Deep Learning Implementation Framework for Image Classification Applications

Alexandros Frangiadoulis
Technological Educational Institute of Crete
Heraklion, Crete
alekospj@gmail.com

Tsampikos Kounalakis
Aalborg University
Copenhagen, Denmark
tkoun@m-tech.aau.dk

Georgios A. Triantafyllidis
Aalborg University
Copenhagen Denmark
gt@create.aau.dk

Nikolaos Vidakis
Technological Educational Institute of Crete
Heraklion, Crete
nv@ie.teicrete.gr

***Abstract*— In this paper, we present the development of a novel deep learning implementation framework for image classification applications. This work focuses on the transition between pure experimental deep learning methods, towards deep learning based applications that can be used from any user. Our proposed framework consists of two parts, the CNNs-Tester a deep network training desktop application and a web-application called PaternF. The desktop application is able to use sets of images depicting individual categories in order to train a convolutional neural network (CNN) model for image categorization, that can be used by our proposed web-application. When using our user-friendly web environment, the user can perform online image categorization by uploading any query image. We experimentally show the capabilities of the developed application and present a use-case that supports our claims for the user-friendly capabilities for our novel deep learning implementation framework.**

I. INTRODUCTION

Deep learning algorithms [1] represent the latest research in artificial intelligence and machine learning. These algorithms provide the required infrastructure, thus yielding state-of-the-art results in many research problems. Their capabilities draw the attention of researchers from sciences, other than computer science. Especially, many sciences are dependent from the use of imaging systems including, but not limited to, medical imaging [2], plant identification [3] and many others.

Current deep learning training systems have improved over the years, the development of deep learning methodologies. There are available deep learning libraries [4][5][6][7], that can be only implemented as functions inside a computer program. Standalone graphical training tools [8][9][10] are also available, usually implementing deep learning libraries. However, the aforementioned systems are usually targeted for computer scientist. As a result, the advantages of deep learning are not fully functional from non-specialist.

In this work, we focused in the design and development of a deep learning framework suitable for non-familiar users. Our novel framework implements deep learning visual recognition introducing a desktop and a web-based application. The desktop application termed CNNs-Tester, allows the user, to use images from a local directory in order to train a user-defined deep learning model.

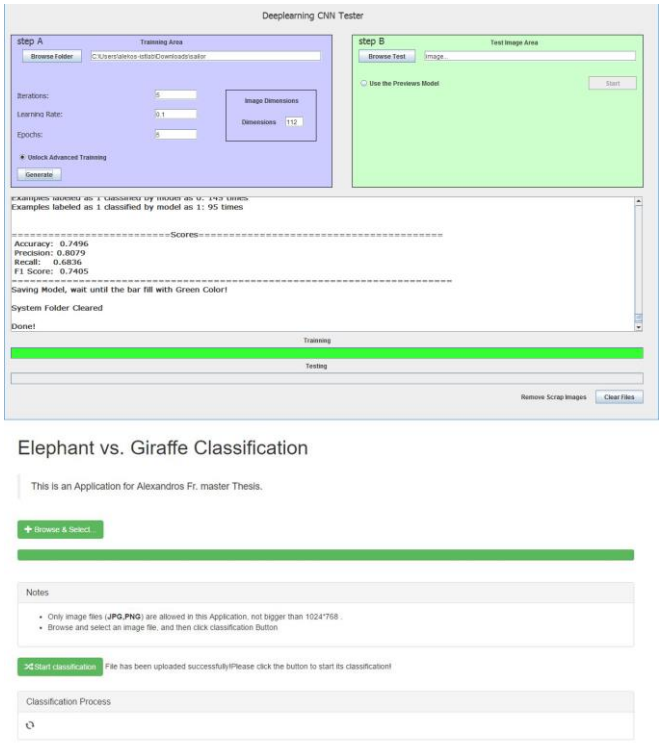

Figure 1: CNNs-Tester & PaternF

Subsequently, the web-application termed PaternF, uses the trained model from the CNNs-tester enabling image recognition. The simple and very clean designing for both our applications, results to a very user friendly graphical interface.

The proposed framework will contribute:

- To make researchers, other than computer scientists, to deep learning imaging algorithms.
- A tool for introducing the technology of deep learning for junior scientists.
- The creation of an implementation framework, in order, other developers can implement their own applications.
- Compatibility on multiple platforms due to its Java-based implementation.

The rest of the paper is organized as follows, in Section II, we review the related work. In Section III, we present our desktop application system. Our web-based application is presented in Section IV. In Section V, we describe the advantages of our proposed framework, and present an example use case. Conclusions are drawn in Section VI.

II. RELATED WORK

As described in the introductory section, the development of deep learning created a need for tool needed for research and implementation.

A. Deep Learning training methods.

Caffe [5] is a Deep Learning framework developed from Berkley University. It is designed in a physical way, and can uses CPU and GPU for the training. Models and network architectures can be accomplished without hard computing, i.e, using predefined functions. However, this is not a framework for non-familiar users because it requires knowledge of computer programming.

MatConvNet [6] is a toolbox for Mathworks Matlab, a program which many engineers are familiar with. The disadvantage with MatConvNet, is that due to the Matlab environment, all design and work it is mainly restricted inside the program. Only with advanced programming processes the models can be transfered to other platforms.

Theano [7] is a deep learning library implemented in Python. It is able to provide the use of CPU or GPU depending on the project. However, the use of Python makes this library restrictive to its use from users, other than programmers.

To summarize, all these above deep learning tools are not using a graphical interface. So, it is very difficult to use from non-specialist users.

B. Web-Based Deep Learning Methods

DIGITS [9] is a deep learning GPU Training system developed by NVidia. It is a tool specialized in experiments of image classification and object detection tasks. DIGITS allows the user to develop deep learning architectures and observe the results in real time, using some of the aforementioned libraries. The user can perform many experiments with the visual aid of monitoring results. However, DIGITS focus on designing and training networks rather than creating applications.

TensorFlow [10] is an open-source software library powered by Google. Using this software the user can easily depict an architecture through graphs and perform a plethora of experiments. With a flexible architecture experiments can be deployed to one or more CPU or GPU in a desktop, server or even a mobile device with an API. Similarly to Nvidia DIGITS, TensorFlow gives the functionality of designing and supervise a deep learning architecture.

But an with Nvidia DIGITS these applications does not allow the user to construct his/her own application in a simplistic way. As described in the ensuing sections, our implementation framework allows user interaction in such a way, where non-expert users could learn and create their own deep learning applications.

III. DESKTOP APPLICATION CNNS-TESTER

Our goal was to create a user-friendly and easy to understand application, for visual deep leaning algorithms, i.e, Convolutional Neural Networks (CNNs), thus performing their user-defined experiments.

As a result, we developed a multi-threaded java application able to construct CNN models. Our proposed desktop application implements a Java-based deep learning library called Deeplearning4j [8]. The Java implementation provided us the compatibility with various systems and operating systems.

A. CNNs-Tester Training State

When CNNs-Tester is initiated the user can see the starting frame (Fig.1). In order to start the training phase, user can select a desired directory contained labeled images. The directory must include different sub-folders for images from each category. At this point, users can select a simple training with standardized settings or to unlock advanced training setting where users can customize CNN training. Then the CNN model will be generated by clicking on the button *Generate.* After the click of the button the training process bar is starting to fill up, and the text area will provide additional details for the CNN training state. When training is complete (Fig.2) the text box will indicate information about the training results. The provided information include statistics for CNN Accuracy, Precision, Recall and F1 Score.

Accuracy is model's accuracy during training, **Precision** is the positive prediction for the classifier, **Recall** explains how possible is it for the classifier to detect a positive example, and

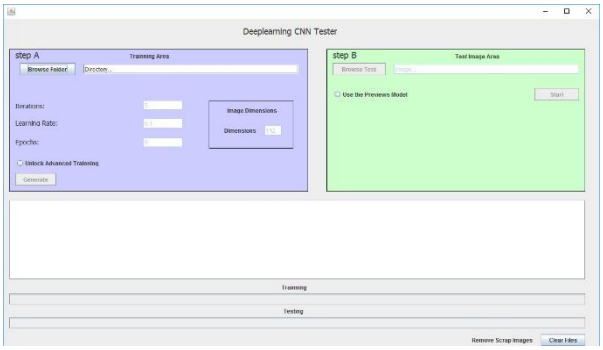

Figure 2: Starting Frame CNNs-Tester

F1 score is a number between zero and one, which provides the quality of the training. The closer we get to one, the better the training. Actually, F1 score is defined by the average of precision and recall [8].

After the training of the first CNN model, a folder is created at the user's local disk where all models are stored for later use. Inside the folder the user can will find some files describing the trained CNN model. These files can be used for the migration of a CNN model to another system or used from the CNNs-Tester and PatternF for further testing.

A. CNNs-Tester Testing State

After the off-line CNN training (Fig.2), the user may retrain the resulting model or perform a test with a new image unknown to the trained model. For CNN model testing, the button *Browse Test* must be selected. The user may search through directories in order to test a single image. After the image is selected, the Start button is unlocked automatically and by clicking it, the testing will start. The testing process is completed as indicated from the testing process bar (Fig.3). If user desires, can perform another test after completion of the first one. When the program opens again, it can used a previous model and perform another classification test.

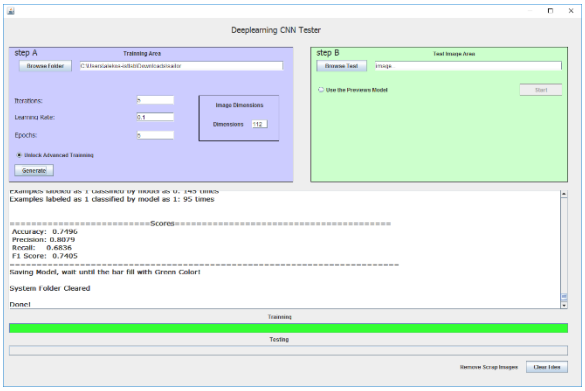

Figure 3: CNNs-Tester Training finished

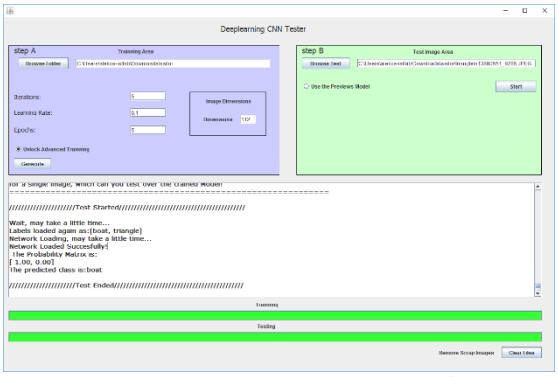

Figure 4: CNNs-Tester training and testing finished

IV. WEB-APPLICATION PATERNF

Figure 5: PaternF starting page

In order to enhance user experience and functionality of our proposed method we provide a very simple web-based application termed PatternF. The user can use this application to upload image selected from local directory. When the image uploads to the application is able to send a request for image classification from a CNN model. The result of image categorization will be shown in the text area with the name of the category which was predicted (fig.5). Then, if it is desired, the user can upload another image and repeat the same steps.

A. PatternF Testing State

In order to start the testing phase with our proposed web-application the user has to upload an image using the button *browse & select.* When the examined image is selected, a servlet in the background will upload it to the server automatically. When the image is successfully uploaded to the server, the uploading process bar is filled and the button *start classification* becomes available.

Figure 6: PaternF image uploaded and classification finished

The examined image is resized automatically from the server. The server is then starting the re-construct of the network architecture from the files in the server's directory. These files, are those which the CNNs-Tester created after a model training. Then, the pre-trained model in the server side, creates a testing process resulting to a probability matrix. The system will choose the biggest value inside this matrix, which corresponds to the more likely category, and this is what we return as a categorization result.

PaternF application serves as a demo application, where other developers or junior scientists can develop their own web-based applications.

V. EXPERIMENTS WITH CNNS-TESTER

A. *Experimental Setup*

During the development of our proposed applications, it became apparent that we needed evaluation for the functionality of our system. For this reason, we decided to conduct some experiments with the use of our proposed CNNs-Tester. We looked for image databases suitable to conduct supervised learning [11] experiments. We ended up with the following databases.

Simplicity is an image Database from the research team of professor Jia Li [12]. It contains 2360 images, and it constructed for image retrieval and annotation.

Figure 7: Images from SIMPLIcity

NEC Animal is an image database consisted of 5000 photos of 60 different types of toys animals with 72 images for each category [12], it is a multi-capture database, in which they use the same item for captures in different angles.

Figure 8: Images from NEC Animal [12]

Final**, Rabbit** is an image database compiled from 22 objects, with an average of 30 different images per category.

Figure 9: Images from Rabbit database

Then, we started experiments using different settings for epochs, iterations and learning rate in the CNNs-Tester. It should be noted, that the split between training-set and test-set is automatically happening from the CNNs-Tester, and it uses 80% of the images for training, with the remaining 20% for testing.

A. *Framework experimental assessment.*

First, we tried 5 different experimental setups for the SIMPLIcity image database [11] with the results presented in Table 1. We can see that better results are achieved, from using an increasing number of iterations. Also, from the number of epochs we can understand that the results are not optimized, with just repeating the same procedure. The learning rate is the same at all experiments, because we think that would be best for all occasions.

Experiment	Accuracy	Precision	Recall	F1 Score	Epochs	Iterations	Learning Rate
1	0.0798	0.1593	0.1022	0.1245	1000	1	0.1
2	0.3417	0.2988	0.3535	0.3239	10	1	0.1
3	0.3786	0.3699	0.4117	**0.3897**	5	1	0.1
4	0.3119	0.3842	0.344	0.363	3	1	0.1

Table 1: SIMPLIcity image database results

It is commonly accepted and reasonable that, every dataset has different needs and customization in the experimental settings. The question is, which customization will be the best. This one is going to be found only by experimenting on the data. In the table 2 we can see NEC Animal dataset [12] results.

Experiment	Accuracy	Precision	Recall	F1 Score	Epochs	Iterations	Learning Rate
1	0.2132	0.2876	0.102	0.1506	10	10	0.1
2	0.149	0.1666	0.0947	0.1207	10	1	0.1
3	0.1833	0.1242	0.0658	**0.**086	10	100	0.1
4	0.2426	0.3049	0.1914	**0.2352**	20	1	0.1
5	0.2352	0.3115	0.1883	0.2347	30	1	0.1

Table 2: NEC Animal dataset results

In the table 2, as we can see for the NEC animal image database, we achieved the better results for 20 epochs and 1 iteration, this shows once again how different is the data from an image dataset to another. In this occasion we pass 20 times all the data from the network in the training procedure for better results.

To conclude, we used the image database Rabbit, which contains 18 different classes of images and the three-dimensional information for each image. For this experiment we only used two-dimensional images. We notice that with same number of iterations, but with less epochs (number the network initialized all the images) we have better results. This is mainly due to the quality of the base and the number of classes. Also, observe that after some point the results stop being good, similar to previous databases.

Experiments	Accuracy	Precision	Recall	F1 Score	Epochs	Iterations	Learning Rate
1	0.3639	0.4322	0.3236	0.3701	10	5	0.1
2	0.2891	0.5478	0.3181	**0.4025**	5	1	0.1
3	0.2463	0.3196	0.2781	**0.**2974	20	1	0.1
4	0.335	0.3732	0.3272	**0.**3487	30	1	0.1
5	0.1007	0.1007	0.1007	0.1007	50	1	0.1

Table 3: Rabbit dataset results

Regarding the above experiments, we can observe that every image-database has its own settings, and with parameter customization we can achieve optimal results for each examined dataset. We also noticed the existence of one more-beneficial parameter for each dataset, and after this point the results are not optimal. Finally, it is good to note again that in this work does not presents CNN architecture performance but the capabilities of the framework itself.

B. Framework Evaluation one use cases

When a user wants to use our applications, the first thing is to open the CNNs-Tester. After this, user can browse into local files to find the appropriate directory containing an image dataset. At this point if the user can customize the CNN model training. Next, the user can start the training and waits until it finishes. When the training stops, one trained model exists in the application's directory. Also, statistics for the training can be seen in the CNNs-Tester text area. Now, the user can just may simply record the statistics, or to continue the test procedure. In the beginning of testing procedure an image must be select. When selected, the user can start the testing procedure, and after a few second the system will return the category of the examined image. Finally, our proposed web-application can function as an online CNNs-Tester, providing a demonstration for the user to create a web application on his own.

II. CONCLUSION

In this paper we achieved the development of novel deep learning implementation framework for image classification applications. The framework we proposed, has two parts., CNNs-Tester a deep learning desktop application and a web application called PaternF. The CNNs-Tester, can handle image databases to conduct training on deep learning models. Also, it is useful for experiments with statistics for the quality of the training and independently for an image over the trained model. Subsequently, we implement a web-application, which uses a pre-trained model from the first application, and achieve to gives image identification results. Finally, the designing of all this work, considered as an implementation framework, because of the open-source character which gives to the user the ability to learn from our work.

III. REFERENCES

[1] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow and A. Y. Ng, "On Optimization Methods for Deep Learning," 2011.

[2] B. Menze and Z. Tu, "Medical Computer Vision: recognition techniques and applications in medical imaging," in *1st MICCAI-MCV workshop*, 2011.

[3] A. Kadir,, L. Edi, L. E. Nugroho,, A. Susanto and P. I. Santosa, "Performance Improvement of Leaf Identification System," *nternational Journal of Advanced Science and Technology*, vol. 44, 2012.

[4] "www.teglor.com," [Online]. Available: http://www.teglor.com/b/deep-learning-libraries-language-cm569/. [Accessed 2016].

[5] "caffe.berkeleyvision.org," 2016. [Online]. Available: http://caffe.berkeleyvision.org/.

[6] "/www.vlfeat.org," [Online]. Available: http://www.vlfeat.org/matconvnet/. [Accessed 2016].

[7] "deeplearning.net," [Online]. Available: http://deeplearning.net/software/theano/.

[8] "deeplearning4j.org," [Online]. Available: http://deeplearning4j.org/. [Accessed 2016].

[9] "developer.nvidia.com," [Online]. Available: https://developer.nvidia.com/digits. [Accessed 2016].

[10] "www.tensorflow.org," [Online]. Available:

https://www.tensorflow.org/. [Accessed 2016].

[11] "wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Supervised_learning. [Accessed 2016].

[12] "www.stat.psu.edu," [Online]. Available: http://www.stat.psu.edu/~jiali/index.download.html. [Accessed 2016].

[13] "ml.nec-labs.com," [Online]. Available: http://ml.nec-labs.com/download/data/videoembed/. [Accessed 2016].