Department 07
Master Computer Science

Deep learning - Dog Breed Classification

Realization of an native Android app using deep learning algorithms

Alice Bollenmiller, Andreas Wilhelm
WS 17/18 IG
January 25, 2018

Contents

1. Introduction

In this chapter, a short introduction leads to the subject of Deep learning. Furthermore, the scope of this work and its points of referencee are described and localized.

1.1. Deep learning

In 2015, AlphaGo - a computer program developed by Google's DeepMind Group that was trained to play the strategic board game Go - was the first program to defeat the multiple European champion Fan Hui under tournament conditions. Back then, terms like Artifical Intelligence (AI), Machine Learning and Deep Learning were talked of, because those were the reason why a computer program was able to defeat a human being. One of the most frequent used techniques of AI is Machine Learning. It uses algorithms to parse data, process it and learn from it. The result is a prediction or determination as a conclusion of what was learnt from the dataset. Deep Learning - which is part of the Machine Learning techniques - sells its application particularly in the field of language and image processing.

In 1958, Frank Rosenblatt introduced the concept of the perceptron which is the fundamental idea of all Deep learning approaches. It consists of multiple artifical neurons which are coupled with weights and biases. In the case of a single-layer perceptron, the input nodes are fully connected to one or more output nodes. During the learning process the weights are adapted according to the learning progress. When such a structure is extended with layers, it becomes a multi-layer perceptron (MLP). This basic structure can be found in special neural network architectures e.g. Convolutional Neural Networks (CNN). CNNs which are frequently used for object detection and image/ audio recognition etc. consists of multiple neurons. When a neuron receives an input, it performs a dot production and adapts the weights. Because of their special structure, CNNs are able to detect local properties of an image. Basically, the network represents a differentiable score function which is applied on the raw image pixels and computes the class scores as an output.

1.2. Terms of Referencee

The problem of Deep learning architectures is their high performance requirements regarding computational power. Common frameworks for open source development in the field of deep learning which are discussed in section 2.1 introduced several models for the integration in mobile applications.

In order to overcome the above mentioned problem, optimizing methods will be combined with appropriate models which require less computing power and are suitable for mobile application integration. As a result of this work, a dog breed analyzer will be implemented. This mobile app will take a live camera stream as an input and determine the breed of the focused dog. The three highest probabilities of breeds will be shown by the app.

2. Methodological fundamentals

This chapter describes the most frequently used frameworks in deep learning for developing applications. Furthermore, common models for deep learning are introduced followed by suitable models for mobile integration. The chapter closes listing key requirements for an appropriate dataset which increase the quality of the training results.

2.1. Common Frameworks for Deep Learning Applications

The demands on neural networks increases with the complexity of problems to solve. Concurrently, there's an expanding offer of deep learning frameworks with a varity of features and tools. The most common used ones are represented in the following section.

2.1.1. Tensorflow

In 2015, the Google Brain Team introduced the most popular deep learning API Tensorflow which is an open-source library for numerical computation. Its current version 1.4.1 was released on December 8th, 2017. Tensorflow is primarily used for machine learning and deep neural network research. Based on the programming language Python, Tensorflow is capable of running on multiple CPUs and GPUs. Furthermore, C++ and R are supported by Tensorflow. Another feature is the possibility to generate models and export them as .pb file which holds the graph definition (GraphDef). The export is done by protocol buffers (protobuf) which includes tools for serializing and processing structured data. When loading a .pb file by protobuf, an graph object is created which holds a network of nodes. Each of those nodes represent an operation and the output is used as input for another operation. This concept enables an user to create self-built tensors. To simplify the usability, Tensorflow developed a high-level wrapper of the native API which is calles Tensorflow Slim. Futhermore, in order to run Tensorflow on performance critical devices like e.g. mobile devices there are two lightweight solutions of Tensorflow are available: Tensorflow Mobile and Tensorflow Lite. The latter one is an evolution of Tensorflow Mobile and still in developer mode. But both are predestinated for integration in mobile applications.

2.1.2. Keras

In order to simplify the utilization of Tensorflow the Python based interface Keras can be configured to work on top of Tensorflow. It allows building neural networks in a simple way and is part of Tensorflow.

2.1.3. Caffe

Another deep learning library is Caffe which was developed by Berkeley AI Research (BAIR). Based on C++ or Python, it focuses on modeling CNNs. An main advantage of Caffe is the offer of pretrained models available in the Caffe Model Zoo.

2.1.4. Torch and PyTorch

Besides Tensorflow and Caffe, Torch is another common deep learning framework. It was developed by Facebook, Twitter and Google. Based on C/C++, Torch supports CUDA for GPU processing. Like above mentioned frameworks, Torch facilitates the building of neural networks. The Python based version of Torch is available through PyTorch.

2.2. Common Models in Deep Learning Applications

- short differences between different architecuteres (?, CNN, RNN)
- AlexNet, Mobilenet, Inception, VGG, -¿ short decsription, useCases, important things, differences

2.3. Qualified Models for mobile App Integration

- Mobilenet, Inception etc -¿ short decsription, useCases, important things, differences

2.4. Key requirements for an appropriate dataset

Supervised learning tasks such as image recognition are based on operations where an output is taken as an input for the next node. Every raw pixel input is taken to compute an intermediate representation - a vector containing all learned information about the dataset. As a consequence, the training results are only as good as the dataset itself. For better accuracy its important to train a model on a variety of images for each object which should later be classified by the model. It's recommended to take images of an object which were taken at different times, with different devices and at different places. Otherwise, the model will concentrate on other things like for example the background instead of details about the object itself. Therefore, a huge dataset is required especially for non pre-trained models. Training a model from scratch will require a huge dataset, a lot computing power and time. Whereas pre-trained models only require a small dataset of about hundreds of images. For that reason, a pre-trained model will be used in this work.

3. Concept

First, this chapter describes the selection of the appropriate framework. Futhermore, the structure of the model which was used for classification is explained based on its architecture. The chapter closes with the class diagram of the mobile application.

3.1. Frameworks

As a deep learning framework Tensorflow was used to retrain the model. This decision was mainly based on recommendations. Even companies like e.g. NVIDIA Corporation, Intel Corporation etc. use Tensorflow. It is one of the common frameworks for deep learning applications and also provides solutions for integration in mobile apps. Furthermore, Tensorflow provides a variety of tutorials for working with neural networks. Beside those advantages, there is a large community about Tensorflow talking about issues and solutions.

To run the tensor within a mobile application, the first approach was to use Tensorflow Lite which is still in development state. But many attempts resulted in corrupt models which caused the app to terminate. Because of this experiences Tensorflow Mobile was used to optimize the model for app integration.

3.2. Model based Architectures

Andi - general architectures of models -¿ n Mobilenet, Inception

3.3. Application based Architecture

Alice

4. Realization

In general, this chapter describes the methodical procedure of solving the above mentioned problem section 1.2. After describing the used dataset all required software and hardware components are explained in detail. Furthermore, the chapter leads through the installation steps of Tensorflow and the setup of Android Studio. Followed by the installation process the retraining of a pre-trained model is depicted. Afterwards, the re-trained model is tested and validated. The chapter ends with the description of the realization of the Android app.

4.1. dataset

Andi

4.2. hardware environment

Andi used CPU, GPU -¿ NVIDIA, handys

4.3. installation of software

Andi This chapter includes all necessary steps for installing the software environment including Tensorflow.

4.3.1. Prerequisites

The software environment was set up on the Linux distribution Ubuntu 16.04 LTS. To install the software environment for Tensorflow Python is required. Therefore, the current version of Python 3.6 was installed by default. Tensorflow also supports Bazel which was installed by following command.

```
sudo apt-get install openjdk-8-jdk

echo "deb [arch=amd64] http://storage.googleapis.com/bazel-apt stable jdk1.8" | sudo
  tee /etc/apt/          sources.list.d/bazel.list
curl https://bazel.build/bazel-release.pub.gpg | sudo apt-key add -

sudo apt-get update && sudo apt-get install bazel

sudo apt-get upgrade bazel
```

Listing 1: Bazel Installation

Futhermore, the package and environment management tool Anaconda was installed by the following steps: First, the Anaconda installer was downloaded from `https://www.anaconda.com/download/#linux`. During the installation process the prompts were answered by the default suggestions except the following prompt: "Do you wish the installer to prepend the Anaconda3 install location to PATH in your /home/aw/.bashrc ? [yes—no]". "yes" was typed in and conda was tested using the "conda list" command.

As an app development environment the free IDE Android Studio in its version 3.0.1 was installed by downloading from `https://developer.android.com/studio/index.html`, extracting and following the instruction steps. Required dependencies are installed by Android Studio itself. So, the SDK in the version 26.1.1 was used.

In the development of native Android Apps Java is used as the programming language. Within the installation of Anaconda, the JDK in the version 8 was installed.
- Android system, CUDA, CUDNN

4.3.2. Tensorflow based on Python

4.3.3. Tensorflow based on Bazel

- e.g. Workspace changes for Android SDK, msse4.2

4.3.4. Installing Android Studio and its Delevopment Kit

- also possible with bazel but easier Android studio (needs correct versions of sdk, ndk)
- SDK, NDK
- IMPORTANT: tf versions updaten (same as trained)

4.4. building the models

Alice bis Steps, Andi ab Optimierung, time GPU/CPU -¿ evtl extra subsubsection:
- execution methods -¿ Bazel and Python (incompatible versions)
- Mobilnet -¿ steps, optimierung
- Inception -¿ steps, optimierung
- time related differences of execution
-¿ time CPUs/GPU

4.5. Output Tests and Validation

Andi - test pictures and if it works -¿ label image
- validation script?! –¿ in Evaluierung

4.6. Implementation of an native Android App

Alice - list all necessary things to do (e.g. tensorflow version, Interpreter -¿ load Model)

4.7. Deployment and Validation

Alice

5. Evaluation

Andi - prio von nierdig zu hoch
- regarding implementation time
- regarding performance
- regarding quality in accuracy
- handy perfomance?

6. Conclusion

- tutorials not complete, different
- which model is better
- prospects, improvements, Recommendations

Beispiele frs referenzieren:
In Figure 1 ist das HS Mnchen Logo zu sehen.
Oder auch eines Codes wie in listing 3.

```
1
2 bottleneck_path_2_bottleneck_values = {}
3
4
```

Figure 1: FH-Logo

```
def create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor):
  """Create a single bottleneck file."""
  tf.logging.info('Creating bottleneck at ' + bottleneck_path)
  image_path = get_image_path(image_lists, label_name, index,
                              image_dir, category)
  if not gfile.Exists(image_path):
    tf.logging.fatal('File does not exist %s', image_path)
  image_data = gfile.FastGFile(image_path, 'rb').read()
  try:
    bottleneck_values = run_bottleneck_on_image(
        sess, image_data, jpeg_data_tensor, decoded_image_tensor,
        resized_input_tensor, bottleneck_tensor)
  except Exception as e:
    raise RuntimeError('Error during processing file %s (%s)' % (image_path,
                                                                 str(e)))
  bottleneck_string = ','.join(str(x) for x in bottleneck_values)
  with open(bottleneck_path, 'w') as bottleneck_file:
    bottleneck_file.write(bottleneck_string)
```

Listing 2: Some python code

Sectionrefs: In section 2 ist vieles noch nicht fertig.
SubSectionrefs: In section 2.1 wird dann nher auf den Inhalt eingegangen.
SubSubSectionrefs: In section 4.3.2 gehts ans eingemachte.

Beispiele frs zitieren:
Fr einen noch besseren berblick, kann das Buch von **?** oder auch andere refs wie **?** hinzugezogen werden (**?**).
Wenn in klammern und Seitenzahl (**?**, p. 3)

als compared, aber ohne Seitenzahl (cmp. **?**)
als compared mit Seitenzahl, das nd heit "no date", da keine Jahrezahl vorhanden (cmp. **?**, p. 5)

List of Figures

A. Anhang

A.1. Anhang vom Anhang