

**EGI-Engage**

Integration of assisted pattern recognition tools

6.1

|  |  |
| --- | --- |
| **Date** | 06 November 2015 |
| **Activity** | WP6 |
| **Lead Partner** | CSIC |
| **Document Status** | DRAFT |
| **Document Link** | https://documents.egi.eu/document/XXX |

Abstract

This deliverable addresses the technical point of exploring the integration and deployment of pattern recognition tools on EGI specific resources, including for example servers with GPUs or other relevant hardware for image/sound recognition.

**COPYRIGHT NOTICE**



This work by Parties of the EGI-Engage Consortium is licensed under a Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/). The EGI-Engage project is co-funded by the European Union Horizon 2020 programme under grant number 654142.

**DELIVERY SLIP**

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Name*** | ***Partner/Activity*** | ***Date*** |
| **From:** | Eduardo Lostal, Francisco Sanz | BIFI/WP6 |  |
| **Moderated by:** |  |  |  |
| **Reviewed by** |  |  |  |
| **Approved by:** |  |  |  |

**DOCUMENT LOG**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Issue*** | ***Date*** | ***Comment*** | ***Author/Partner*** |
| **v.1** | 22/10/2015 | Document creation | Eduardo Lostal / BIFI |
| **v.1.2** | 23/10/2015 | Added Executive Summary, Introduction and Feedback on Satisfaction sections | Eduardo Lostal / BIFI |
| **v.1.3** | 26/10/2015 | Added Service architecture section and statistics | Francisco Sanz / BIFI |
| **v.1.4** | 26/10/2015 | Added Release notes and Future Plans sections, provide statistics on Feedback on Satisfaction section | Eduardo Lostal / BIFI |
| **v.1.5** | 26/10/2015 | Revision | Eduardo Lostal, Francisco Sanz / BIFI |

**TERMINOLOGY**

A complete project glossary is provided at the following page: <http://www.egi.eu/about/glossary/>

**Contents**

[1 Introduction 5](#_Toc434587548)

[2 Service architecture 6](#_Toc434587549)

[2.1 High-Level Service architecture 6](#_Toc434587550)

[2.2 Integration and dependencies 7](#_Toc434587551)

[3 Release notes 8](#_Toc434587552)

[3.1 Functional Requirements 8](#_Toc434587553)

[3.2 Non-Functional Requirements 9](#_Toc434587554)

[4 Feedback on satisfaction 9](#_Toc434587555)

[5 Future plans 12](#_Toc434587556)

**Executive summary**

This deliverable deals with the task of exploring the integration and deployment of pattern recognition tools on EGI specific resources from a technical point of view. It includes the research on hardware and software solutions for image/sound recognition. After a short introduction, the architecture of the proof-of-concept prototype is presented along with the integration and dependencies of the solution. Fulfilled requirements are listed and briefly detailed followed by some numbers and comments on the performance of the abovementioned prototype. Some future work is proposed to continue the improvement of the current development

Some frameworks were tested as a result of the state of the art research on pattern recognition and convolutional neural networks systems. Caffe was the selected to develop the prototype. It is Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center. Some datasets have been tested on a GPU optimized for the libraries used by Caffe. Results are good enough but some foreseen work is needed to improve their performance.

It is possible to conclude that the framework selected works properly for the goals of the prototype. It has been deployed in the available servers running the tests over a particular GPU over which the used libraries are optimized.

# Introduction

The present deliverable addresses the search and testing of a framework that may be used for image recognition on the cloud. Therefore, it encompasses several issues going from the research on the state-of-the-art, the selection of the tool, its deployment, preparing the tests and evaluating its performance. Besides being able to learn, that is, the framework must include a neural network able to be trained, it must be possible to run in on the cloud taking advantage of its benefits.

Research of the computer vision and deep learning tools raised two frameworks that suit the requirements and could be eligible for testing. On one hand, Pastec that is an open source index and search engine for image recognition based on OpenCV[[1]](#footnote-1). It was tested on a server at BIFI facilities resulting on an easy installation and use with acceptable results. After some doubts about its performance, main responsible for Pastec development were contacted who confirmed that it is general purpose framework that would not work for the requirements of the project. On the other hand, the second framework and, eventually, the one that was chosen for its use, is Caffe[[2]](#footnote-2) a deep learning framework that uses Convolutional Neural Networks and that is prepared to be deployed on the cloud.

As well as this framework, an app developed for Android phones was developed as a proof-of-concept. It accesses to a web service through an API that allows user to make classification requests to the network.

The remaining of this document presents the architecture of the framework, requirements list and details about its performance.

|  |  |
| --- | --- |
| **Tool name** | Caffe |
| **Tool url** | *Please provide url if applicable* |
| **Description** | Caffe is a deep learning framework made with expression, speed, and modularity in mind. |
| **Customer of the tool** | *Please describe customer of the tool. Customer commissions the service provider to receive the service, doing so on behalf of a number of users – specify the organisation type/category of the service e.g. NGI; RI; Resource Provider* |
| **User of the service** | *Large research groups; individual researcher* |
| **User Documentation** | <http://caffe.berkeleyvision.org>  |
| **Technical Documentation** | <https://github.com/BVLC/caffe>  |
| **Product team** | BIFI |
| **License** | *Please provide license* |
| **Source code** | *Please provide url* |

# Service architecture

The Caffe framework provides two main services, with the first one, *train*, a new model can be trained, with the second one, providing one (or several) image(s) it(they) can be classified according to the trained model. The first service, is a highly computing bound task, and it can be speed-up using GPU (CUDA). The second one, is a more I/O bound task, so it can run without GPU in an efficient way.

As said previously Caffe is a deep learning framework than can run both in CPU or GPU[[3]](#footnote-3), the speed-up using GPU versus CPU is x9 (using the AlexNet model), moreover, one can find x34 speed-up GPU + CuDNN v3[[4]](#footnote-4). In order to use the CuDNN v3 library a GPU with Maxwell (4th generation) or Kepler (3rd generation) architecture is required.

## High-Level Service architecture

Both services can be implemented easily in the cloud, although we deployed the train service both in a non-cloud host and a cloud host.



Figure 1 Basic architecture

For the first service, training the model, it is highly recommended a Maxwell or Kepler GPU with at least 4GB of RAM. We implemented this service both in cloud and non-cloud hosts. We want note that as said in the official documentation, a portable version of Caffe can be compiled using the command *make portable.*

The second service was deployed only in a non-cloud host, but it can be easily launched in the cloud, as it is mainly a web service pointing to the Caffe binary in order to do the image classification. The Caffe framework provides also a web-service written in flask to allow web classification. We developed a new web-service to offer a API to be used, for example, with our android app described below.

## Integration and dependencies

Installing Caffe is a straightforward task if one follows the official documentation[[5]](#footnote-5). All the requirements and steps are published in the previous page. Here is listed the main requirements:

* CUDA is required for GPU mode.
	+ library version 7.0 and the latest driver version are recommended, but 6.\* is fine too
	+ 5.5, and 5.0 are compatible but considered legacy
* BLAS via ATLAS, MKL, or OpenBLAS.
* Boost >= 1.55
* protobuf, glog, gflags, hdf5

Optional but highly recommended dependencies:

* cuDNN for GPU acceleration (v3)

Some others dependencies in order to use the python tools can be fulfilled following the official documentation using the pip tool. Moreover, there is a receipt to build an AMI with Caffe to be launched in the cloud in the following URL: <https://github.com/BVLC/caffe/wiki/Caffe-on-EC2-Ubuntu-14.04-Cuda-7>

In order to train a new dataset, we used the oxford-102 approach, maintaining two different folders, one for the Caffe framework itself, namely $CAFFE\_HOME and the second one that holds:

* the dataset itself
* the scripts to manage this dataset, that is, splitting the images into the test group, val group and train group
* The model to be used to train, mainly the train\_val.prototxt and deploy.prototxt files explained below.

# Release notes

## Functional Requirements

* The framework accepts picture as input for its classification
* The framework’s output provide the five more accurate results as the classification
* Every result in the output comes with a number that indicates the certainty that the framework has on that result
* The framework can be deployed to run over a cloud infrastructure
* The framework can be deployed on GPUs
* The framework can be accessed as a web service
* When the framework is accessed as a web service, it must allow users to upload the pictures
* The framework allows to train the models
* Android app uploads the pictures to the web service
* Android app displays the output of the framework in the phone
* Android app manages the connection to the server (the framework)
* Android app allows to take pictures to be uploaded
* Android app allows to choose pictures from the phone gallery to be uploaded

## Non-Functional Requirements

**Performance**

* Classification time should be under 20 seconds
* Accepted input picture formats are at least JPG and PNG

**Security**

* Source code must be available at a Github account

# Feedback on satisfaction

The framework was tested through several datasets to assess its performance. First data set was a control one to ensure the proper behaviour of the tool. It was chosen a dataset from the Oxford Robotics Research[[6]](#footnote-6) already tested on Caffe. Then, replicating their steps, it would be possible to learn how the platform works and check that it gives as a result the expected output. Installation and test of this dataset was successful and the accuracy obtained was higher than the 95%, as expected according to Oxford results. To achieve that results, a pre-trained model from Imagenet dataset was used over which a fine-tuning, that consists in more training over the previous model with the images of the new data set, was applied.

The steps giving shape to the roadmap for making use of new datasets can be extracted from this example. Given a new data set, first of all documents for training must be created. train\_val.prototxt and deploy.prototxt (names may vary, but these are commonly used) contain the definition of the neural network with all the layers. In both of them, at least the last layer must be modified to update the number of outputs to the new data set. Since final users are not expected to have expertise on Neural Networks, the rest of the layers of the network can remain as before. In a nutshell, intermediate layers are trained to recognize colours, shapes, edges, etc., while final layers are the ones that orchestrate all that information to choose the correct (estimated) output. That is why it makes sense to reuse previous definition of the network. It will be also necessary to update the solver.prototxt that is the file with the definition of the model, learning rates, etc. and again it is advisable modifying it as less as possible. Following the example of the Oxford data set fine-tuning, global learning rate is reduced, while the final fully connected layer learning rate is higher than the others. In this document, it can be also set the number of iterations. That number is relevant since too less iterations mean the network is not properly trained, but too many iterations may get an over-fit network. Besides the synset\_words.txt document with the string of the output, train.txt and test.txt files must be also prepared. They contain the address of an image per line along with the proper output. Those files are used to train and test the network. A third similar file may be created with the pictures for validation (not used by the framework during training). Thus, the framework can be evaluated through not familiar pictures. Once all these files are ready, it is possible to train the network.

The second data set tested was one from Portuguese flora. It is made up of almost 1890 pictures of which 63 genus were used as outputs. Aiming to build a consistent data set, genus with less than 30 pictures were discarded. Those 30 pictures were divided equally into the three files: train, test and valid. An algorithm was used to unsort the lines of those files what improves the final accuracy of the model since modifications during learning are doing for every genus incrementally, while if the whole modifications are done at the beginning of the learning for one genus, subsequent changes on the network that do not include that genus will produce their features getting forgotten. First attempt for training got an over-fit network after more than 400000 iterations. Subsequent trainings were done following two different models of fine-tuning based on Oxford works. First one, it was realized through AlexNet approach getting around 51% of accuracy after 20000 iterations in less than three hours with a tiny stabilized loss. Second one, over VGGS approach getting around 56% of accuracy after 20000 iterations in less than three hours. As an example, in the following it is displayed a snapshot of the GPU resources and performance during one of the trainings (statistics are similar for both approaches).



Figure 2Output of nvidia-smi during training

As shown in illustration 1, training the Portuguese dataset average consumption with the GTX960 is placed between 110 and 130W while the memory usage on the GPU is around 1591MiB.

Previous results are considered reasonably good. It is worth mentioning that contents of the data set are one of the most significant aspects that may increase (or decrease) the final accuracy. For instance, the Portuguese data set contains a small set of samples for each output (used 10 pictures for training and 10 for testing, although this splitting is the recommended by oxford-102). Moreover, pictures of the same genus are very distinct among them (some contains only the flower, some the whole plant from farther distance, some of the stalk…) what makes harder for the framework to extract features of the plant to be classified. Keeping in mind those considerations, given the provided framework, building a consistent and goo data set will affect significantly on the final accuracy of the framework.

The third dataset was provided by the Real Jardín Botánico[[7]](#footnote-7). It consists on 662 different images, but only 6 genus has more than 20 images. We followed the same algorithm than the previous dataset to split the images into three groups, 10 for training, 10 for testing and all the rest to validating. After 9000 iterations we reached 70% of accuracy, but we don't consider this test as a valid one due to the small amount of different images.

---- HERE WILL BE SOME NUMBERS COMPARING THE DIFFERENT GPUs, DATASETS, ACCURACY ---

The developed app for Android phones allows the user to take a new picture or choose one from the phone gallery to be uploaded to the server where an instance of the framework has been deployed. Server returns the output of the framework that it is displayed on the phone. The app works fine as a proof-of-concept. The Apk takes 1.205.229 bytes (1,2 MB) on disk. When using WiFi upload time for the picture is very reduced, while using the data plan upload may take up to 50 seconds. Connection time is very small, as well as the output sending time.

Three different github repositories were created along the project life, one for the android application available at:

* <https://github.com/Ibercivis/CaffeUseExample>

And two more for the source code of the two datasets scripts.

* <https://github.com/EGI-Lifewatch-CC/orchidee>
* <https://github.com/EGI-Lifewatch-CC/portuguese-flora>

# Future plans

The final goal of this part of the project is to demonstrate the viability of using cloud infrastructures for deep learning frameworks and how final users will benefit from that kind of deployment. Therefore, future work is focused on that direction. Current work has been carried out as a proof-of-concept, a prototype of what can be done. Use of the cloud will allow increasing scalability moving to better trained networks with a higher level of complexity. Once the model is working properly, next step is its deployment on the cloud and study its performance.

Oriented to get a tool useful for the final user, a web-based tool would be advisable as an entry-point to the framework automating and hiding low-level details to the user. In addition to this tool, Android app could be improved focusing the efforts in reducing the time for pictures uploading to the server.

Finally, as above-mentioned how good a data set is determines significantly the final accuracy of the network. Thus, some effort should be done in addressing the assessment of the data sets ensuring that they are suitable for training a network providing good results.

1. <http://www.pastec.io> [↑](#footnote-ref-1)
2. <http://caffe.berkeleyvision.org> [↑](#footnote-ref-2)
3. More precissely, one needs a CUDA GPU [↑](#footnote-ref-3)
4. <https://developer.nvidia.com/cudnn> [↑](#footnote-ref-4)
5. <http://caffe.berkeleyvision.org/installation.html> [↑](#footnote-ref-5)
6. <https://github.com/jimgoo/caffe-oxford102> [↑](#footnote-ref-6)
7. <http://www.rjb.csic.es/jardinbotanico/jardin/> [↑](#footnote-ref-7)