DEEP LEARNING FRAMEWORK, FOR IMAGE CLASSIFICATION APPLICATIONS

by

ALEXANDROS FRANGIADOULIS

B.A. Technological Institute of Crete, 2014

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF APPLIED INFORMATICS
AND MULTIMEDIA

SCHOOL OF APPLIED TECHNOLOGY

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

2016

Approved by:

Dr. Nikolaos Vidakis

Abstract

In this master thesis, we design and construct a Deep-Learning Framework, which is a model for building image classification applications. We demonstrate the “CNNs Tester” and a web-application structure. “CNNs Tester” uses images to create a trained model, after that the model can be used in the web-application’s classification procedure. This Thesis started as a need for a transition, from purely experimental work to something more practical. The results are two applications a) CNNs Tester, b) PatternF. These two applications can be also considered as generic framework.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to thank my supervisor Dr. Nikolaos Vidakis for all the support throughout my Master Thesis. Second, Dr. Tsampikos Kounalakis for his advice during my years as an undergraduate and graduate student, which was critical for the selection of my research area. Also, Dr. Triantafyllidis Georgios and Dr. Georgios Papadourakis, who provided valuable perspective to my goals with their guidance. I would also like to thank ISTlab crew, in particular Antonios Providakis, Dimitrios Maramatakis and Georgios Ktistakis for the ongoing support. Finally, I would like to extend my Special thanks to Dr. Evelyn-Eleni Minis, Panagiota Xatzi and to my family and friends, who always support me.

Dedication

To my grandfather to whom I owe my name

1. Introduction

1.1 The Problem

In this project, we start with one basic problem and then we go deeper, we want an application which could create Deep Learning models. Using images given from user, combined with training statistics too. Not especially using common databases, but in most occasions images collected from the user.

For this purpose, we decided to use Convolutional Neural Networks [1] (CNNs) and Supervised Learning [2], with scope to create an accessory Deep Learning tool, both for familiar and non-familiar users.

The second problem came up, was that we wanted to demonstrate the efficiency of this technology, so our problem divided to two smaller problems, first, we needed to construct an application to resolve the first issue (training of images) second demonstrate the technology to prove the efficiency. The try to resolve the above problems, described to the next pages.

1.2 Goals

The goals of this Master thesis are as follows:

- Greater knowledge in technology of Deep Learning on Computer Vison matters.
- Manufacturing a tool to assist researchers of Computer Vision, for better knowledge of the quality of an image-database.
- The smooth introduction of junior scientists in the art of Deep Learning, to better understand how it works in its totality.
- Widespread use of the described tool, for people who want to test the efficiency of image-databases, such as researchers from other sciences (Biologists, Architects, Artists).

- As described to the problems, to design and construct one framework, in order to demonstrate the usability of a pre-trained model from an application.
- To create a web-application using a pre-trained model with (CNNs) [1], which facilitates recognition between categories of objects.

1.3 Contributions

With the creation of this project we want to contribute to science. Purpose is to help familiar and non-familiar with this scientific area for better understanding of Deep Learning. Also we want to create two applications and share the source code of theme, so that somebody find a starting point, and then in the beginning contribute himself.

1.4 Thesis Outline

This thesis will outline the procedure of a java application construction using Deep Learning technologies, particularly CNNs. The following chapters will demonstrate the efficacy of such tools for research purposes.

1.4.1 Chapter 2-Literature Review

The primary goal of this chapter is to provide information on the knowledge required to accomplish this thesis. The whole work is divided into individual tasks and focuses on the requirements.

1.4.2 Chapter 3-Deep Learning Java Desktop Application

In this chapter, having finished the description of our Literature, we discuss about the procedure of design, using professional methods for our main task. All as mentioned, we will see

the route from the initial idea to design, and to determine completion time, when results meet the goals we set above.

1.4.3 Chapter 4 Web Application Framework, using CNNs

This Chapter discusses the webutility of this Master Thesis work, we reach the point to discussing about our web-application framework, after the implementation of the first application. As well as for the aforementioned chapter, we describe the procedure of design, through the calculation of time, it would take for the implementation of this task.

1.4.4 Chapter 5 Conclusions

In this final chapter, express our opinion over the work we did as a whole, such also the knowledge earned, through the root of manufacture. In addition to this, we comment on our work, and develop plans for future projects, as well as improvement of the project at hand.

2 Literature Review

In this chapter of Thesis, we discuss the literature and knowledge necessary need to reach our objective. In the beginning we will talk broadly about computational intelligence, and then we will analyze our individual literature issues. Succinctly at this part of our work, discussing about Deep Learning and specially for Convolution Neural Networks. Finally it is worth noting that this process was performed with Java and related libraries.

2.1 Computational Intelligence

Computational Intelligence, is described as the ability of an information system to learn patterns from a collection of given data and is often referred to as Machine Learning, although the term is not generally accepted. This title can be used for themes such as Fuzzy Logic, Neural Networks, Evolutionary Computation, Learning Theory and Probabilistic Methods. Collectively, they are recognized as algorithmic development used to resolve problems relating to the topics mentioned above [3].

The conversion of biological processes and learning cases that have copied functions from nature separately, is proof that nature teaches us once again its usability An important question that arises, is whether computers are able to achieve intelligence? Much research has been done on this subject since Alan Turing first posed this question, but there are still many physical processes that cannot be approached by computers. Although Artificial Intelligence and Computational Intelligence request a similar long-term goal, to reach physical intelligence, which is the intelligence of a machine that could perform any mental work that a human being can, there is of course a clear difference between them.

The Artificial Intelligence (A.I.) supported by hard computing techniques, while Computational supported on soft computing methods. implement methods from Computational Intelligent, and especially from the sophisticated and new form of Neural Networks, the Deep Neural Networks or otherwise Deep Learning with usage of Convolutional Neural Networks.

2.1.1 Types of Learning

In Machine Learning, there are three types of learning depending on the signal or the feedback that are available to the learning system. These are: Supervised Learning, Unsupervised Learning and Reinforcement Learning. We will briefly see the educational categories and we will choose the best form for our problem. These types of learning described as:

- **Supervised Learning**: Is a type of Machine Learning, wherein a function uses labeled training data as an input (in our problem, images), is transformed to a pair of consisting input object (in our problem, as a vector) and the desired output is a supervisory signal. During training, a supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for recognize other examples. The best case scenario is the algorithm specify an unseen example correctly, at the test operation[4].
- **Unsupervised Learning**: Use a function, to find a structure inside a set of unlabeled data. The data is unlabeled, so there is no error or reward signal to evaluate the possible solution. This is the main point that differentiates it from the other two categories [5].
- **Reinforcement Learning:** Is the area of Machine Learning, that has borrowed features from science of Behaviorist Psychology. Dealing with, how software agents must take actions, in such a way that rise the cumulative reward. This problem given the general nature that characterizes it, is studied and in many other fields [6].

As we proceed, it is evident that the unsupervised learning tactic is not the best solution to the problem posed due to the fact that as described above, our goal is to design something which can use labeled images from a user. It is possible that the use of this ploy would have been a better use for such problems as “Density Estimation” or “Statistics”. The Reinforcement Learning as we see, is a very broad field, and is surely not what is needed. Thus we chose to use the supervised.

2.2 Deep Learning

Deep learning, also known as Deep Structured Learning, Hierarchical Learning or Deep Machine Learning, is a chapter of Machine Learning supported by mass of algorithms, which are intended to model high-level abstractions in data by using multiple processing layers, often with complex structures or conversely, composed of multiple non-linear transformations. Deep Learning belongs to the Machine Learning. In that it makes use of methods pattern learning methods of data, such an example of input as image, can be mirrored in many different ways, such as a vector of force values per pixel, or in a more abstract way as a set of acmes, districts of a particular shape. Many representations are much better than others at simplifying the learning task. One of the Deep Learning guarantees, is that the features of different algorithms, will be replaced with unsupervised or semi supervised feature learning and hierarchical feature extraction [7].

The exploration of this topic, strides to make a better and better representations and construct models from large-scale unlabeled data. Some of the representations are inspired by advances in neuroscience and are loosely based on the interpretation of information processing and communication patterns in a nervous system, such as neural coding, which attempts to define the relationship between various stimuli and associated neuronal responses in the brain [8].

A lot of Deep Learning tools such as Deep Neural Networks, Convolutional Neural Networks, Deep Belief Networks and Recurrent Neural Networks have been used in fields like Computer Vision, speech recognition, natural language processing, audio recognition and bioinformatics where they have been shown to produce state-of-the-art results on various tasks. Deep learning has been characterized as a buzzword, or a rebranding of neural networks [9] [10]. In this master thesis, we use Convolutional Neural Networks [1] and apply them to image recognition and categorization.

2.2.1 Why Use CNNs

Convolutional Neural Networks it's a new technology, which has become a big trend, resulting in a great number of people engaged in developing it. There is also a plethora of tools for each different environment from which a programmer may choose, like as ***Deeplearning4j*** [11], ***Caffe*** [12] and ***Torch***[13]. Also it's better to understanding, because of the biological-base

design of it. Thus, enabling us to understand more and to intervene in it. CNNs also in contrast with similar technologies, have the advantage of capturing features from the input data, without the existence of a feature extraction algorithm, it taking semi-supervised the features from the buffering data and creating feature map automatically, as a result, we can save a lot of time from the feature extraction procedure. In fact, we design the architecture of our network, and we allow it to choose what it considers important from the given data.

2.2.2 Convolutional Neural Networks for Visual Recognition

Deep learning is a subfield of machine learning that works with the usage of learning levels of representations, corresponding to a hierarchy of features, factors or concepts, where higher-lever concepts are characterized from lower-lever ones. The same lower-lever concepts can help to define a lot of higher-lever concepts.

Deep Learning can acquire the knowledge of multiple levels of representation and abstraction, this helps to understand a plethora of data, transformed to vectors such as images. The theme of Deep Learning arises from the Artificial Neural Networks, Multilayer Perceptron which encloses more hidden layers is a Deep-Learning system [14]. The main story at this work, is classified as a problem of Computational Intelligent, that has to do with recognition of objects in an image. The tool used, is CNNs for visual recognition. The architecture and characteristics are further analyzed below.

2.2.2.1 Structure Characteristics

Altogether, we define as Deep Neural Networks (DNNs), every Multilayer Perceptron with multiple hidden layers. Convolutional Neural Networks (CNNs) [1] , is a type of DNN. CNN is a productive recognition algorithm, which is universally used in pattern recognition and image processing. It uses many features such as simple structure, less training parameters and adaptability. The ‘weights” shared network structure makes it more similar to biological neural networks. It reduces the complexity of the network model and the number of weights.

Normally a CNN is comprised of two layers, one for semi-supervised feature extraction, and another for pooling. Each neuron is connected to the local receptive fields of the previews layer to extract the local feature. When the local feature is ready, the positional relationship between them and the other features will also be determined.

The second layer, is that of the feature map. Every feature map is a plane, the weights of the neurons in the plane are equal. The structure of a feature map, uses a sigmoid as an activation function, which can be a different type at each level. This makes the feature map have "sift" invariance. In addition, since the neurons in the same mapping plane share weight, the number of free parameters of the network is reduced.

Each convolution layer in the convolution neural network is followed by a computing layer which is used to calculate the local average and the second extract, this unique two feature extraction structure reduces the resolution. CNNs are used to detect the displacement types of distorting invariance of two-dimensional graphics. Since the feature detection layer of CNN learns by training data, avoids explicit feature extraction and implicitly learns from the training data when we use CNN. Also, locally in the map plane the weights are the same, so the network can learn jointly. This is the significant advantage of a convolutional neural network over a simple neural network. CNNs, with the advantage of shared weights topically, it's an efficient tool adjustment in voice detection and image recognition applications.

The real layout of CNN is in fact very common to a biological Neural Network, the complexity of the network significantly decreases with the shared weights. In particular, multi-dimensional input vector image can directly enter the network, in such a way to reduce the complexity in feature extraction and classification process. In Reality, Deep Learning can accomplish the approximation of complex function by a deep nonlinear network structure [14].

2.2.2.2 Algorithm Operation

The CNN algorithm itself, is a multilayer perceptron structure, is a specific tide stand of two-dimensional image information's. The standard existing layers are always:

- Input Layer
- Convolutional Layer
- Sample Layer
- Output Layer

In addition to the CNN architecture we can have more than one of Convolutional and Sample layers. CNN is not necessity exist a Boltzmann Machine [15]before and after at the adjacent layers of the neurons for all connections in CNN algorithms. For each neuron it is not

necessary to see the entire image as it is possible to focus on a local area. In addition, each neuron parameter the same such as the sharing of weights and each neuron has the same convolution kernels to deconvolution image.

CNN algorithm as we said before make uses of two main actions, convolution and sampling.

- **Convolution:** Is the process which the output is a convolution layer ***Cx***, so it uses a trainable filter ***Fx***, the deconvolution of the input image which also referred to as Feature Map and a bias ***bx***. In this order is possible to supply desired results.
- **Pooling or Sampling:** Is the second process in which n pixels of each neighborhood through pooling steps, become a pixel, and then by scalar weighting ***Wx* + 1** weighted, adds bias ***bx* + 1**, and then by an activation function, produces a narrow n times feature map ***Sx* + 1**.

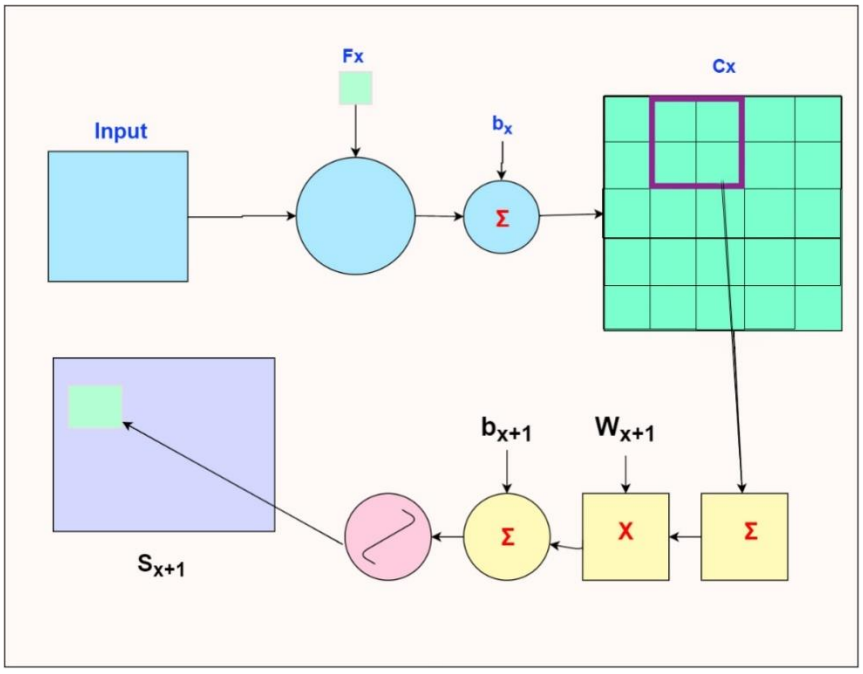

Figure 1: Process of CNN

The key of CNN architecture is the local receptive field, which sharing of weights and sub sampling by time or space, so as to extract features and reduce the size of the training parameters. The advantage of CNN algorithm is that its avoid the district feature extraction, so

learn from training data, the weights of the same neuron on the surface of the feature mapping, thus the network can learn alongside, reduce the compellability of the network. With the usage of the sub sampling structure oriented from time or space, cane accomplish some degree robustness, scale and deformation displacement. It need a good network topology and input information, if they set correctly we have more efficient results, in image processing and speech recognition. [14].

2.2.2.3 Architecture Design of CNNs

The "Implementation of Training Convolutional Neural Networks" [14] provides an example that helps to understand the operation of the CNN algorithm. The CNNs' operating algorithm, has both an experience in Architecture design and is also an unendingly requires much debugging in a workable application. For example; importing gray image of 96*96, during the preprocess is transformed into 32*32 of the size of the image. Designing a convolutional model architecture with 7-layer depth, which contains:

Input Layer

Assume as an input a 32*32 image after the preprocessing, at this point we have 17 different pictures.

Convolutional Layer C1

C1 layer adopts 6 kernels for convolution, which each of these kernels is 5*5 and can produce six different feature maps. Each feature map contains (32-5 + 1) * (32-5 + 1) = 28 * 28 = 784 neurons. At this phase there are6 * (5 * 5 + 1) = 156 parameters to be trained.

Sampling Layer S1

S1 Layer, also contains six feature maps, each one of which has 14*14=196 neurons, the sub sampling window is a matrix of 2*2, sub sampling step is, so the S1 Layer have 6 * 196 * (2 * 2 + 1) = 5880 connections every feature map in the S1 layer contains a weights and bias, so a total of 12 parameters can be trained.

Convolutional Layer C2

At this layer we have 16 feature graphs, where each of them contains (14-5 + 1) *(14-5 +1) =100 neurons, and espouse full connection, with each characteristic figure used to own six convolution kernels, with six characteristics from the S1, convolution and figure. Each feature graph contains a number of 16*(150+1) =150 weights and bias. So C2 contains a batch of 16 *(150+1)=150 parameters which could be trained.

Sampling Layer S2

In this layer we have 16 feature maps with 5*5 neurons, S2 totally contains 25*16=400 neurons. S2 on characteristic figure of sub sampling window is 2*2, so exist 32 trainable parameters.

Hidden Layer H

As a connection layer for all hidden layer H has 170 neurons, each one is connected to the 400 neurons on the S2. So H layers has 170*(400+1) =48120 feature map's parameters.

Output Layer F

The output Layer F for all the above connections, includes 17 neurons. A total number of 17*(170+1) =2907 parameters ready for training [14].

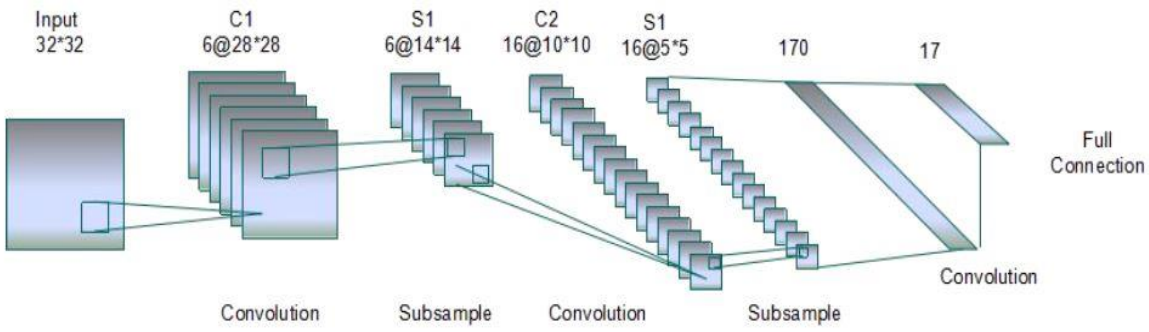

Figure 2: Training Steps of CNN Architecture[14]

Worth noting that, every data we want to use, works better with modified architecture specially for them.

2.3 State-Of-The-Art

Deep learning is a newly efficient technology applied in solving many problems containing mostly continuous attributes. Having a high ratio of supervised/unsupervised sets. Deep Learning is a State-of-the-Art in many areas of computer science. In addition, in Computer Vision problems, it provides effective solutions such as face recognition, image and digit classification. Some of the most popular technologies using Deep Learning describing below.

2.3.1 Deep Learning on Network Traffic Identification

Most network traffic identification systems are based on features, which use the port numbers, static signatures, statistic characteristics. The biggest problem in traffic identification is to define which is the feature for recognition. This requires lengthy work in the flow of data, which can be can be translated to a length of time. This solution, has no usage in unknown protocols. A solution to this problem coming with Deep Learning, as we read in *"The Applications of Deep Learning on Traffic Identification"*[16].

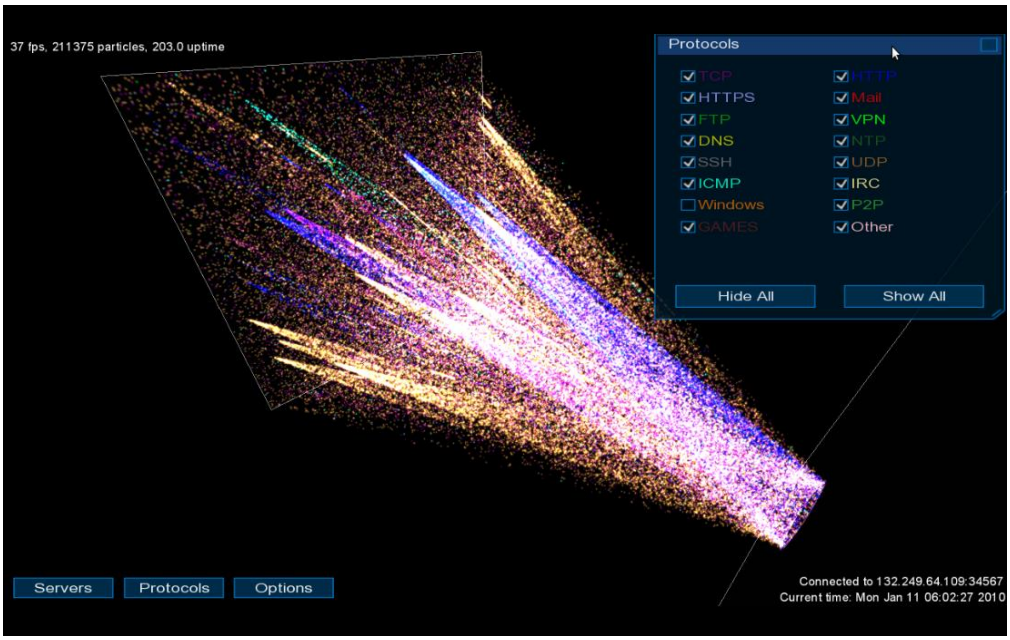

Figure 3: Network Traffic[17]

In the aforementioned work that we talk about, a Deep Neural Network is recommended for learning of Network Traffic characteristics. Also accomplished protocol classification and anomaly protocol detection.

2.3.2 Google Brain

Google Brain [18] Is a research project using Deep Learning which grows in google. It is the attempt to build a large-scale Deep Learning software. The aim of this project is to managed simulation with computer's clusters of the human's brain operation. After a report in the New York times, Google Brain achieved training itself to recognize cats, based on 10 million digital images taken from YouTube videos.

Figure 4: Cat Detection from Google Brain [19]

2.3.3 State-of-the-Art Visual Recognition

According to "DeepFace: Closing the Gap to Human-Level Performance in Face Verification" [20]. At this paper the scientific team achieve visual recognition in quota of 97.35%, which is a record for visual recognition. With a network with more than 120 million parameters using several locally connected layers, and in contrast with my method without

locally sharing of weights. Achieved to train a large facial dataset, an identity labeled dataset of 4 million facial images from more than 4.000 identities.

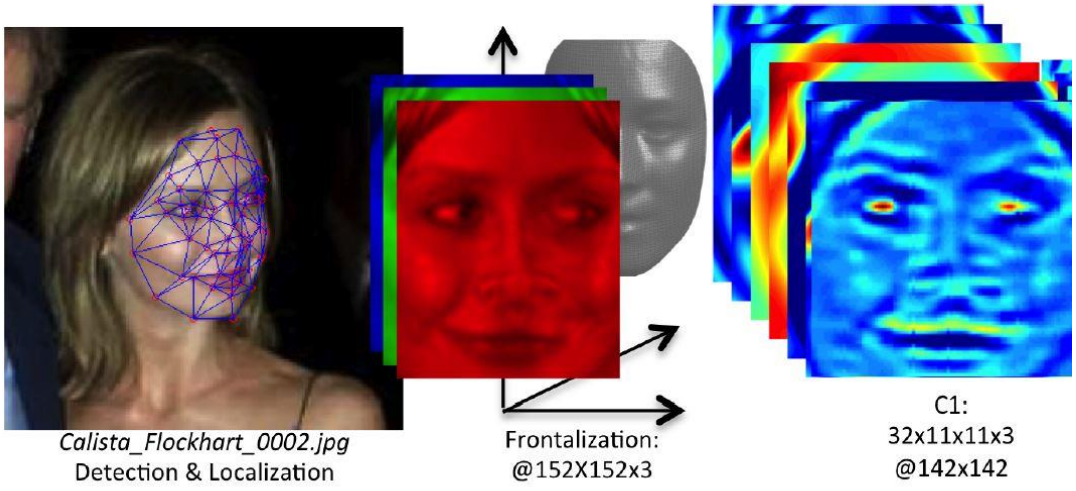

Figure 5: 'Deepface' photo-matching, nearly good to human's brains [21]

2.3.4 Apply Deep Learning in Financial issues

Deep Learning is a tool which can be used in from several sectors of computer science. As computers grow more complex, can handle bigger problems in this hi-tech era. It is the norm for computers and information systems to be used for the management of financial operations. The writing team of the paper "Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks" [22], use an auto encoder composed of stacked Boltzmann Machines [15], to extract features from a stock prices data base, that enables it to discover an enhanced version of the momentum effect in stocks.

2.3.5 ImageNet Classification with Deep Convolutional Neural Networks

In this paper work (ImageNet Classification with Deep Convolutional Neural Networks [23]), a deep Convolutional Neural Network was created, to classify the 1.2 million images of high resolution in ImageNet LSVRC-2010. There were 1000 different classes. Test data they

top-1 and top-5 error rates of 37.5% and 17%, was improved better when compared to the last state-of-the-art.

Figure 6: ImageNet Data-Set view [24]

2.3.6 Autonomous

Autonomous is a robot using Deep-Learning gadgets, which aims to be a tool for researchers. A researcher could buy it, with the pre-installed features such as Google Tensor Flow [25], Robot Operating System (ROS) [26], Torch [13], Theano [27], Caffe [12] and Cuda + cuDNN powered from NVIDIA [28],such also other mechanical features could carry out many experiments of Deep Learning in the physical world. Which would help the researcher to understand with more efficiently, the utilitarian value of this scientific field, as a result in the root of time people take goods and services which improve their lives.

Figure 7:Autonomous, robot with Deep Learning Gadgets

2.3.7 NVidia Digits

Digits is a Deep Learning GPU Training System (D.I.G.I.T.S.).It's a Deep-Learning tool with which somebody can conduct experiments of image classification and object detection tasks.

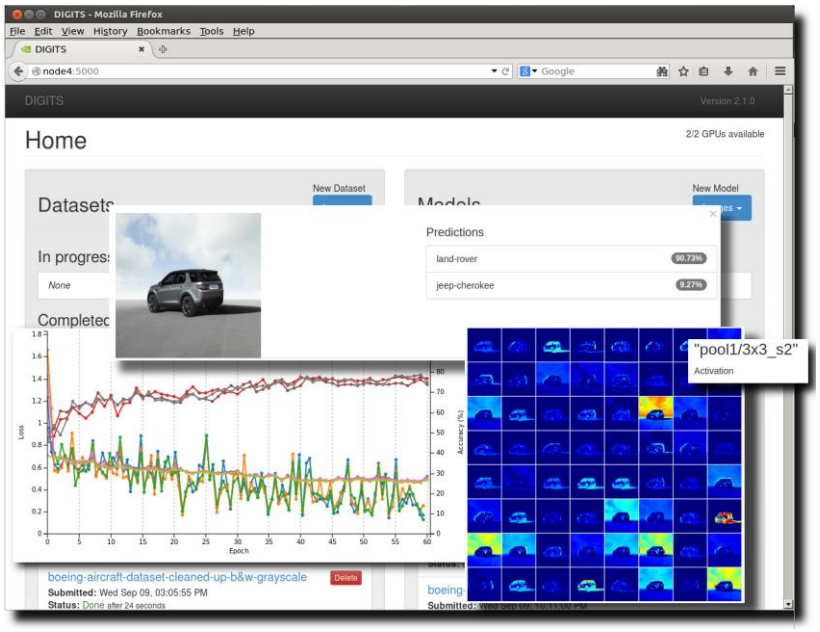

Figure 8: NVidia Digits

Digits is similar to CNNs Tester, but a more complex one, which allows to the user to decide the Deep-Learning architecture and observe the results in real time. The user may carry out many types of experiments in addition the monitoring will aid in this. Also as it is referred toin their site “as it is referred completely interactive, so that you can focus on designing and training networks rather than programming and debugging” [29].

2.3.8 TensorFlow

TensorFlow is an open-source software library powered by Google. Using this software can easily depict an architecture through graphs and perform a plethora of experiments. With a flexible architecture you can deploy to one or more CPU or GPU in a desktop, server or even a mobile device with a simple API.

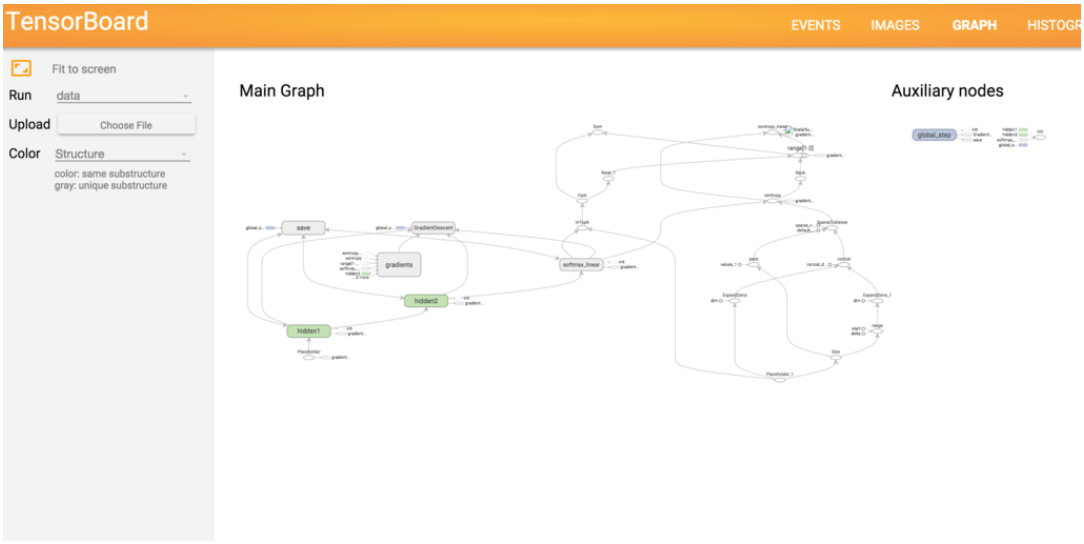

Figure 9: TensorFlow

3 Java Desktop Application using CNNs

In this chapter we describe the root from basic idea to final design of our Java Desktop Application, which we named CNNs Tester.

3.1 Basic Idea

The basic idea of this master thesis work, is to create and design an application, which can be used as a tool of Deep Learning. Due to previous involvement with the field of Computer Vision in older projects, we decided making an application using CNNs, which could manage image experiments. Therefore, we proceeded to design an application, with which one could easily train a model with labelled images (Supervised Learning [2]). In the second act of this project a researcher can test an image over the trained model. Also the application discussed, could create model for further use as we show bellow to the next chapter. Worth noting, that an application GUI like this, with Deeplearnig4j can be compared with only two known applications, TensorFlow [25] and Digits [29].

3.1.1 Purpose of Implementation

A significant query which must answered, is what is the goal we want to accomplished with this work? With the completion of this application we want to make a user-friendly and easy to understand application, for Computer Vision experiments by using CNNs. Also, with the upload of the source code in GitHub, we would like the researchers can find a background, to modify with their architecture, and manage their own experiments.

3.2 Methodology and Tools

Final implementation of our main work selection of environment and framework for the accomplishment, and also the main architecture from the CNNs palette.

3.2.1 Environment

Initially, after conceiving the idea, it is crucial to decide upon the technology that will be used for reaching our goals. Deep Learning is new and currently considered a trend. Which explains and makes it reasonable that many frameworks would exist, in different environments. The original idea was to implement in C++ and Caffe [12] bookmark, but after researching and experiments realization, decided that Caffe may be a very useful tool which can use Cuda-Cores too (the biggest advantage), but was not suitable for our problem.

Due to the nature of the laboratory where this project done "ISTlab T.E.I. Crete", selected to proceed with Java. So, we had to find a compatible framework to proceed. Immediately the decision was made. The solution selected was to work with DeepLearning4j [11]. Also a New bookmark developed in San Francisco, which is an open source distributed Deep Learning project, for java and Scala. provides of 24hr support online via open-chat, which aided greatly.

3.2.2 Architecture

The design of the architecture for this project was a major issue for this work. We ended up to implement with supervised learning which uses labelled data, so became a search for similar work to help us, a compatible architecture selected for this work, was this from Hiren's Dutta [30] "GitHub" profile, in the project "AppLocalScnes". In light of the above we proceeded to design a CNN with five layers, and initially started experiments with images of 112*112 pixels, in order to make all needed modifications. Each layer of CNNs performing three or four major operations, the first is input and the last is output, while the two intermediate are the convolution and the Pooling.

- **Input:** taking a signal as an input, and promoting it to Convolution.
- **Convolution:** computes a value from each input pixel.
- **Pooling:** applies a kernel filter and in depend of the sigmoid function taking samples from the signal.
- **Output :** promotes the signal to the next layer.

As mentioned above and described in chapter 2, the two major operations are Convolution and Pooling.

.

3.3 Design

By design we mean, all the things we do after capturing the idea, to aid us in organizing our work. At this chapter we will see all the diagrams we made to help us finalize the implementation's boundaries. We also speak for all these independencies which needed to implement before the first edition of our program.

3.3.1 Diagrams and Tables

Atempt to show the content diagram of our application, which include an overall brake-down of our project. In addition, presentation of Class and Use Case diagram, in which are drafted all the uses, which the user or the application itself can perform. Following the tables with the work packages and the independencies.

3.3.1.1 Content Diagram

A Content Diagram in which boxes represent the components that needed to complete a project. In this piece of work, namely CNNs Tester Java Application, the basic component described from boxes is all the knowledge of Deep Learning theory, Deeplearnin4j which is the bookmark that we used, the Java environment and the images given from the user.

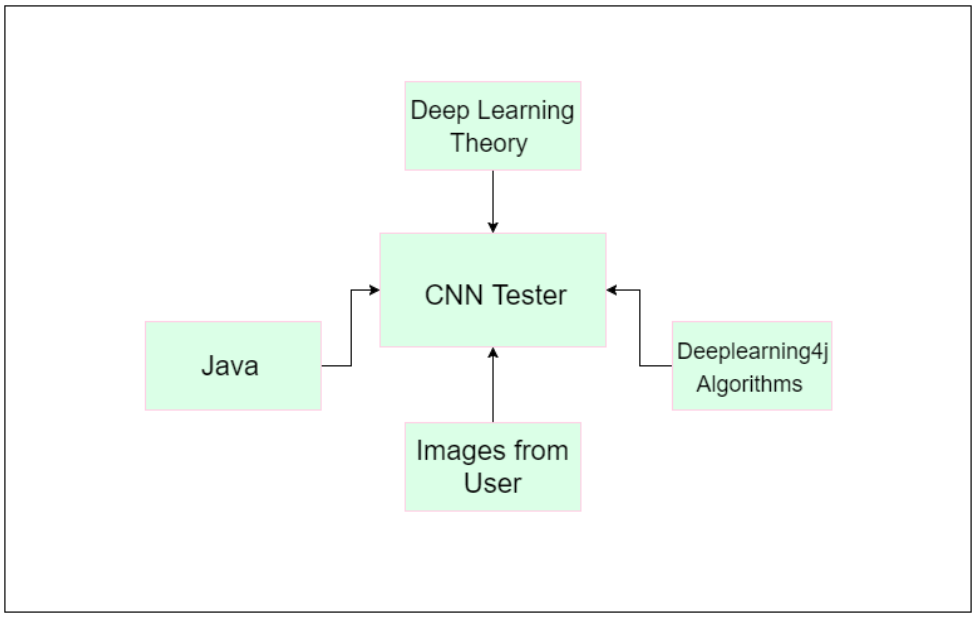

Figure 10: CNNs Tester's Content diagram

3.3.1.2 Use Case Diagram

Use Case Diagram is a simple representation of the actor interaction in our system, also showing the relationship between the actor and all the use cases in which actor is involved. In the Use Case diagram, the different type of actors are distinguished.

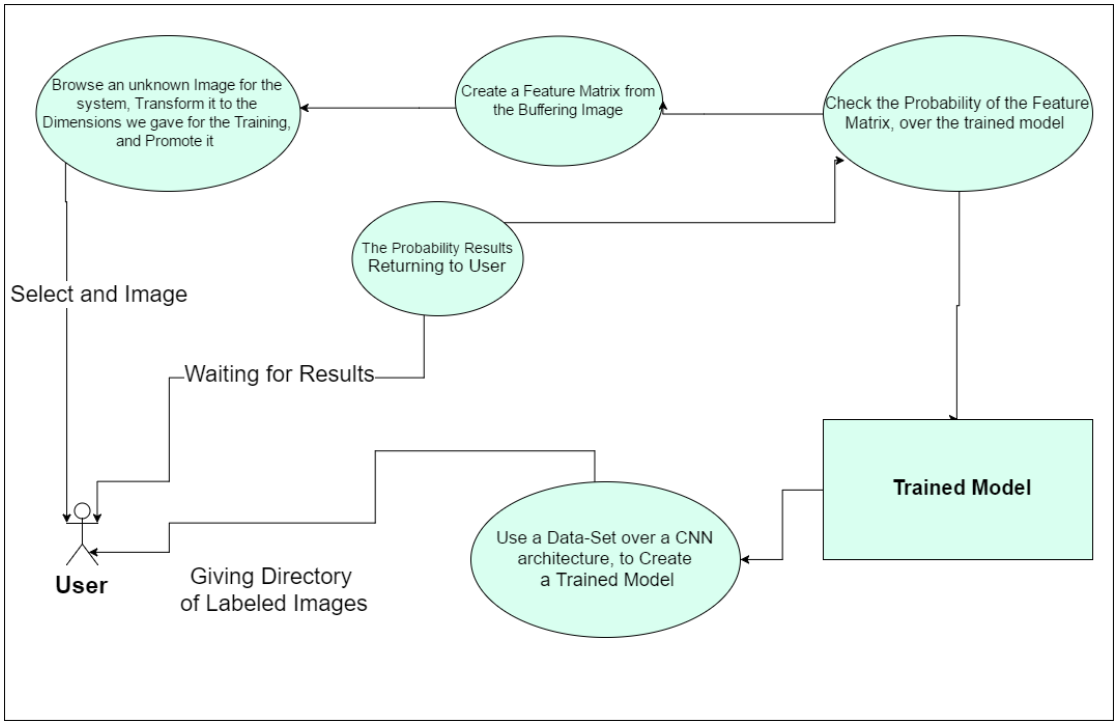

Figure 11: CNNs Tester's Use Case diagram

3.3.1.3 Class Diagram

In the class Diagram can be seen all the classes of our project and the connections between them. Also separately in each class we see the attributes and the methods. In this application there are three basic classes. The first one is the "main" which includes the form design and all the user choices arising from the selection of the buttons, also is connecting the other two classes, for the total operation of our program. The "Training" class is that includes the algorithm for the training of a CNN and all to components to resize a group of images and create a trained model. Finally, the "TestOnImage" class is the one that includes the reload a trained network, and with the use of a new image to create a test.

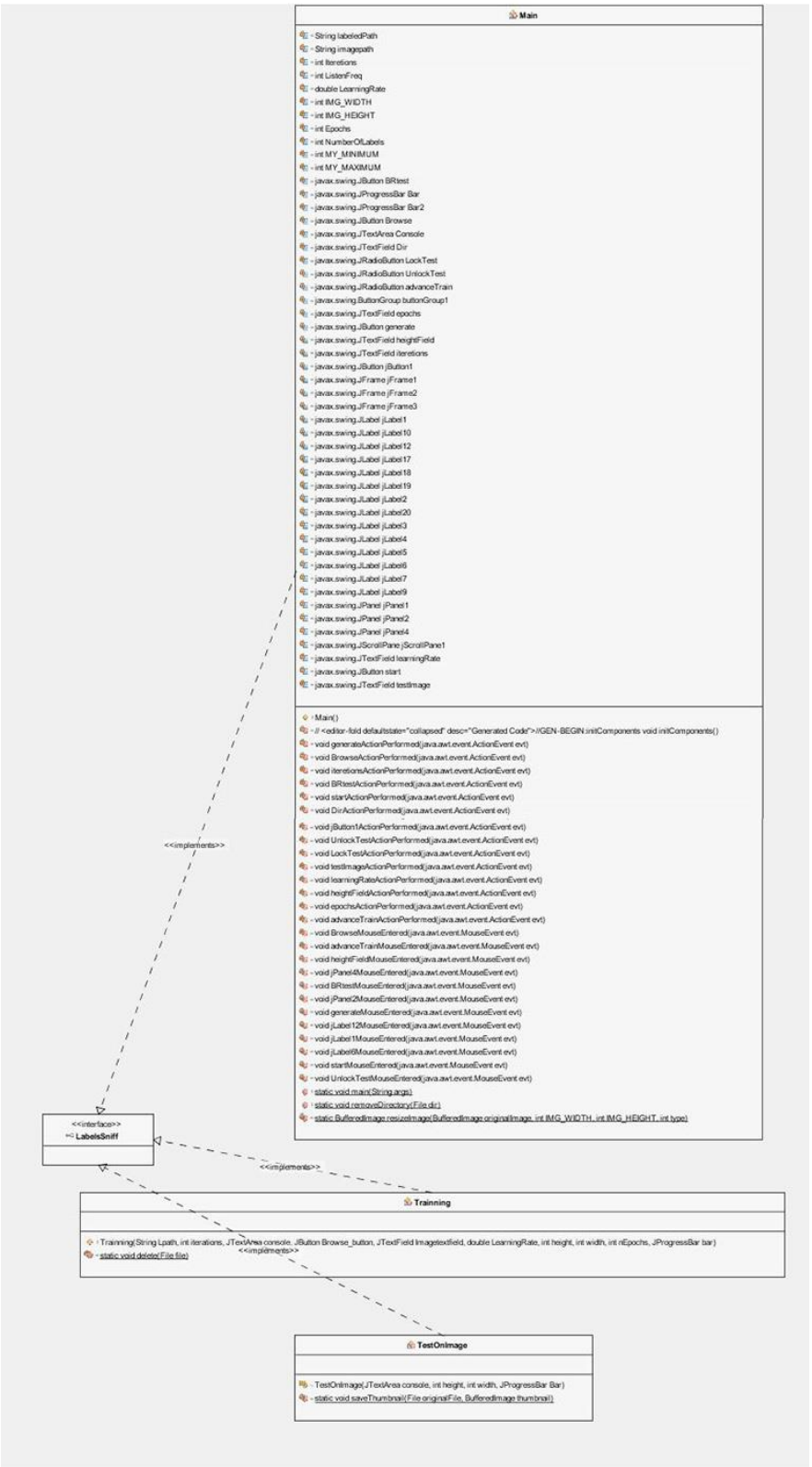

Figure 12: CNNs Tester's Class diagram

3.3.1.4 Table of Work Packages

Activities in the work-packages composing our work. Also the completion time in weeks, taking into account that it was implemented by one person.

Work Packages	**Activity Number**	**Activity Title**	**Completion Time (weeks)**
First Steps After Idea Catch	1	Environment Selection	3
	2	Frameworks and Tools	1
	3	Deeplearning4j Learn	6
Experiments	4	First Experiments	2
Program Finalization	5	Components Algorithms writing	2
	6	"CNNs Tester's" Form Creation	1
	7	Import Code from experiments to Form Functions	2
	8	Corrections	2
	9	Improvements	1

Table 1: Table of Work Packages for CNNs Tester

3.3.1.5 Table of Independences

In this table, by using the above knowledge we present the independencies, as regard the implementing order.

Activity's Number		**Activities Independencies**
1	Must be finished before	2 can start
2	Must be finished before	1.6 cam start
3,1,2	Must be finished before	4 can start
4	Must be finished before	5,6,7 can start
5,6,7	Must be finished before	8,9 can start

Table 2: Table of Independencies for CNNs Tester

3.3.2 Phases

In this Chapter we discuss in few words the phases of our work until final implementation.

3.3.2.1 Phase 1-First Steps

During the first phase after the initial capture of our idea we needed to find a way of implementation. So, we began to look at various technologies and tools which could be used for the implementation of an application like this. Before we started, we thought that an implementation of this project with C++ and "Caffe" would be beneficial, so started reading for Caffe [12] framework online. After a little time, and with communication with other developers , the decision had come, the best way to implement it was through java. This is due to the fact that java is an open source language with many components. In addition, help on programming issues with java, was available in the ISTlab.

Finally, having chosen the environment, what was left was to choose the proper Deep Learning framework. During the search performed decided the use of Deeplearning4j [11] framework, which despite being new had many examples and experiments through which a new users could learn. Finally, the fact that this framework has enough approach in the usage of CNNs, which are the main components of my work, was critical for the selection.

3.3.2.2 Phase 2 – Experimental Work

First experiments started with Deeplearning4j. Initially we used existing projects from GitHub [31], and later resolved our single problems. Over time and work-hours, finally established a code for training Convolutional Neural Networks with images from a local directory and another one which checks the probability of an image over the model that created to the previews problem. All that remained, was to transform through a class all the images in to standard square dimensions. To reach the implementation, it should have combined them all together in a java form as is being described in the next Phase

3.3.2.3 Phase 3-Program Finalization

The final phase of this project is implementing the third work package from the work-package's table. After completing the experimental work, and knowing the work further needed for program finalization, we began designing through a main class all the component classes needed.

Initially, the program's form started to take shape. Next, started using code from the experiments, with the purpose of making the form a working Java application. After a lot of

corrections and anticipated problems that came up, we finished the first edition of the CNNs Tester.

Later the first edition was followed by improvements suggested from supervisors, after the implementation of those we had a concrete edition. There are definitely improvements that can be further made. These are described in the Summary's chapter future work.

3.4 CNNs Tester – How it works

In this section we will discuss, the functions of the program in the different use cases made. Running the .jar file we will see the present frame:

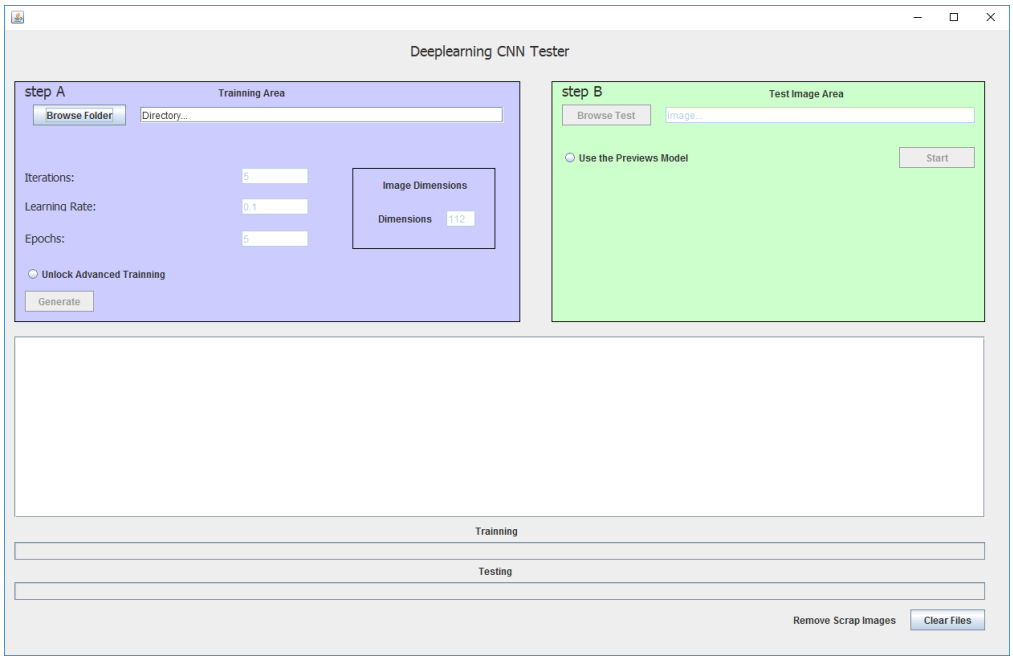

Figure 13: CNNs Tester's starting frame

Now, we must browse a folder with labelled images using the *Browse Folder* button. The Image's folder must be organized like this:

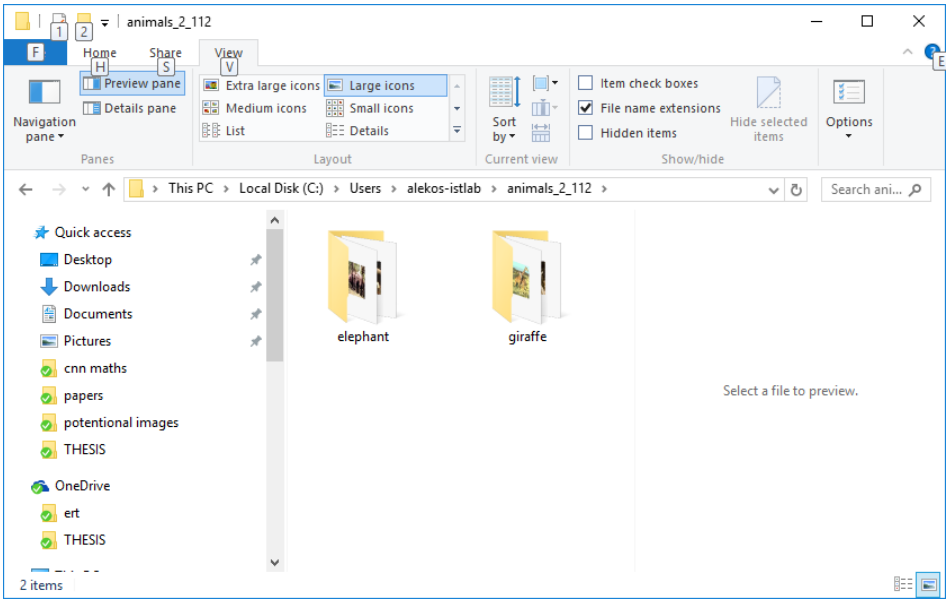

Figure 14: Shape that must have a Selected folder

After choosing the desired we have two choices, we can make a simple training without any further customization or the advance training, in which we have the ability for more customization. It depends from the user what will choose to do. By pushing the generate button we initiate the training and see the above:

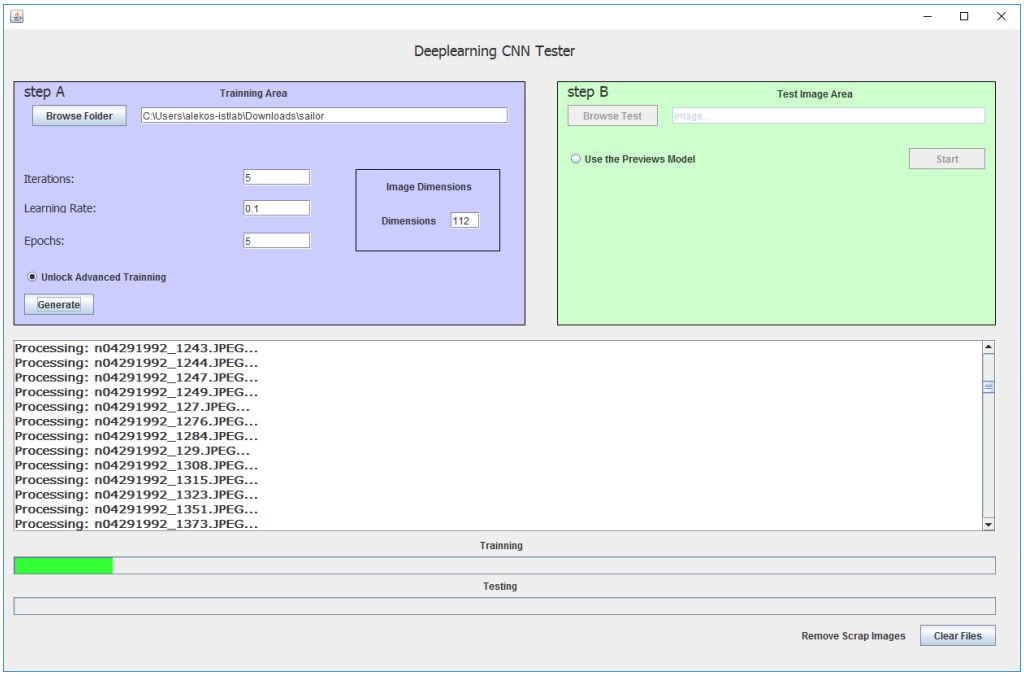

Figure 15: CNNs Tester's Training mode

When the Training finish, we see the training bar at 100%, and in the console window some statistics Accuracy, Precision, Recall and F1 Score.

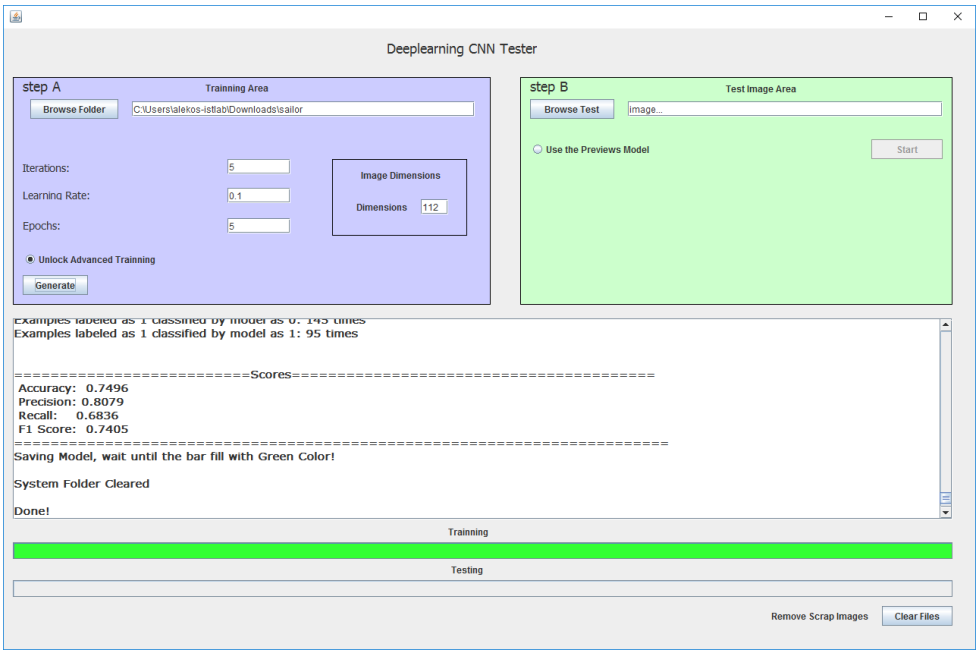

Figure 16: CNNs Tester Training end

Accuracy is model's accuracy in the training, **Precision** answers: “Given a positive prediction from the classifier, how likely is it to be correct? **Recall** explains how possible is it for the classifier to detect a positive example, and **F1 score** is the most significant number. The F1Score is a number between zero and one, and explains the quality of the training, the closer we get to one, the better the training. In actual is an average of both precision and recall [32]. It is also worth noting that we can experiment with the values in advanced training for better results.

After the training has ended, we may want to stop the program and only write down the statistics, otherwise we can perform a test on the trained model with a new image which our system never seen before. So we use the button *Browse Test*, and search through our folders for a single image to test. After the image selection we unlock the Start button and by clicking it we initiate the testing. Then we are wait until bar reaches 100%. If desired, we can perform another test after completion of the first one.

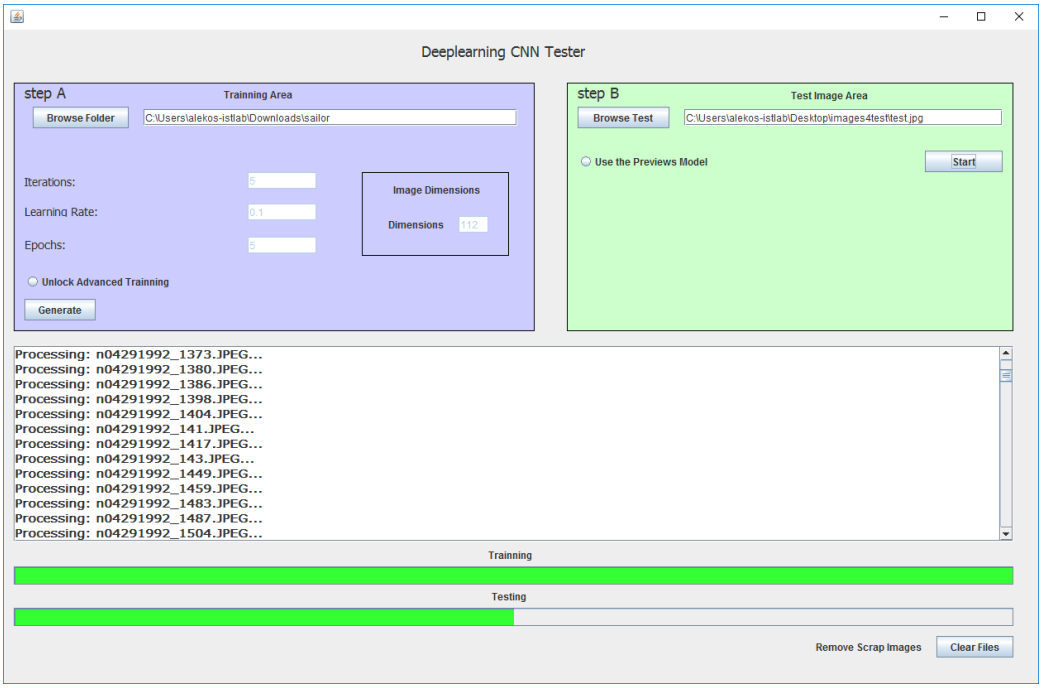

Figure 17: CNNs Tester, test have been start

The frame we see when the test ends would be like:

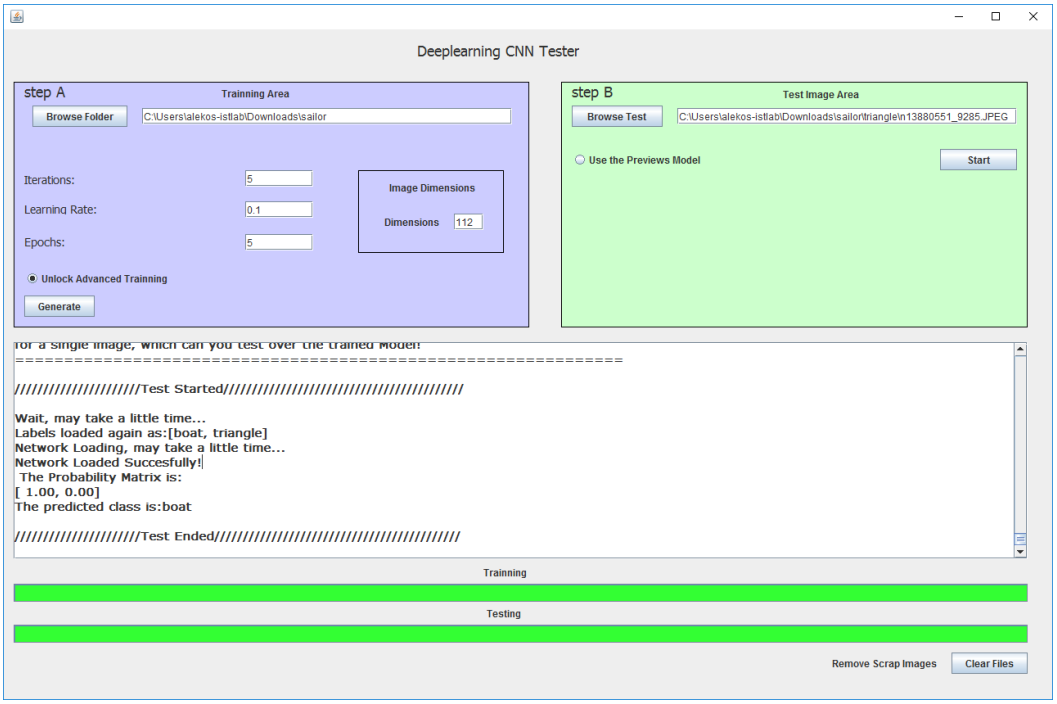

Figure 18: CNNs Tester, after the test finished

We can see in the console that the labels loaded from a. json file that we created in the training step are correct, this means that the trained model loaded with all it settings. Also we can see the results of the test and the prediction that the system did for our selected image.

Also, if we have a trained model in our application directory, we can make a test in another time. We open our application, and select the radio button in the step B, *Use the Previews Model.* Now we can with the same steps to start testing. The test in this case will be as is presented below:

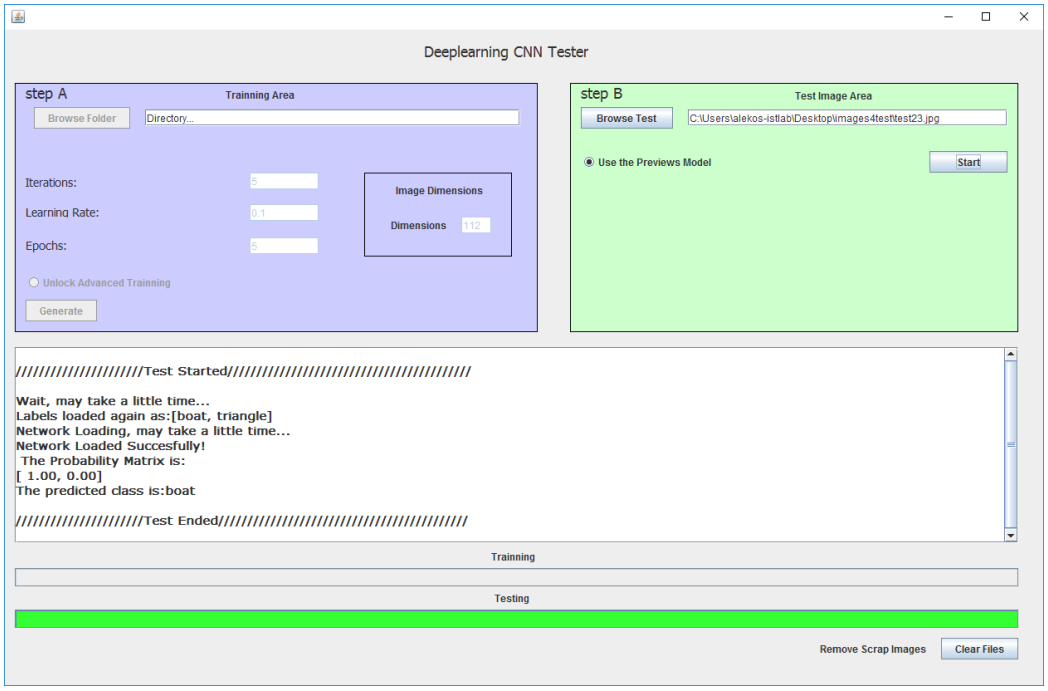

Figure 19: CNNs Tester with usage of a previous model

After the first run of our program a folder is created on our disk. We can find it at C:\DDL_APP directory, inside the folder we can find some documents which are details.yaml, info.json, model.bin and settings.json. The details.yaml have the characteristics of our architecture, in the info.json we save some variables which want to parse to the training step, the model.bin is the model with all the knowledge from the training (weights), and final the settings.json is actually the same document with the details.yaml. This project using the. Json file.

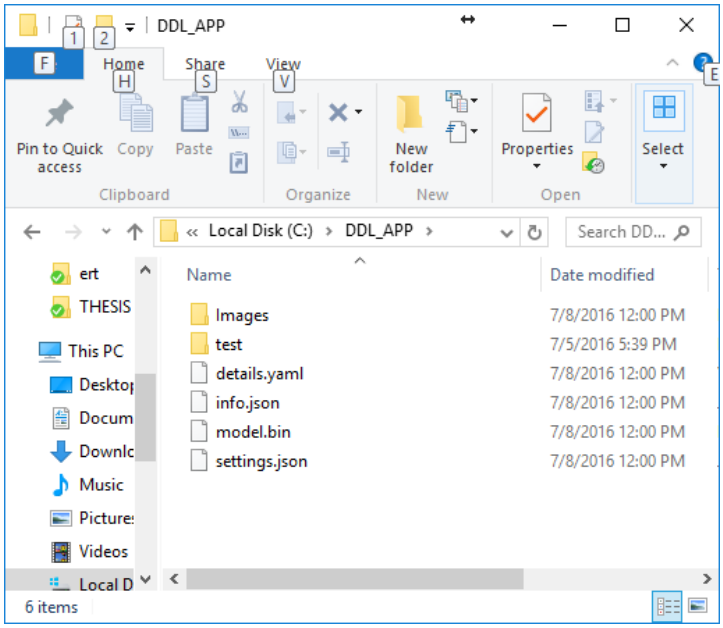

Figure 20: The folder that CNNs Tester creates

Plus, in the same folder we can see a sub-folder *Images*, in which our system converts all the selected images to the desirable dimensions. Also the sub-folder test in which the selected image for the test converted and stored. After running the program all the images are deleted automatically. Except if, we close the program forcefully while it is running, we have the next time to use the button *clear images*. This button deletes all the converted images, such also the image in the test folder.

3.5 Summary

In this topic, we will see some statistics from the use of well-known image-databases to our architecture, and then we will discuss what we gained from this work.

3.5.1 Architecture's Tests

In the tables below we see the results of Accuracy, Precision, Recall and F1 score for some well-known image-databases as we aforementioned. In the beginning we used one of James Z. Wang research data group taken from an online source [33]. Common work is described at the journal "SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture LIbraries" [34].

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

Table 3: SIMPLIcity Image Database results

Figure 21: Images from SIMPLIcity

As we can see above, it is not necessary that by using more epochs or iterations we will receive better results. Every dataset has different habits, and needs different customization in the settings. The question is, which customization will be the best? This one is going to be found only by experimenting on the data. Therefore, we understand that the experimental work as described in work-packages, is a very significant work. Below is a matrix of results for a well-known Image-Database of computer vision NEC Animal dataset [35] results.

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

Table 4: NEC Animal dataset results

Figure 22: Images from NEC Animal

To conclude, for comparison we used our own image-database from an older project. The image-database “Rabbit” contains 18 different classes of images and the three-dimensional of each image. For this experiment we needed two-dimensional images. In future projects it would be possible to conduct an experiment with 3D images as well. The matrix with the results as follows:

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

Table 5: Rabbit Dataset results

Figure 23: Rabbit Dataset view

Regarding to the above experiments we can demonstrate what we initially stated that every image-database has its own habits, and with different customization of our program we achieved the optimal result for each. Also we noticed the existence of one “more beneficial” point for each occasion, and after this point the results are not optimal. Finally, it is good to note again that in this thesis work we do not promote that architecture but the framework itself, however, given the opportunity one could download from GitHub the application’s source code and use their architecture.

4 Web Application Framework, using CNNs

4.1 Basic Idea

One of the purposes of this master thesis, based on Deep Learning, is to create a web-application which can recognize an item. The basic idea is that, we have a pre-trained model from the application above and we can use it with web technologies, to perform image identification tests. This is presented in the following chapter.

4.1.1 Purpose

The exact purpose of this work is the creation of a web-application framework, which one in common usage with the above application, it would be able to create his own App. Therefore, the user makes and optimize a training at "CNNs Tester", and subsequently uses the trained model (model.bin) document to reload the knowledge. This chapter describes the method with which was done.

4.2 Methodology and Tools

In the following section, we will discuss the tools and the list of sub-problems which must be solved in order to arrive at a stable edition of a framework like that.

4.2.1 Environment

The environment in which the project was fulfilled is Java. A java framework for Deep Learning was essential, so we used again Deeplearning4j. Helped also the fact that is an object oriented language and has direct relationship with server design application and an internet-oriented logic.

4.2.2 Architecture

This architecture of the web based version is simpler than the architecture of the desktop presented in chapter 3.

At the web based architecture we made use of a pre-trained model from the “CNNs Tester”. Loaded again in in addition with the settings of our architecture from a. json file. This is achieved by use of Apache Tomcat server [36] and with a servlet specially designed to upload images. It is also worth noting, that the image classification done with the same classes used from the CNNs Tester for testing.

4.3 Design

Below, all the design diagrams and tables are presented.

4.3.1 Diagrams and Tables

4.3.1.1 Content Diagram

Content Diagram is a simple representation of our work, which shows the basic components.

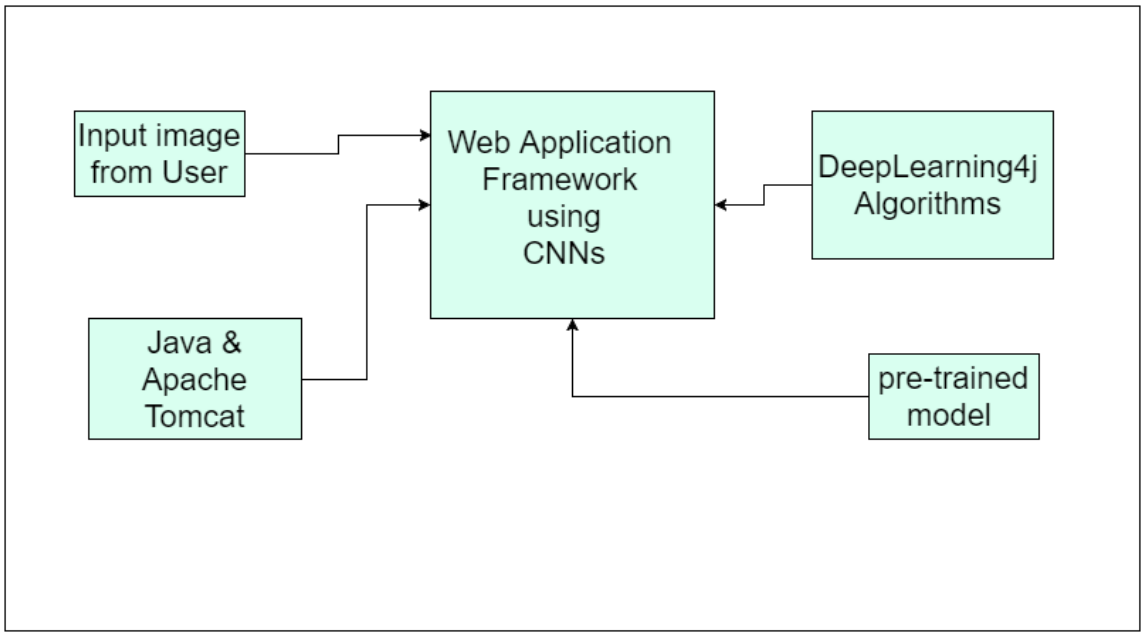

Figure 24: Web-application’s framework Content diagram

4.3.1.2 Use Case Diagram

A Use Case diagram, which represents the functions of our web-application's framework. The same apply, for every application construct with this framework's structure. Worth noting, that is logic to have many similarities with the "CNNs Tester's" Use Case diagram.

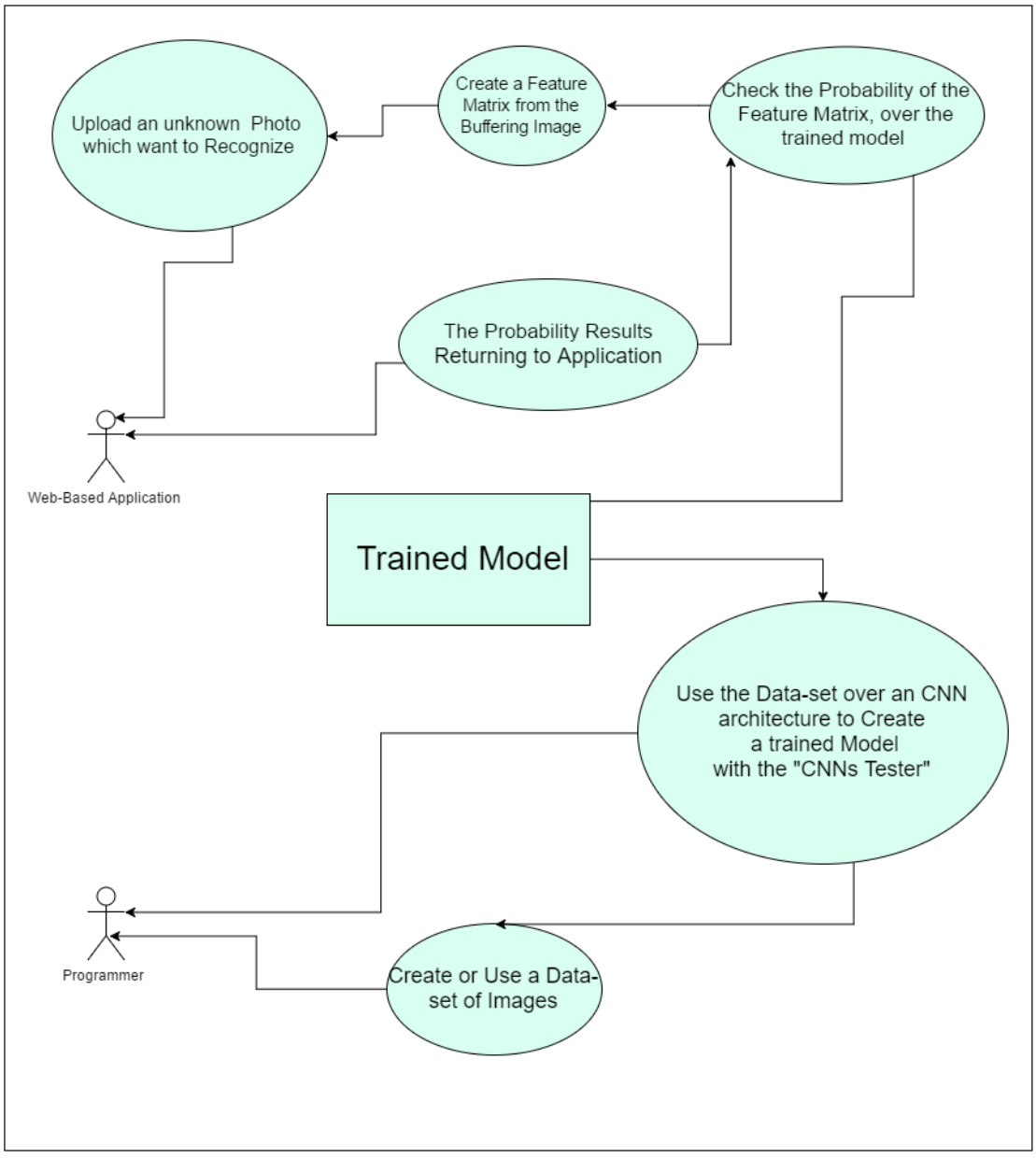

Figure 25: Web-application's framework Use Case diagram

4.3.1.3 Class Diagram

This Class diagram shows the relationships between the classes of our Web Application. We can see the servlet which upload an image after the call from the index page. Also, the Image_Classification class which performs the main job in our application. Furthermore, there is, a class to resize an image in to desired by the system dimensions.

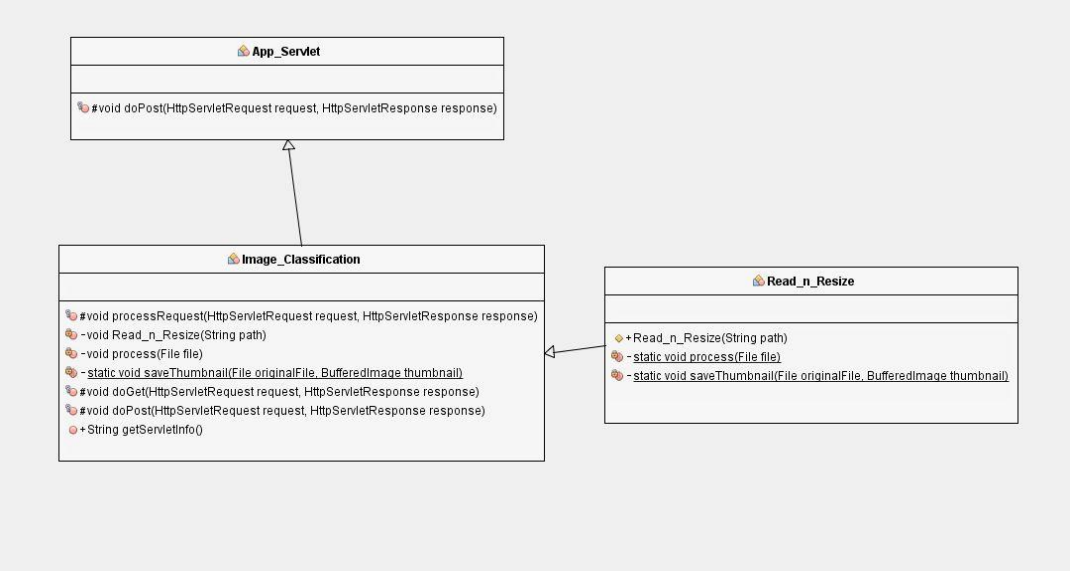

Figure 26: Web-application's framework Class diagram

4.3.1.4 Table of Work Packages

In this table we can see the needed work for our web-application framework which is being separated in work packages, and integration time in weeks.

Work Packages	**Activity Number**	**Activity Title**	**Completion Time (weeks)**
First Steps After Idea Catch (same with CNNs Tester)	1	Environment Selection	3
	2	Frameworks and Tools	1
	3	Deeplearning4j Learn	6
Experiments (same with CNNs Tester)	4	First Experiments	2
	5	Components Algorithms writing	1
	6	Web-A	1

Web-Application Framework Finalization		Application Index design	
	7	Import Code from experiments to work with index	2
	8	Corrections	1
	9	Improvements	1

Table 6: Web-application's Framework Work-Packages' table

4.3.1.5 Tables of Independencies

Here we can see the activities' independencies in common with the table above.

Activity's Number		Activities Independencies
1	Must be finished before	2 can start
2	Must be finished before	1.6 cam start
3,1,2	Must be finished before	4 can start
4	Must be finished before	5,6,7 can start
5,6,7	Must be finished before	8,9 can start

Table 7: Web-application's Framework Independencies' table

4.3.2 Phases

Reference to the work happening individually for each work-package.

4.3.2.1 Phase 1

This, is the same as we described to the chapter three. This is due to the fact, that we selected the same background to implement both of the two applications.

4.3.2.2 Phase 2

The same applies to phase 2, A great number of experiments performed, with purpose the better understanding of the CNNs.

4.3.2.3 Phase 3

At this project's part, we demonstrate the usefulness of Deep Learning., through a web-application. This application use a model from "CNNs Tester", and with some Deeplearnig4j tools, makes a precision to an unknown image, always relative with those categories used for training. As a result of this process, was the mentioned web-application framework. In which one can copy the structure, and create an application too.

We use an Apache Tomcat server, HTML5 and a servlet to upload the images. The process of classification taking place with Deeplearning4j functions once again. Our main page is in HTML5, and with usage of Ajax technology we can use the same java code, as "CNNs Tester". Next, we demonstrate how the existing example (PaternF) of this framework works.

4.4 PaternF Application

To use an application like this, we visit a web-page, in which we can see the main Index frame. In this example we have a very simple problem to solve, we want to classify an image and see if it is an elephant or a giraffe. So the first think to do, is to upload an image selected from our personal computer.

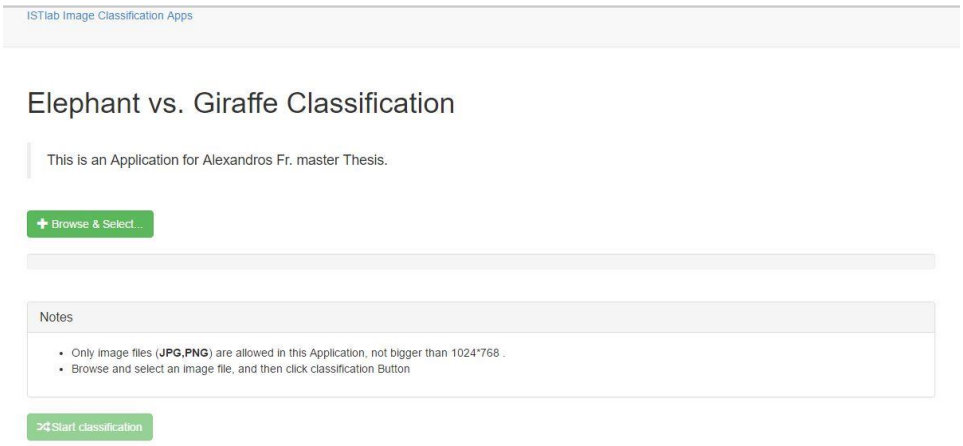

Figure 27: PaternF starting page

In the next frame, we see the image uploading bar. When the upload is completed the "*Start Classification*" button unlocks.

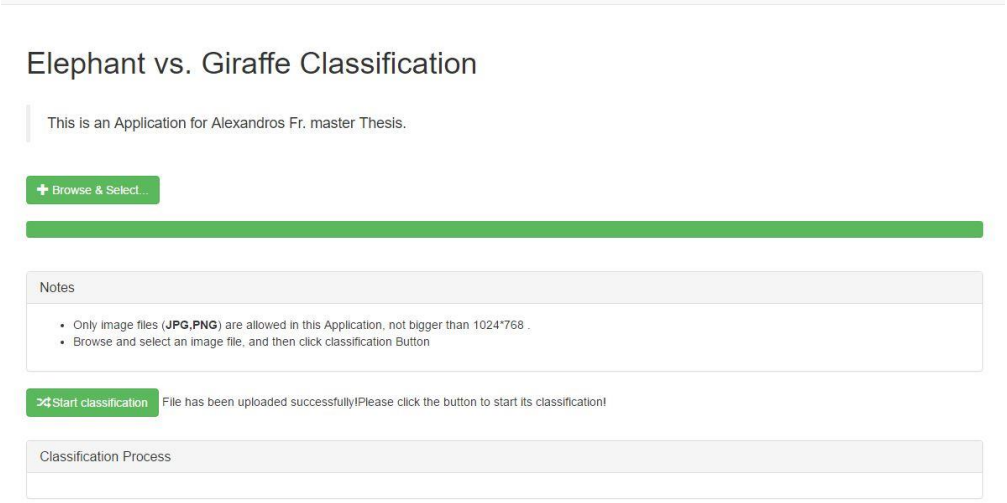

Figure 28: image upload completed

After we use the button of classification, the class "TestOnImage" starts from the Ajax and we wait for the results.

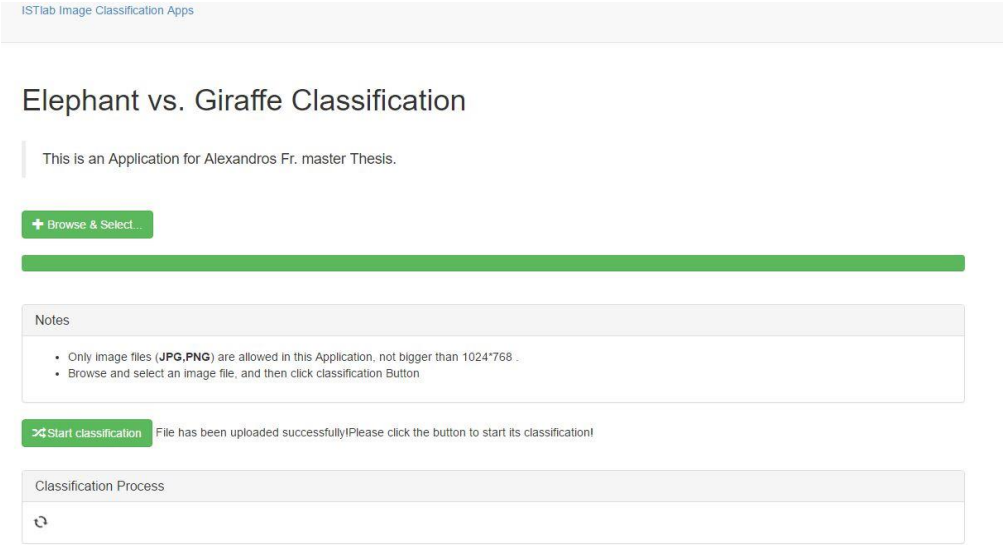

Figure 29: PaternF waiting for result

During the wait, the two little arrows move, is understandable that the classification finished, when the arrows stops, and see the message in the classification process area.

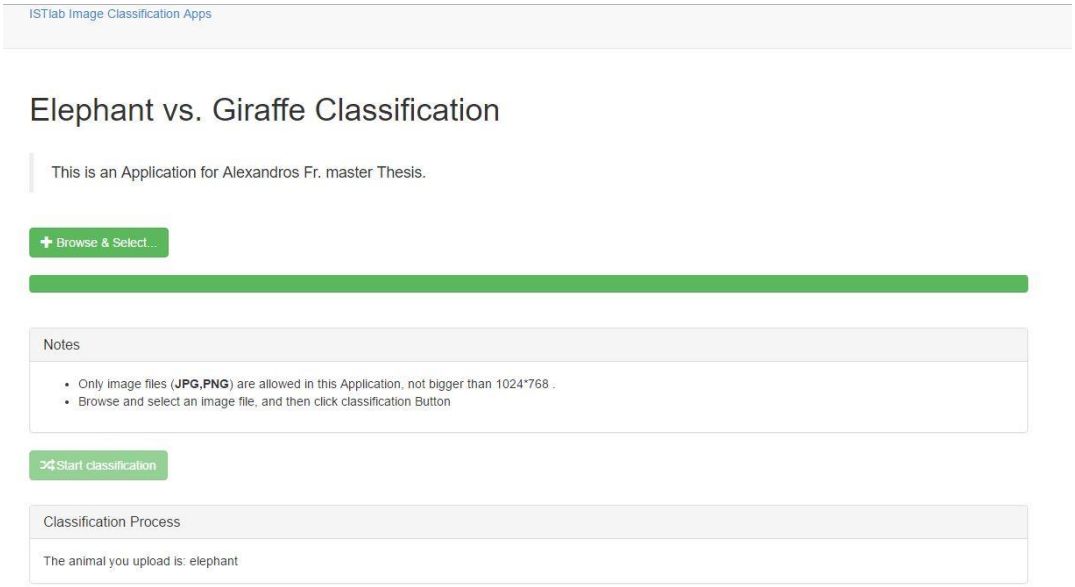

Figure 30: PaternF results

4.5 Summary

During this work, our existed knowledge for the web applications improved. Also we achieved to demonstrate the usefulness of this technology. Therefore, we achieved to create an existing application, in which we can support the implementation of a more complex.

.

5 Conclusion

In conclusion, in this master thesis work, we created two applications which can be translated as framework for a total work. Firstly, we have the "CNNs Tester" in where a model is trained with the knowledge from the images used. Next, we saw that the customization of the training give better results and after some experiments a trusted model can exists. The second part of the designed framework, which uses a model from the first, can recognizes objects related to the classes we used for training. It is therefore a complete project. There are a number of things that could be added which are described in Future work.

5.1 Improvement of existing work

The two applications are very close to the requirements set. Many extensions can be found. Some of these are:

- **Visualization Panel:** Surely the next step is to add a visualization panel, so a researcher can watch the training.
- **User Friendliness Improvement:** We ended up with a useful GUI, but there is much that could be added, such us a preview of the image in testing Step B.
- **Architecture Customization:** It is very important to achieve customize architecture, in this way the usefulness such as the use cases will multiply.

5.2 A step further

The next step in the research work is the use Deep-Learning for other scientifically areas than Computer-Vison. The intension of project's implementation is to use Deep-Learning for financial misdemeanors. Also, the hope of art improvement is a dream of this research for future scientifically projects.

5.3 Knowledge Gained

After the completion of this work, constructing a Deep-Learning framework provides a plethora of gains which is good to referred to. One of the most important artifacts is the ability to transformed from strictly scientific work, to something that can be put to practical use. Being familiar with Convolutional Neural Networks, is a learning. The knowledge is optimized to an upgraded level by working on in this field.

5.4 Contributions

Finally, those in which contribute are much than the desirable in the beginning. Two applications created, easy to work from experts and non-experts. This whole implementation, can also considered as framework. This because of the ability somebody to download the code from GitHub, and with an easy implementation due to design create his own application. It is giving compatibility, for training with images, and test in different systems because of the java. All the aforementioned gathered to a Deeplearning4j GUI, and creates an application which exist also few more.

6. References

[1] L. Lab, "deeplearing.net," [Online]. Available: http://deeplearning.net/tutorial/lenet.html. [Accessed 10 12 2015].

[2] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Supervised_learning. [Accessed 9 12 2015].

[3] A. P. Engelbrecht , Computational Intelligence An Introduction, 2007.

[4] M. Mohri , A. Rostamizadeh and A. Talwalkar , "Foundations of Machine Learning," *The MIT Press,* 2012.

[5] M. I. Jordan and C. M. Bishop, "Neural Networks," in *Computer Science Handbook*, Boca Raton, Chapman & Hall/CRC Press LLC., 2004.

[6] R. S. Sutton , "Temporal Credit Assignment in Reinforcement Learning (PhD thesis)," Massachusetts, 1984.

[7] A. H. Song and Y. S. Lee, "Hierarchical Representation Using NMF," in *Neural Information Processing. Lectures Notes in Computer Sciences 8226.*, Springer Berlin Heidelberg, 2013.

[8] B. A. Olshausen, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature* , pp. 607-609, 1996.

[9] R. Collobert, "Deep Learning for Efficient Discriminative Parsing," VideoLectures.net., 2011.

[10] L. Gomes, "Machine-Learning Maestro Michael Jordan on the Delusions of Big Data and Other Huge Engineering Efforts," *IEEE Spectrum.,* 20 October 2014.

[11] "deeplearning4j.org," 2. [Online]. Available: http://deeplearning4j.org/. [Accessed 27 1 2016].

[12] "caffe.berkeleyvision.org," Berkley University, [Online]. Available: http://caffe.berkeleyvision.org/. [Accessed 27 1 2016].

[13] "torch.ch," [Online]. Available: http://torch.ch/. [Accessed 29 1 2016].

[14] T. Liu,, S. Fang,, Y. Zhao,, P. Wang, and J. Zhang, "Implementation of Training

Convolutional," University of Chinese Academy of Sciences, Beijing, China.

[15] D. H. Ackley , G. E. Hinton and T. J. Sejnowski , "A Learnimng Algoritmh for Boltzman Machines," *Cognotive Science*, pp. 147-169, 1985.

[16] Z. Wang, "The Applications of Deep Learning".

[17] "Caida," [Online]. Available: http://www.caida.org/~alice/docs/bsod.html. [Accessed 30 06 2016].

[18] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Google_Brain. [Accessed 30 06 2016].

[19] "Google," [Online]. Available: https://googleblog.blogspot.gr/2012/06/using-large-scale-brain-simulations-for.html. [Accessed 30 06 2016].

[20] Y. Taigman, M. Yang, M. Ranzato and L. Lior, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *CPVR2014*, 2014.

[21] "welivesecurity," [Online]. Available: http://www.welivesecurity.com/2014/03/19/facebooks-deepface-photo-matching-is-nearly-as-good-as-human-brains/. [Accessed 30 06 2016].

[22] L. Takeuchi and Y.-Y. A. Lee, "Applying Deep Learning to Enhance," 2013.

[23] A. Krizhevsky, I. Sutskever and G. E. Geoffrey, "ImageNet Classification with Deep Convolutional," Toronto.

[24] A. Karpathy, "http://karpathy.github.io," [Online]. Available: http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/. [Accessed 30 06 2016].

[25] "tensorflow," google, [Online]. Available: https://www.tensorflow.org/. [Accessed 28 01 2016].

[26] "ros," [Online]. Available: http://www.ros.org/. [Accessed 30 06 2016].

[27] "pypi.python.org," [Online]. Available: https://pypi.python.org/pypi/Theano. [Accessed 30 06 2016].

[28] "developer.nvidia.com," [Online]. Available: https://developer.nvidia.com/cuda-gpus. [Accessed 28 01 2016].

[29] "developer.nvidia.com," [Online]. Available: https://developer.nvidia.com/digits.

[30] H. Dutta, "https://github.com/jann2005," [Online]. Available: https://github.com/jann2005. [Accessed 08 02 2016].

[31] "GitHub," [Online]. Available: https://github.com/. [Accessed 15 02 2016].

[32] "deeplearning4j.org," [Online]. Available: http://deeplearning4j.org/glossary.html. [Accessed 04 08 2016].

[33] "wang.ist.psu.edu," [Online]. Available: http://wang.ist.psu.edu/docs/related/. [Accessed 05 07 2016].

[34] J. Z. Wang, J. Li and G. Wiederhold,, "SIMPLIcity: Semantics-Sensitive Integrated," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* vol. 23, no. 9, pp. 947-963, 2001.

[35] H. Mobahi, R. Collobert and J. Weston, ""Deep Learning from Temporal Coherence in Video," *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09),* pp. 737-744, 14-18 June 2009.

[36] "tomcat.apache.org," [Online]. Available: http://tomcat.apache.org/. [Accessed 20 03 2016].

[37] j. Li and J. Z. Wang,, "Automatic Linguistic Indexing of Pictures," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* vol. 25, no. 9, 2033.

[38] "www.vision.caltech.edu," Caltech, [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech101/. [Accessed 05 07 2016].