and more. The only constraint is to use a specific place ID from google, which you can get from the Place detection API.

- How to use it for android

First step, you need to get the api key from google "https://developers.google.com/?hl=fr". To get the API key, you need to register on the developper console from google. This platform is the access to every API provide by the firm. Then, you are free to activate any

Second step, you need to declare in the android manifest function your API key between application tag and the first <Activity> tag.

```
<meta-data
android:name="com.google.android.gms.version" android:value="@integer
    google_play_services_version"/>
<meta-data
android:name="com.google.android.geo.APIKEY" android:value="API_KEY"/>
```

Then, for the third step, you need to ask permissions in the android manifest for internet and GPS location.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Fourth step, you should declare a "GoogleMapApi" builder and add the several API you want to use. In our case, and illustrate by the following snippet, we use PLACE$_{D}ETECTION_{A}PIandGEC$

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addApi(Places.PLACE_DETECTION_API)
                .addApi(Places.GEO_DATA_API)
                .enableAutoManage(this, GOOGLE_API_CLIENT_ID, this)
                .build()
```

For the final step, it's calling the function. You need to use the "Places" package associate with the api you want to use. Then the most important thing is to add your google api builder as a parameter of the function.

```
#For the PLACE DETECTION API
Places.PlaceDetectionApi.getCurrentPlace(mGoogleApiClient, null);

#For the GEO DATA API
Places.GeoDataApi.getPlaceById(mGoogleApiClient, placeId)
```

- Limitations

Google provide a really useful and powerful service but the only constraint is the limitation. We can only have 1,000 free requests per 24 hour period, which is already great but not for a multiple user application. Therefore, they provide a business API plan that can be adapte if needed which give access to 150 000 request per day and can be increased more depending on the number of requests needed.

3.7 Shutterstock/Flickr and Pinterest

Shutterstock/flickr and Pinterest are public web application for hosting and sharing photos and images. For exemple on Flickr nearly 7,000 photos are uploaded per minute. As these services are used by both professional and amateurs photographers, we can find a lot of great and good quality images. It's easy to create an API key, just register on the web site of each applications.

- Exemple to get images URL

Our exemple for getting Image url which can be downloaded after use angularJs API package and store them into a mongoDb database.

First we declare variable for our api connection

```
var api = shutter.v2({
    clientId: 'Your public key',
    clientSecret: 'Your secret key',
});
```

Then we detail the search string we want to crawl.Here we choose the Branderburg gate.

```
var opts = {
    query: 'Branderburg',
    page: 1,
    per_page: 200,
    sort : 'popular'
};
```

Finally, we navigate through the json response to get the url.

```
    api.image.search(opts, function(err, data) {
        if (err) throw err;

        console.log(data.data[0].id);
        var arrayLenght = data.data.length;
        console.log(arrayLenght);

        for (var i = 0; i < arrayLenght; i++) {

        var obj = data.data[i];
        console.log(obj);
        }
```

The response return will be as follow.

- Limitations

These services are limited to 3600 call per hours. And your API key can be block if the service detect an abuse use of the service.

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

When our CNNDroid model has come with an image class, we now want to get informations about the place. Wikipedia represent one of the largest source of informations and has a big and active community which update regularly all the informations. Also, the list of features we can grab from wikipedia is long: Descriptions, images, literature etc.. Also, compared to other API in a technical way, Wikipedia API present three mains advantages:

- The use is made by a simple url call
- There use no use limitations or constraint
- It provides a Json response easily parsable by any program.

The wikipedia API is composed by a principal URL which is "https://en.wikipedia.org/w/api.php".Then you can easily change the language of the response by changing the subdomain of the URL. For exemple to get a french Json content, you can call "https://fr.wikipedia.org/w/api.php". Like our application will be useful for tourists, we need to adapt our content to their language. Then, the principal URL will be enrich with paramters, like the "format" (xml,json,html) or the "titles" to get informations about a specific page. We will see in the second part of this section which parameters we use to call the wikipedia API.

In our program we use two classes:

- HttpHandler.java

The first class, "HttpHandler.java" is here to create a temporary array of bytes from the url and also catch error to check if it is correct and if it returns a value. In a second time it will convert the response stream as a string value. So first, we aim to use the http GET method to open a connection.

```
try {
    URL url = new URL(reqUrl);
    HttpURLConnection conn = (HttpURLConnection)
    url.openConnection();
    conn.setRequestMethod("GET");
```

Then we will catch the errors, for the following cases:Incorrect input url, Protocol exception, Input/Output error during the use of the InputStream() function, or if the response is an empty array of bytes.

```
    } catch (MalformedURLException e) {
            Log.e(TAG, "MalformedURLException: " + e.getMessage());
        } catch (ProtocolException e) {
            Log.e(TAG, "ProtocolException: " + e.getMessage());
        } catch (IOException e) {
            Log.e(TAG, "IOException: " + e.getMessage());
        } catch (Exception e) {
            Log.e(TAG, "Exception: " + e.getMessage());
```

- GetWiki.java

This class is a public class using an asynchronous task. In fact, our classifier will give a class to the image taken by the user, for exemple "Brandenburg Gate" or "Fernsehturm Berlin". Then this label will be used as the input string variable for our GetWiki class. First, the call is made from the mainActivity class, with the following statement:

```
wikipediaInfos = new GetWiki().execute("classLabel")
```

"classLabel" is a string variable, which is dynamically update with the class given by the classifier. So it is also important to give a reliable class label regarding wikipedia when the model is trained, because this label will be used as a parameter in the url which is called:

```
String urlTitle = strings[0];
String url =
    "https://en.wikipedia.org/w/api.php?format=json&action=query&prop=extracts&exintro=&explaintext=&ti
    + urlTitle;
```

The result will consist in a Json array that we need to parse to extract part of the information we want. An expemple of response look like the following:

```
{"batchcomplete":"","query":{"pages":{"156604":{"pageid":156604,"ns":0,"title":"Brandenburg
    Gate","extract":"The Brandenburg Gate (German: Brandenburger Tor) is an
    18th-century neoclassical monument in Berlin, and one of the best-known
    landmarks of Germany. It is built on the site of a former city gate that
    marked the start of the road from Berlin to the town of Brandenburg an der
    Havel....\n"}}}}
```

We truncated the response to keep only the variable we actually use like the title and the description. As you can the response give us several Json levels/objects. We will

use the JsonObject package given in the Android API 25, to parse our Json response. In our example, we would like to get all informations contained the 156606: object. The problem is that we can't guess the Id of the page, so we will do like this:

```
JSONObject jsonObj = new JSONObject(jsonStr);
JSONObject query = jsonObj.getJSONObject("query");
JSONObject pages = query.getJSONObject("pages");
Iterator<String> keys = pages.keys();
String pageId= keys.next();
JSONObject page = pages.getJSONObject(pageId);
String title = page.getString("title");
String extract = page.getString("extract");
```

First, we get the response into a JsonObject class variable. Then we can use string parameters to navigate through the Json file. If we have an object that is not static, like the Id of the page, we use an Iterator keys which will return the value for this Json Object, here it will give us the pageId (156606) as a string. Then we just have to say that we want to go to the next value of the json array using keys.next().
Finally, we get the string value of the response and store them into variables.

4.5 Text to speech

To enhance the user experience of our app, as it is used to recognise famous places, we had a look on how to allow the user to play an audio description of the place while is watching it. Android SDK provide a very useful package:

```
import android.speech.tts.TextToSpeech
```

which has built in methods to parse a string value and read it as an audio track. It can easily be implemented with few lines, following these steps:

First, you need to declare a TTS (text to speech) object, and set the attributes. Let's say in our exemple that "t1" is our object. Many attributes can be changed, like the languages, the speed of the audio, the voice type.We declare in the following snippet the variable 't1' as a new text to speech object in our class context, giving the langage Locale.UK

```
t1=new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
                if(status != TextToSpeech.ERROR) {
                    t1.setLanguage(Locale.UK);
                }
            }
        });
```

Then, you need to get the string variable you want to convert as an audio file. In the following snippet we get the wikipedia description of Branderbuge Gate from our wikipedia API parser (see the previous part)

```
String toSpeak = null;
try {
toSpeak = new GetWiki().execute("Brandenburg_Gate").get().descritpion;
```

```
} catch (InterruptedException e) {
  e.printStackTrace();
} catch (ExecutionException e) {
  e.printStackTrace();
}
```

Finally, you can use the speak() method from your object, with the queue mode (QUEUE FLUSH) which means that media to be played are dropped and replaced by the new entry each time you call this method.

```
t1.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, null , "test");
```

4.6 Firebase API

Firebase is a google service which provide a realtime database and backend as a service. The service is really well integrated with Android. The data is stored on Firebase's cloud and the Firebase SDK for android comes up with several method to store and synchronised in real time all the data.

- How we use it

First, we declare in the build.gradle at the app level the use of Firebase service.

```
compile 'com.google.firebase:firebase-core:10.0.1'
compile 'com.google.firebase:firebase-database:10.0.1'
```

Then, like the googlePlaceAPI, we need to settle an object of type database to start the connection with our database instance.

```
private FirebaseAuth mAuth;
private FirebaseAuth.AuthStateListener mAuthListener;
DatabaseReference database = FirebaseDatabase.getInstance().getReference();
```

The Firebase structure use path to store the data. For exemple, you will have the top level path "users", then each users of you app, and for each users, some features. To store or read data, the method will look like this:

```
database.child("users").child(userId).child("places").push("location");
```

You need to define all the levels you want to have access to. Here we first access the "users" node, then getting the current user ID, and we upload the locations details of his visit into the "places" node.

4.7 How to setup project in Android Studio

5 Outlook

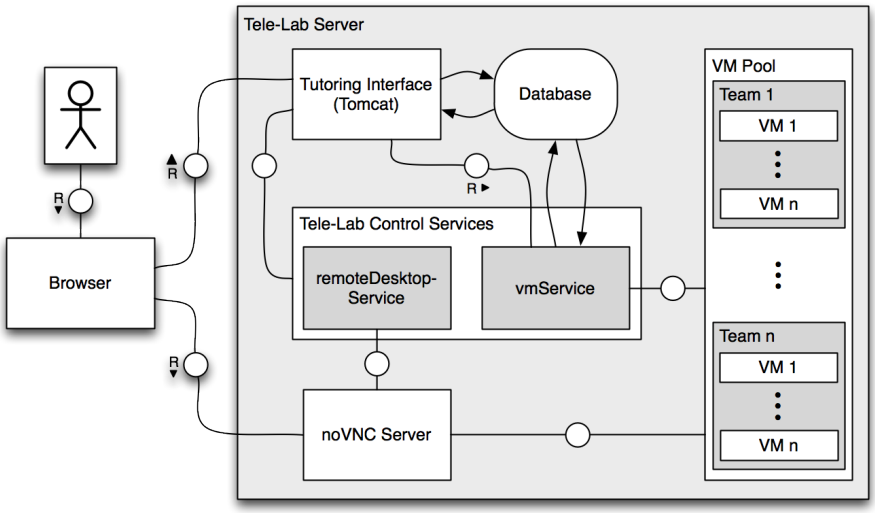

Figure 3: Eine Abbildung, Quelle: [1]

References

[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.