EGI-Engage

Integration of assisted pattern recognition tools

D6.1

Date	27 November 2015
Activity	WP6
Lead Partner	CSIC
Document Status	FINAL
Document Link	https://documents.egi.eu/document/2647

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. The described demonstrator provides a classification service on flower species through image recognition. The images can be uploaded by the user through a Web interface or Android app. The service returns the probable latin names of the flower, and the probability ratio for the name-hits. The described demonstrator was developed and deployed in EGI by the LifeWatch Competence Centre of the EGI-Engage project.

The EGI-Engage project is co-funded by the European Union (EU) Horizon 2020 program under Grant number 654142 http://go.egi.eu/eng

COPYRIGHT NOTICE

DELIVERY SLIP

	Name	***Partner/Activity***	***Date***
From:	Eduardo Lostal, Francisco Sanz	BIFI/WP6	27/11/2015
Moderated by:	Gergely Sipos	MTA SZTAKI/ WP6	25/11/2015
Reviewed by	Eric Yen Mario David	ASGC/Taiwan LIP/LifeWatch RI	17/11/2015
Approved by:	AMB and PMB		3/12/2015

DOCUMENT LOG

Issue	***Date***	***Comment***	***Author/Partner***
v.0.1	22/10/2015	Document creation	Eduardo Lostal / BIFI
v.0.2	22/11/2015	External review changes	Eduardo Lostal, Francisco Sanz /BIFI
FINAL	27/11/2015	Final version	Francisco Sanz /BIFI, Jesus Marco/CSIC

TERMINOLOGY

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

Contents

Executive summary

This deliverable deals with the technical tasks of exploring and deploying pattern recognition tools on EGI specific resources. The demonstrator serves the LifeWatch research infrastructure community by offering a tool that provides value for biodiversity researchers and citizen scientists. 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 numbers and comments on the performance of the above mentioned prototype. Future work is proposed to continue the improvement of the current development.

Several frameworks were tested as a result of the state of the art research on pattern recognition and convolutional neural networks systems. Caffe was selected to develop the prototype. Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center. Training using several datasets has been tested on GPU resources optimized for the libraries used by Caffe. The results are promising but further work will be needed to improve the final performance.

We can conclude that the selected framework seems adequate for the goals of the prototype. It has been deployed in the available e-infrastructure, running the tests over a particular GPU for which the used libraries are optimized.

1 Introduction

The participation of citizens in scientific activities is a great opportunity to, simultaneously, improve our projects and make society aware of their importance. In particular, in the context of biodiversity and ecological research, citizen science can become a very relevant instrument.

The idea behind the work presented here is to offer a better support to citizens contributing observation records, in particular those submitting images, i.e. photos, of living species taken in any place in the planet. The final service will provide an initial classification of the images into species. The images will be uploaded by the user through a Web interface or Android app, and the service will return the probable identification using its scientific name, and the probability ratio for the name-hits.

For the initial prototype it was agreed to start using flower species that have already an existing database available to train the identification algorithm.

This initiative has been advised by the Spanish GBIF Node (LW-JRU-ES /Real Jardín Botánico-CSIC), which holds ample experience in the field.

Cloud sites from the EGI infrastructure provided us with capacity to train the deep-learning Cafee module with sample images, and then also for hosting the Web services and its Web front-end for users.

LifeWatch will greatly benefit of this initiative to extend the coverage, both spatial and in time, of many species, thanks to the collaboration of citizens: the experts that provide a final ok to the identification of the observed record will have a pre-classification that can be very helpful. Notice also that most of the images will be directly geo-tagged using the GPS available in mobile phones or cameras, so the records can be very useful.

As stated, the present deliverable addresses the search and testing of a framework that may be used for image recognition in 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. The following two objectives should be met:

- The framework must include a neural network able to be trained.
- It must be possible to run in the cloud, allowing users to deploy easily their own model using GPUs.

Research on computer vision and deep learning tools raised two frameworks that suit the requirements and could be eligible for testing:

- Pastec that is an open source index and search engine for image recognition based on OpenCV[R1]. 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 a general purpose framework that would not work for the requirements of the project.

- The second framework and, the one that was chosen for its use, is Caffe[R2] a deep learning framework that uses Convolutional Neural Networks and that is prepared to be deployed on the cloud.

The demonstrator service was developed in the form of a web service with a Web front-end. Besides the Web front-end a proof-of-concept app was developed for Android phones too, providing an alternative interface for users. This app accesses the web service through an API allowing the user to perform classification requests to the neural network.

The remaining of this document presents the architecture of the framework, requirements list and provides details about the performance evaluation that was performed on the system.

Tool name	Assisted pattern recognition tools integrated with EGI for citizen science
Tool url	Current version: http://lxbifi21.bifi.unizar.es:5000/ Final instance will be deployed for production under a different URL.
Tool wiki	https://wiki.egi.eu/wiki/EGI-Engage:WP6_Caffe
Description	A demonstrator service built on the Caffe deep learning framework to classify flowers based on uploaded images.
Customer of the tool	LifeWatch
User of the service	Research groups; individual researchers
User Documentation	About the Caffe deep learning framework: http://caffe.berkeleyvision.org
Technical Documentation	About the Caffe deep learning framework: http://caffe.berkeleyvision.org
Product team	BIFI (Eduardo Lostal - eduardol@bifi.es, Fran Sanz -frasanz@bifi.es)
License	Released under the BSD 2-Clause License
Source code	https://github.com/BVLC/caffe ,(source code) https://github.com/EGI-Lifewatch-CC,(training samples)

2 Service architecture

The Caffe framework provides two main services:

- *"train":* a new model can be trained.
- *"classify"*: providing one (or several) image(s) it(they) can be classified according to the trained model.

The first service, is a compute intensive task, and it can be speed-up using GPU (CUDA). The second is I/O bound task, so it can be run without GPU in an efficient way.

As previously stated, Caffe is a deep learning framework that can run both in CPU or GPU[1], the speed-up using GPU versus CPU is a factor of 9 (using the AlexNet model[2]), moreover, one can find 34 times speed-up GPU + CuDNN v3[R4]. In order to use the CuDNN v3 library a GPU with Maxwell (4th generation) or Kepler (3rd generation) architecture is required.

2.1 High-Level Service architecture

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

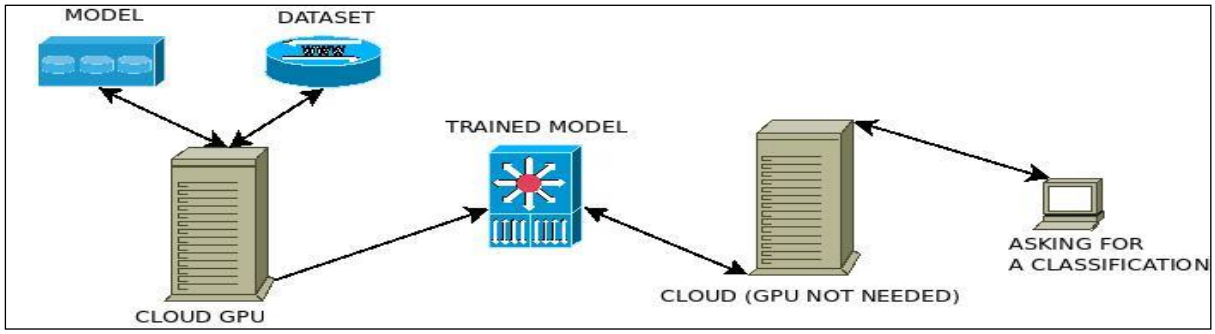

Figure 1: System architecture

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 to note that as stated 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 [R5] 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.

2.2 Integration and dependencies

Installing Caffe is a straightforward task if one follows the official documentation [R6]. All the requirements and steps are published in this official documentation web page. The main requirements are:

[1] More precisely, one needs a CUDA GPU

[2] We are using a slightly modified version of the ILSCR 2012 winning AlexNet[R3]

- CUDA is required for GPU mode.
    - library version 7.0 and the latest NVIDIA driver version are recommended. CUDA 6.* can also be used.
    - 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 are satisfied through the python tools following the official documentation of the pip [R7] tool. Building an AMI(Amazon Image) with Caffe to be launched in the cloud, is described 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 training model: train_val.prototxt and deploy.prototxt files are explained below.

3 Requirements

3.1 Functional Requirements

- The framework accepts pictures as input for its classification.
- The framework's output provides the five more accurate results as the classification.
- Every result in the output comes with a number that indicates the certainty with which 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 training 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 taking pictures to be uploaded.
- Android app allows choosing pictures from the phone gallery to be uploaded.

3.2 Non-Functional Requirements

Performance

- Classification time should be under 20 seconds.
- At least JPG and PNG input picture formats should be accepted.

Availability

- Source code must be available at a Github account.

4 Feedback on satisfaction

The framework was tested with three datasets to assess its performance. This section provides an overview of these tests and their outcomes.

The first dataset was a control sample to ensure the proper behaviour of the tool. A dataset from the Oxford Robotics Research [R8] was chosen since it was already tested on Caffe. Replicating their steps, allowed to learn how the platform works and check that it produces the expected output.

Installation and test of this dataset was successful and the accuracy obtained was higher than 95%, as expected according to Oxford results. To achieve those results, a pre-trained model from the Imagenet dataset was used for fine-tuning. It consists in training over the previous model with the images of the new dataset.

The steps giving shape to the roadmap for making use of new datasets can be extracted from this example, given a new data set:

1. Using oxford-102 files as a template, 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 dataset. 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

dataset 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.

2. 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 dataset tested was one from Portuguese flora. It is a set of almost 1890 pictures of which 63 genus were used as outputs. Aiming to build a consistent dataset, genus with less than 30 pictures was 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 improving 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.

The first attempt for training got an over-fit network after more than 400000 iterations. Subsequent trainings used two different models of fine-tuning based on Oxford works:

- AlexNet approach results in 51% of accuracy after 20000 iterations in less than three hours with a small loss in stability.
- VGGS approach results in 56% of accuracy after 20000 iterations in less than three hours.

Fig. 2 shows a snapshot of the GPU resources and performance during one of the trainings (statistics are similar for both approaches).

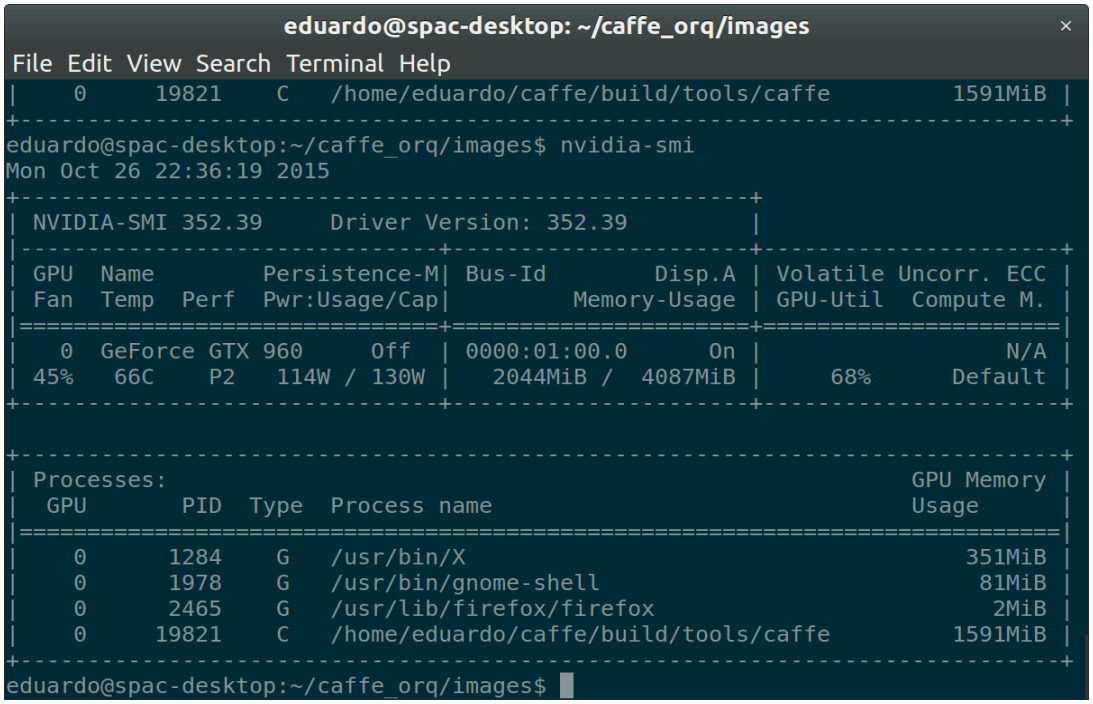

Figure 2: Output of nvidia-smi during training

It can be seen that training the Portuguese dataset the average consumption with the NVIDIA GTX960 is placed between 110 and 130W while the memory usage on the GPU is around 1591MiB.

Previous results are considered reasonably well. It is worth mentioning that contents of the dataset are one of the most significant aspects that may increase (or decrease) the final accuracy. For instance, the Portuguese dataset 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 contain only the flower, while others 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 good dataset will affect significantly on the final accuracy of the framework.

The third dataset was provided by the Real Jardín Botánico [R9]. It consists on 662 different images, but only 6 genus have more than 20 images. We followed the same algorithm previously described to split the images into three groups, 10 for training, 10 for testing and the remaining for validation. After 9000 iterations an accuracy of 71% was achieved. This test was not considered as a valid one due to the small amount of different images.

	Oxford-102		Portuguese Flora		RJB Orchid	
	AlexNet	VGG_S	AlexNet	VGG_S	AlexNet	VGG_S
Real User Sys	50m59.201s 76m10.100s 7m49.956s	133m38.909s 153m51.840s 20m52.008s	52m38.373s 59m56.548s 2m43.432s	125m42.119s 161m39.728s 18m55.464s	47m16.132s 54m6.912s 2m38.708s	125m18.208s 154m51.032s 19m11.284s
Accuracy (iter #)	0.91 (10000)	0.94242 (10000)	0.516923 (10000)	0.558462 (10000)	0.71 (9000)	0.72 (4000)
GPU Memory Usage	1593MiB	3787MiB	1591MiB	3784MiB	1588MiB	3781MiB

Table 1: GTX 960 benchmark, 10.000 Iterations

The app developed 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. The server returns the output of the framework that it is displayed on the phone. The app works correctly as a proof-of-concept. The Apk uses 1.205.229 bytes (1,2 MB) on disk. When using WiFi, the upload time for the picture is much 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 for the source code of the two datasets scripts.

https://github.com/EGI-Lifewatch-CC/orchidee

https://github.com/EGI-Lifewatch-CC/portuguese-flora

5 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 this type 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, the next step is to deploy 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, an 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 dataset is determines significantly the final accuracy of the network. Thus, some effort should be done in addressing the assessment of the datasets ensuring that they are suitable for training a network providing good results.

6 References

R1	http://www.pastec.io
R2	http://caffe.berkeleyvision.org
R3	http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks
R4	https://developer.nvidia.com/cudnn
R5	http://flask.pocoo.org
R6	http://caffe.berkeleyvision.org/installation.html
R7	https://pypi.python.org/pypi/pip
R8	https://github.com/jimgoo/caffe-oxford102
R9	http://www.rjb.csic.es/jardinbotanico/jardin/