VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELGAUM – 590014

A Project Report on

"Natural Scene Classification using Convolutional Neural Networks"

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Engineering
in
Information Science & Engineering

Submitted by:

INDUJA V	**1PI12IS035**
MEGHANA KANTHARAJ	**1PI12IS056**

Under the guidance of

Internal Guide
S.Natarajan
Professor,
Department of IS & E,
PESIT

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

PES INSTITUTE OF TECHNOLOGY

100 Feet Ring Road, BSK 3rd Stage, Bengaluru – 560085

January 2016 – May 2016

PES INSTITUTE OF TECHNOLOGY

100 Feet Ring Road, B S K 3rd Stage,
Bengaluru-560085

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"Natural Scene Classification using Convolutional Neural Networks"** carried out by **Induja V** , bearing USN **1PI12IS035** and **Meghana Kantharaj**, bearing USN **1PI12IS056**, are bonafide students of **PES INSTITUTE OF TECHNOLOGY**, Bangalore, an autonomous institute, under VTU, in partial fulfillment for the award of degree of **BACHELOR OF ENGINEERING IN INFORMATION SCIENCE & ENGINEERING** of **Visvesvaraya Technological University, Belgaum** during the year **2016**. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the above said degree.

S.Natarajan Professor Department OF ISE PESIT	**Dr. Shylaja S S** Professor and Head, Department of ISE PESIT	**Dr. K. S. Sridhar** Principal PESIT

External Viva

Name of the Examiners	**Signature with Date**
1._____	_____
2._____	_____

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

P.E.S. Institute of Technology

Department of Information Science and Engineering

Bengaluru – 560085

DECLARATION

We, **Induja V and Meghana Kantharaj**, students of Eighth Semester B.E., in the Department of Information Science and Engineering, **P.E.S. Institute of Technology, Bangalore** declare that the project entitled **"Natural Scene Classification using Convolutional Neural Networks"** has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree of **Bachelor of Engineering** in **Information Science and Engineering of Visvesvaraya Technological University, Belagavi** during the academic year **2015-16**. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

Name and USN **Signature**

Induja V [1PI12IS035]

Meghana Kantharaj [1PI12IS056]

ACKNOWLEDGEMENT

It is our privilege to express our sincere regards to our project guide, **Dr. S.Natarajan**, Professor, Department of Information Science, PESIT for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the implementation and presentation of this project- "Natural Scene Classification using Convolutional Neural Networks". We wholeheartedly thank you for the support

We deeply express my sincere thanks to **Dr. Shylaja S. S**., Head of Department, Department of Information Science, PESIT for her support and guidance and by encouraging and allowing us to utilize the department resources to their capacity and present this project at our department premises for the partial fulfillment of the requirements leading to the award of BE degree.

We would like to express our gratitude to **Dr. M.R. Doreswamy**, Founder and Secretary, PES Institutions, **Prof. D. Jawahar**, CEO, PES Institutions and **Dr. K.S Sridhar**, Principal, PESIT for providing us with a conductive environment for executing this project

We take this opportunity to thank all our lecturers who have made us capable enough to execute this project and hence directly or indirectly helped in this presentation. Last but not the least we express our thanks to our friends for their cooperation and support and our parents for their supportive and nurturing supervision.

ABSTRACT

The task of classifying images of natural dynamic scenes into appropriate classes has gained lot of attention in recent years. The problem especially becomes challenging when a camera used to capture the input which would be video that is dynamic. Convolutional neural networks have recently been used to obtain record-breaking results in many vision benchmarks. In addition, the intermediate layer activations of a trained network when exposed to new data sources have been shown to perform very well as generic image features, even when there are substantial differences between the original training data of the network and the new domain. In this project, we focus on scene recognition and show that convolutional networks trained on mostly scene recognition data can successfully be used for feature extraction in this task as well. We train networks with different training data and architectures, and show that the proposed method combining multiple features obtains good performance on standard scene datasets. However, performance at scene classification has not achieved the same level of success since there is still semantic gap between the deep features and the high-level context. Considering this fact, we have modeled our convolutional neural network to classify image scenes into four, six and eight classes each achieving 59%, 61& and 64% efficiency respectively.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Humans are extremely good at perceiving natural scenes and understanding their contents. However, we know little about how or where in the brain we process the natural scenes. How is it that the brain determines whether it is looking at beach or a mountain? Work on this project is concerned with how we categorize natural scenes; that is, how do we process different images we encounter in everyday life?

Research has been done by the most prestigious labs on understanding human vision. This has been a major field of interest as it is one of the basic questions that envelop life. The research done until now have been able to explain how object recognition is so easy for human beings. The processing of visual data collected by human eyes happens in the ventral visual stream, which is simply, a hierarchy of areas of the brain that aid helps in object recognition. Humans easily recognize different sized and colored objects, and classify them in the same category. This happens because of the invariances in the visual data we develop while processing.

When we look at any object, our brain extracts the features swiftly in such a way that the size, orientation, illumination, perspective etc. doesn't matter. We tend to remember an object by its shape and underlying inherent features. It doesn’t matter how the object is placed or aligned, how different the illumination is, how big or small it is or what side of the object you view. There is a hierarchical build-up of invariances, as mentioned earlier, first to inherent features of the object, then to position and scale and then to point of view and the more complex transformations requiring the interpolation and merging between several different object views from multiple viewpoints of slightly differing angles of sight.

We have the cells in our visual cortex which respond to simple shapes considering their lines and curves for processing. As we move along the visual cortex, the cells get more complex. These cells respond to more complex objects like faces, cars etc.

Further, neurons along the ventral stream show an increased size in receptive field as well as in the complexity of the stimuli they cater to. Humans take surprisingly appreciable little time to recognize objects and categorize them respectively.

From the above observation, we can derive that there is some sort of feedforward processing of the given information going on which means that the information processed by the cells in the current level of the visual cortex in the ventral stream hierarchy is used in the next level of this hierarchy. This process of feedforward helps speed up the process by majorly thus making object recognition by human beings so quick and efficient.

Now that we know the mechanism by which humans recognize objects when the visual data enters the human visual system, and also how the data gets processed, we try to implement our findings to validate the same. The problem arises where we are still not very sure about how our brain categorizes the processed data and how they get organized.

In order to substitute for these tasks to form a good model of human visual system, we extract features from an image and make the computers learn from the processed data. In the real world, objects vary in many ways some of them being-size, illumination, angle, occlusion, perspective etc. The variations in these make the same object look very different from each other to a machine as it is presented and viewed in a different perspective.

It is easy for human beings to immediately recognize these objects from every point of view, but it is challenging for computer machines to do the same job, even in a longer time. One way to approach this setback would be to store all possible sizes, angles, perspectives at all illuminations etc. This would be illogical, inefficient and infeasible as it would take an enormous amount of space and time to recognize an object even if we invest the huge amount of money required. Human beings can identify a partially occluded chair and categorize it accordingly but with machines, there is a high chance of them failing to do so in similar situations because the occluded chair is now considered as a new object by them.

Scene understanding is the ultimate step of artificial human natural vision models and is a challenge for machine vision because a scene contains predictive information on multiple scales of processing. Computational models of scene recognition have attempted to identify scenes and use them for image classification.

Humans do a better job in categorizing rivers, lakes and mountains when the presented images are globally blurred than locally scrambled, but they classify coasts, forests and plains better locally scrambled than in globally blurred images. In addition, intact images are easier to identify than manipulated ones. Similar evidence indicate that neither local nor global

information is more predictive than the other at all times, and that the brain makes use of scenic information from multiple scales in images for scene recognition.

If a new technology has to be accepted and acknowledged it has to fulfill certain criteria such as it should be fast enough and have sufficient high accuracy. Generally object recognition is computationally very intensive as many features are considered and yet the accuracy is comparatively low. The technology where you point your machine device at something and it being able to completely understand its surroundings is far from reality, but we are advancing towards it.

Addressing the problem under discussion, some powerful features and algorithms have been recognized and formulated in the recent years. The features considered are invariant to scale, rotation and also illumination to a certain extent. Each of these feature descriptor attributes have their own advantages and disadvantages listed. Using these features, researchers have tried to come up with their own algorithms and formula which attempt to be as close to the human visual system as possible. The purpose of this is to increase the efficiency in so that the recognition accuracy increases as the current technologies are able to achieve an accuracy of around 65-70% only, depending on the data set, machine used and environment conditions.

Scene classification is an important problem for computer vision. It has received considerable attention in the recent past. It differs from the conventional object classification, as in a scene is composed of several entities often organized in an unpredictable layout. Early efforts at scene classification targeted problems such as distinguishing indoor from outdoor scenes etc. Subsequent research was inspired by the literature and biological views on human perception. More recently, there has been an effort to solve the problem in greater generality, through design of techniques and models that are capable of classifying relatively large number of scene categories. These methods rely on local region descriptors, viewing an image as an order less collection of descriptors.

The space of local region descriptors is then quantized, based on clustering mechanism, and the mean vectors of these clusters are chosen as their representatives. A set of cluster means forms a guide, and a scene is characterized as a frequency vector over these in the guide. These descriptors are formed into feature vectors which are passed as inputs through an artificial machine which performs brain like functions to learn the characteristic features of the objects. Researchers have designed multiple models that mimic the activities performed by the brain

with respect to visual applications. This field of research has been termed as computational neuroscience.

Computational neuroscience is the field of study that deals with understanding brain function in terms of the information processing properties of the biological structures that make the human nervous system. It is an interdisciplinary science which encompasses diverse fields like computer science, neuroscience, cognitive science, psychology, electrical engineering, mathematics and physics.

Though they have common applications, the field of computational neuroscience differs from psychological linkage and machine learning. It focuses on functional and biologically realistic neurons and subsequently, neural systems studying their physiology and dynamics. The resulting models encapsulate the essential features of the human biological system at multiple spatial-temporal scales such as learning and memory.

Therefore, these computational models help formulate hypotheses that can be tested by biological and psychological experiments in real living organisms for correctness. The most prominent and widely used technology for mimicking human thought processes happens to be neural networks.

In the hemispheres of human brain, we have a primary visual cortex, known as V1, which contains 140 million neurons and 10 billion of connections between the neurons. Human vision involves not just V1 stage, but also series of visual cortices called V2, V3, V4, and V5, of which each does more complex image processing compared to previous stage.

Human brain has been trained and tuned by evolution since the inception of mankind and we adapted to understand the visual information world provides. Recognizing scenes isn't easy but, we are extremely good at making sense of what we see and all the image processing work is unconsciously done.

Neural networks approach the problem in a way similar to how human brain does it. The idea is to take a large number of images, known as training examples, and using them, develop a working system that learns from these training examples. Simply put, the neural network uses the day to day examples to automatically infer rules for recognizing underlying patterns. Further, when the number of training examples increases, the learning is better as the network being trained can learn more about objects variations, and hence improve its accuracy.

Neural networks are primitively an organization of layers which are made up of a number of interconnected nodes called neurons, each of which contain an associated activation function. Patterns in the real world are presented to this interconnected network through the input layer. This input layer is connected to hidden layer and communications happen between one or more of these hidden layers. This is where the actual processing is done with the support of a system of weights.

The hidden layers then link to the output layer, the last layer in the network, where the class in indicated. Neural Networks contain a simple learning rule that modifies the weights of the connections according to the patterns of provided input that it is presented with. In a sense, neural networks learn by example also with trial and error as their biological counterpart they replicate, the human brain, do.

Learning is a supervised process which that occurs in every cycle or epoch through a forward activation transfer of outputs, and the backwards error movement of weight adjustments. In simpler terms, when a neural network is initially presented with an input pattern it makes a random guess as to what the input might be, and then it later sees how different the answer it generated was from the actual class and makes the required adjustments to the connection weights in the neurons.

Once a neural network is trained to a satisfactory level it may be used to analyze, explore and classify other data. This can be done when the user no longer has to specify any training runs and instead allows the network to run in forward propagation only. New inputs are passed to the input layer. Here, they into and are processed by the hidden layers just as in the training phase, however, here the output is retained and backpropagation doesn't occurs. The output of a forward propagation epoch is the predicted class for the input data.

Over-training of a neural network is also a possible outcome, meaning that the network has been trained exactly to respond to only one type or class of input and when this happens, if this happens the learning can no longer occur as the network can no longer consider any other class than the one it has over trained for, rendering it redundant and blind.

Neural networks are considered to be good at approximations. They work best given the system you are using them for a model has some good high tolerance to error. They are more suitable for:

- capturing associations or discovering regularities between the input set of patterns
- in cases where the volume or number of variables or diversity of the considered data is high
- the relationship between the set variables, though ambiguous, are vaguely understood
- the relationships between variables are difficult to describe adequately with simpler, easier or more conventional approaches.

Depending on the application's purpose and the strength of the patterns contained within data internally, the network generally trains quite well given that the neural network is used for a suitable problem where the relationships may either be dynamic or non-linear. Neural Networks serve as an efficient alternative analytically to conventional techniques which quiet often get limited by strict rules and assumptions in terms of normality, linearity, variable independence etc. As a Neural Network has the ability to capture various kinds of relationships, the users can also observe a quick and relatively easy model at work, which otherwise may have been very hard or impossible to solve.

Neural networks, for good results, require a large number of independent runs many times to determine the best solution or class. Neural Networks are a great way to develop more advanced techniques of machine learning, such as deep learning.

Deep learning can be defined as an artificial neural networks that is composed of several layers. It is a growing trend in machine learning as it yields favorable results in certain applications where the target function is quiet complex and the input data-sets are huge.

Deep learning is termed so because of their comparison with the structure of artificial neural networks. Forty years ago, these neural networks had depth of only 2 layers deep as the machines weren't equipped in hardware to allow more layers on larger networks but now, due to advance of technology in hardware, we can commonly see neural networks with 100+ layer being implemented.

The layers can be stacked. These layers are made of neurons on top of each other. The first ie.., lowest layer takes raw input data like images, text, sound, etc. Each neurons stores some information about the data they process during encounter. Every neuron in the layers send information they process up to the next layer of neurons, which generally learn a more abstract

version of the data passed to it. The higher up you go, the more abstract the features you learn will be.

In deep learning, neural networks are algorithms that extract features automatically with deep learning whereas many older technologies engaging in extracting features manually during feature engineering. For example let's consider an RGB image as the input where all we have to do is input the raw images into the artificial neural network bypassing the process of us taking the input image and manually computing its features such as the distribution of its colors, number of distinct colors, etc., .Artificial Neural networks have excelled in image processing applications, now they are being tried with all kinds of other input datasets such as raw text, numbers etc. Here, progressively they are proving quiet reliable and efficient.

Where neural networks prove advantageous is that we don't necessarily need to figure out the features beforehand. Along with this, advantage is that Neural networks is flexible as it can be used for many other problems as a substitute for algorithms like Support Vector Machines, Supervised Linear Classifier, Regression, Bayesian Network, Decision Trees, Clustering and Association Rules. Also, the strength of neural networks lies in its fault tolerance capabilities and is very scalable.

Deep Learning is dynamically expanding set of breakthrough techniques dealing with multiple layer neural networks training rather than being one kind of neural network. The first few originals designs of Deep Learning techniques were the Convolutional Neural Networks, Restricted Boltzmann Machine and Sparse Autoencoders.With the help of these algorithms, many complex problems have found solutions and many others have observed increase in efficiencies. Convolutional neural networks have especially found their calling in image processing problems.

Convolutional neural networks are a special type of artificial neural networks. Though it is a type of Deep learning architecture, it has some more added feature which help it generate the good results out of extremely complex problems. They are simply put, neural networks with utilize convolutions. We may understand convolutions to be a sliding window function applied to a matrix that represents the input. This sliding window is called by various names such as kernel, filter, or feature detector.

The kernel is also a matrix itself of dimensions n x n whereas the input matrix is of bigger dimensions than the filter. The kernel values are multiplied with the corresponding values

element-wise with the original matrix and then they are added to the generated values. For a full convolution layer, this is done for each element in the input matrix by sliding the filter over the whole matrix.

CNNs are primitively several layers of convolutions stacked, with nonlinear activation functions like ReLU or tanh. These are called activation functions and are applied to the results obtained. Each input neuron is connected to each of its corresponding output neuron of its corresponding layers in the traditional neural network architecture. The technical term given to these layers in ANN architecture is fully connected layer, also affine layer. But contrastingly, in convolutional neural network architectures, we do not connect each output to each input of the corresponding layer whereas convolutions are applied over the entire input layer in order to compute the output of that layer.

Having convolutions applied to input results in local connections, where every region of the input is also linked to a neuron in the output. In this case, each convolutional layer applies different filters, typically in the range of hundreds or thousands, thus merges the computed results. Convolutional neural network architecture also includes another addition compared to regular neural networks in the form of something called pooling or subsampling layers.

An important hyper parameter of Convolutional Neural Networks are stride size of the convolutions. Stride size is nothing but the definition of the shift we want in terms of input pixels after every step of the convolution filter. Generally the stride size is set to 1 ie.., consecutive pixels start off the next convolution and consecutive application of the filter are made to overlap. The bigger the stride, the lesser the number of times we will have to apply the filter onto the image, and the smaller the output size.

A significant aspect to convolutional neural networks are special layers called pooling layers which are typically present after the application of convolutional layers. In these layers, the inputs are subsampled accordingly, where the simplest way to achieve this is by applying a max function over the results obtained after each filter. Pooling, like convolutions, do not have to be applied over the entire matrix range, but can be applied over tinier windows too.

Pooling is done for many reasons out of which one property of pooling is the fact that it provides a fixed size output matrix, which traditionally is necessary for further classification processes. Consider an example where there are a thousand filters in convolutional layers and to each of them max pooling is applied. This will result in a 1000-dimensional output without

consideration of the size of the taken filters or the size of the considered input. This quality of the pooling layer allows us to be able to use variable size images and also variable size filters and in spite these, we get the outputs in the same dimensions which can be feed into a classifier.

Another advantage of pooling is that it also reduces the output dimensionality as required while keeping the most important information contained intact. Each filter can be imagined to have detected a specific feature and if this extracted feature happens to be present in the input image somewhere, the yielded result of the application of the filter to that region will contain a large value compared to the small values obtained from the other regions of input image and when the max operation is performed, the information regarding whether or not the marked features from the input image are kept track of but the details of where exactly those features were extracted from is lost in the extraction.

During the training stage, a convolutional neural network learns the values of its filters all by itself, the learning being based on the task that is requires to be performed. Considering an example in Image Classification, the convolutional neural network may have to learn to detect edges from the raw pixels observed in the first layer and then use these detected edges to further detect simple shapes during the second layer, and then consecutively use these obtained shapes in order to detect the higher-level patterns contained, such as objects or facial shapes in corresponding higher layers. The final layer is then typically a classifier that makes efficient usage of these obtained high level features.

The two important aspects of classification with CNNs which are-

- Location Invariance
- Local Compositionality

Invariance can be explained with an example. Consider an image in which you want to determine if a car is present or not. You will start processing with convolutions. With each layer, due to the sliding of filters over the entire image, the image becomes abstract, and the position of the car, if present in the image, will not matter. Similarly, pooling makes criteria such as translation, rotation and scaling invariant.

The other important aspect to consider is local compositionality. We know than in convolutional neural networks, the consecutive layers compose a lower-level representation to a higher-level representation with each filter. This is the reason why convolutional neural

networks are significant to Computer Vision. Therefore it is reasonable that we build lines and edges from variance, shapes from lines and edges, more complex objects from objects based on locally available information.

Summarizing, a Convolutional Neural Network can be built by designing the architecture according to the needs of the problem at hand. The various criteria that has to be decided in an architecture are-

- Input representations
- convolution filters, their size and quantity
- pooling techniques (average, max functions)
- activation functions (tanh, relu)

CNNs are observed to run very fast compared to other techniques in certain problems. Also they carry out representations of their products quiet efficiently. The filters learn as the training progresses to ensure a better understanding of the input. Convolutions form a major part of computer graphics. Also, they happen to be implemented on the hardware level of GPUs. For these reasons, Convolutional Neural Networks prove to be a popular choice for image recognition problems.

Convolutional neural networks have implemented many huge problems, some of the most prominent ones are ImageNet Large Scale Visual Recognition Challenge which have yielded highly efficient architectures such as GoogLeNet, AlexNet, etc. These architectures on being implemented have been proven so efficient that the performance of tests on ImageNet are Quiet close to human performance. Despite of these successes, even the best algorithms struggle with thin or small objects, such as feather quills, flower stems, ants. Also, they face problems with non-natural scenes or images i.e.., images that have been distorted by picture filters. Compared to machines, humans still hold an upper hand in image recognition, but work is being done at the moment to better computer vision in these blind spots.

CHAPTER 2

PROBLEM DEFINITION

CHAPTER 2

PROBLEM DEFINITION

Human vision is a marvel, especially from machine vision point of view. Modelling a vision system based on human vision is a highly complex methodology and though good efficiencies have not been achieved, the results have been satisfying. The models are being worked upon for better scene recognition. In this project, we modeled a classification system to classify images into four, six and eight classes in separate subsequent tests. The classification was done using Convolutional Neural Network, which is apt for image classification based problems.

Image dataset are first pickled. These pickled files are then used for further processes. The images are passed as inputs to convolutional neural networks, where convolutional layer, pooling, dropping, nonlinear transformations are some of the important functions applied on the input vectors. After the application of the layers of the architecture model, the classes are predicted for the testing dataset of images.

CHAPTER 3

LITERATURE SURVEY

CHAPTER 3

LITERATURE SURVEY

In the past decade, a lot of work is being done in this field of scene classification since the past decade which includes the recognition of natural scenes from single image inputs to being able to classify natural scenes taken from videos. Here, in this section, we will discuss some of the interesting previous works which directly relate to our project.

Koskela et al., in their project have focused on scene recognition and tagging and have shown that convolutional neural networks which are trained mostly on object recognition oriented data can be used successfully for feature extraction applications as well as they are used for object reccognition. Their implemented model consists of training a total number of four convolutional neural networks with different training data and experimentation with different architectures. They also show that the previously proposed methods of combining various multiple scales and multiple features in order to obtains excellent performance when four different natural scene datasets are used.

Scurka et al., Grauman et al., Jegou et al., Perronin et al. and Wang et al. have, in this field of single image recognition tasks have used bag-of-features based approach towards their problem statements. Bag-of-words was popular among research communities and these methods were directly based off the principle of geometric independence or spatial representation of the byproducts.

Later on, the discussed methods were further enhanced by the inclusion of weak geometrically weak information with the help of spatial pyramid matching as done by Schmid et al. This particular technique is used for histogram intersection at the various stages of the pyramids for the matching of features.

Although, approaches that rely on convolutional neural network has been successful in achieving comparatively even higher accuracies as we can observe in some of the recent works done by Gong et al, Krizhevsky et al., Sermanet et al., Zeiler et al. With their research, a lot of interest has been generated in a lot of recent research works done on architectures and the applications of CNNs for computer visual classification and also recognition tasks. Krizhevsky

et al. have designed the CNN architecture such that it consists of five convolutional layers which is followed by two fully connected layers or neural networks which is of dimensions 4096, followed by an output layer.

The work done by Zeiler et al. indicates the output that we get from the fifth max-pooling layer was shown to preserve the global spatial information. Even the layers of activation features taken from the fully connected layer were discovered to be quiet sensitive to global distorting criteria such as rotation, translation, and scaling as shown by Gong et al. although, these have been proven to be very powerful yet general feature descriptor tools for high level computer vision tasks. These experiments have paved way for many similar techniques being worked on currently. Gong et all have pre-trained on large datasets with CNNs, with datasets such as ImageNet and have been proven to be efficiently used in scene classification along with having achieved quiet impressive accuracies.

Sermanet et al., Donahue et al. and Jia et al have experimented with several early CNN implementations like DeCAF, Caffe and OverFeat and have trained on very large datasets that are available for the process of feature extraction stage in order to perform image classification tasks and have compared the performances of the different implementations. Also Donahue et al and Zeiler et al. have the ImageNet trained model of these discussed implementations and they have been shown to generalize obtained results well in order to accommodate other datasets as well.

Razavian et al. have CNN features obtained from object recognition datasets which have also been used for obtaining higher accuracy in various high level computer vision problem statement tasks. But on the other hand, research in natural scene classification from images has been dominated and encapsulated by the idea of finding the more powerful yet robust local spatial-temporal feature descriptors which is followed by the embedding of weak global information in order to find the most appropriate representation of the given input images.

Derpanis et al. have experimented with the spatial-temporal based approaches that were introduced by spatial-temporal oriented energies which also resulted in the introduction of the YUPenn dataset. Along with these conclusions, the very same work concluded that even minutely relatively simple spatial-temporal feature descriptor attributes were able to achieve consistently higher performance with both of the datasets, the YUPenn as well as Maryland datasets, compared to earlier approaches.

Currently, Feichtenhofer et al. have devised the state-of-the-art approach with the bags of time approaches along with proposing a bag of visual words for natural scene classification and here the local features which were extracted via some spatio-temporally oriented filters are employed for classification and these are encoded using a learned dictionary and then consequently dynamically pooled. Also, this technique currently holds the highest accuracy on the two previously mentioned datasets compared to all other peer-reviewed studies.

The recent work done by Duran et al. has used a novel three dimensional CNN architecture for spatio-temporal scene classification problems where this methodology is shown to have promising results along with marginal performance improvement over currently followed methods. Another work done by Xu et al. recently has used Caffe framework for CNN implementation in order to obtain better than the current state of art performance for the event detection problem with the classification being done on the TRECVID-MED dataset.

CHAPTER 4

SYSTEM REQUIREMENTS SPECIFICATION

CHAPTER 4

SYSTEM REQUIREMENTS SPECIFICATION

A System Requirements Specification is a structured collection of information that embodies the requirements of a system. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. The functional requirements define what functions the software should perform and non-functional requirements specify the constraints on design and implementation and other performance and scalability related details. Requirements must be measurable, testable, and related to identify the needs or opportunities, and be defined to a level of detail sufficient for system design. The software requirements specification document enlists enough and necessary requirements that are required for the project development.

4.1 HARDWARE REQUIREMENTS

The hardware requirements for this project are:

1. Minimum 4 GB of secondary memory space should be available for storing the dataset and processing them.

2. For good processing speed, a minimum of 4 GB RAM is needed.

3. For faster processing with a bigger dataset, GPU is required.

3. A 64 bit system is preferred for faster processing.

4.2 SOFTWARE REQUIREMENTS

The software requirements for this project are:

1. Python 3

2. Python libraries used:

- Os
- Numpy
- Pickle
- PIL
- Random
- Gzip
- Math
- Theano
- Functools
- Operator
- Time
- Tkinter
- Sys
- Socket
- Importlib
- datetime

3. An image viewer software to view images and the ROC curves and confusion matrices.

4. A text editor to view and edit python code

5. Any Operating system that supports the above mentioned requirements. But the application has been tried and tested on Ubuntu 12.xx and Windows 10.

4.3 NON-FUNCTIONAL REQUIREMENTS

4.3.1 DEPENDENCIES

1. The user should ensure the input files are not corrupt and is responsible to maintain the integrity of the files.

2. The user has to ensure he/she does not specify a critical directory (such as root) and also not keep any other files other than images.

3. The application requires permission to read, write and create files. Hence, the necessary permissions must be granted and appropriately specified.
4. Image Processing takes time since it has to process 6000 images.
5. Training of the model also takes time as it has to process large dataset.
6. User input for testing the model should be of a valid type and should not be a mix of two or more considered categories.
7. The user should not stop the execution of the process midway.

4.3.2 ASSUMPTIONS

Following are the assumptions made-

1. The software dependencies have all been accurately installed without any errors or warnings.
2. Images used for testing and training exclusively belong to a single category.
3. The hardware on which the program is running has power backup.
4. The system has the required space.
5. The system has the ability to cache the model and decode it when necessary.

4.3.3 USER REQUIREMENTS

The user requirements is represented by the use case diagram. Use case diagrams generally indicate the relationships between actors and use cases. Use case diagram help represent the model a system or a subsystem of an application. One use case diagram expresses how a module can be used by a user and in order to express how a system works a number of use case diagrams may be drawn. The use case diagram for this project is as follows-

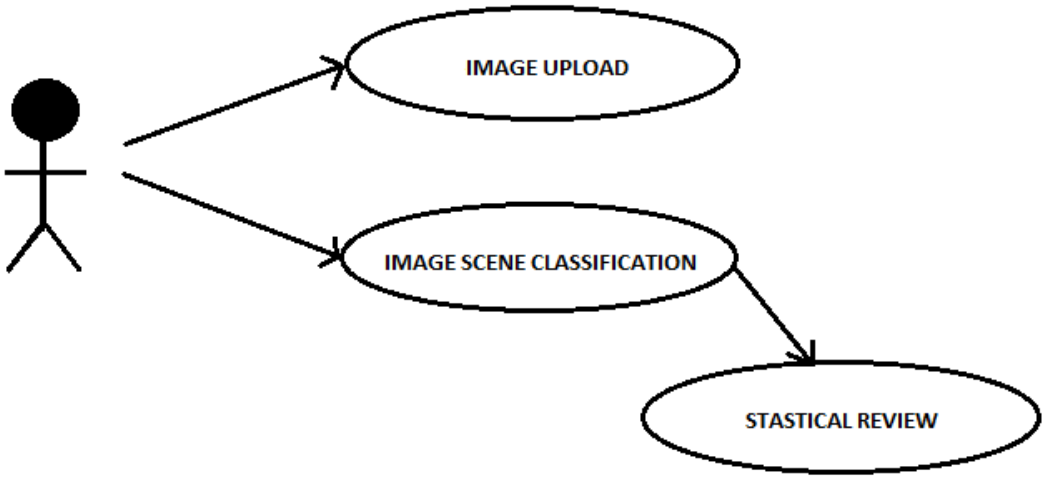

Fig 4.1 Use case diagram

4.4 GANTT CHART

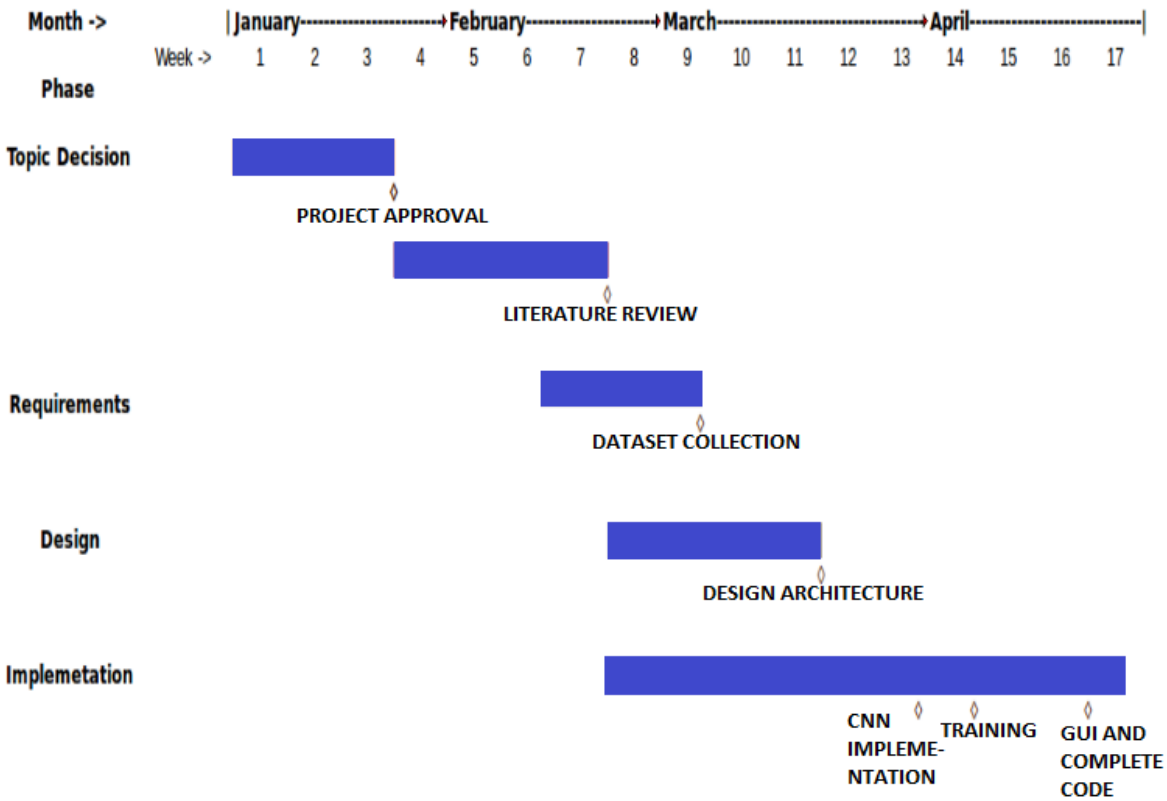

Fig 4.2 Gantt chart

CHAPTER 5

SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

5.1 BLOCK DIAGRAM

The block diagram of this project is as follows-

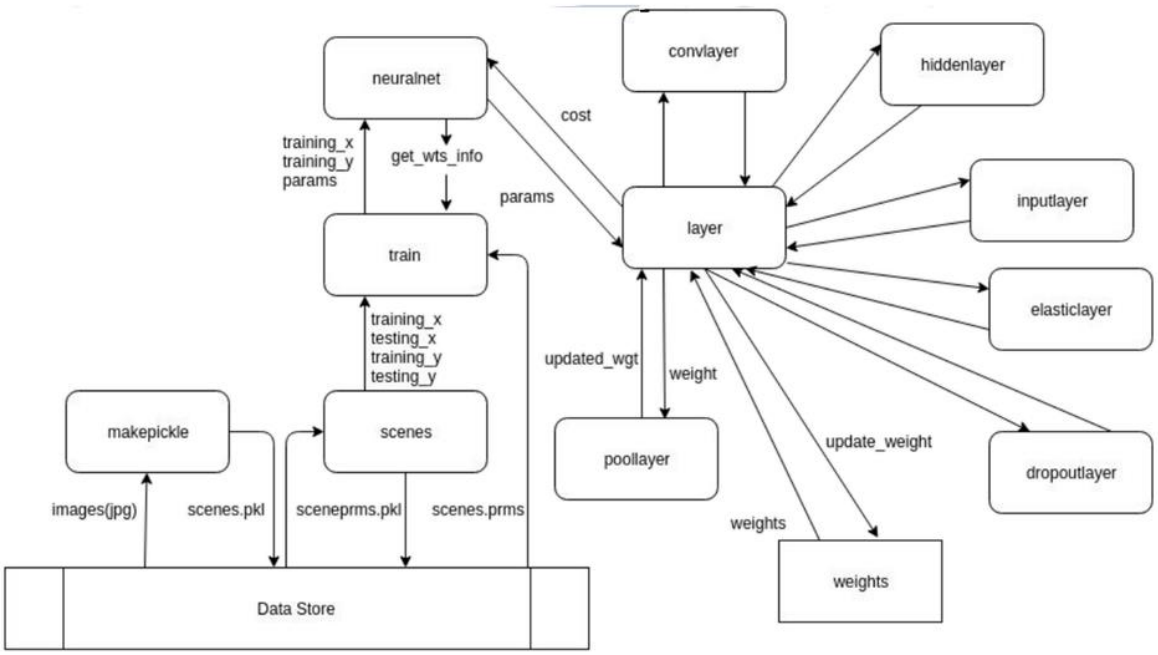

Fig 5.1 Block diagram

5.2 IMPLEMENTED MODULES

The implementation of Convolutional neural networks involve a number of modules each exhibiting and executing certain actions the conventional model of convolutional neural network is entitled to carry out. The modules implemented in this CNN module have been listed below-

- **Train.py**

  This module represents the main working code of this project. This module is used to run the implementation as it holds together all the basic elements that

make the Convolutional neural network model. This model is run to train the CNN architecture for the given input dataset.

- **Makepickle.py**

  This module deals directly with the reception of the input dataset. This is the module intended to be executed first. It inputs the images from the dataset and converts them into python objects. This module generates the input vectors for the Convolutional Neural network.

- **Scene.prms**

  This file contains the details about the other modules that make up the convolutional neural network. It contains the several variables and attributes, along with their values that are require to describe the various stages of a convolutional neural network.

- **Scene.py**

  This module forms a connecting pathway between the input and the training network. The input which had been converted into input vectors as python objects are read and loaded into the convolutional neural network architecture with the help of this module.

- **NeuralNet**

  This module is the full implementation of an artificial neural network. Artificial neural networks are used as the fully connected layers at the ending stage of a regular CNN in order to classify the input into its category from its abstract version.

- **Layer.convlayer**

  This module implements the convolutional layer part in a convolutional neural network. This layer is important to a convolutional neural network as it implements the concept of convolutions while reading an input into the network.

- **Layer.poollayer**

  This module implements another important definitive aspect of a convolutional neural networks. This module represents the pooling layer which is used to subsample the inputs to form a distinctly more abstract version of the processed inputs.

- **Layer.hiddenlayer**

  This layer forms the hidden layers of a fully connected neural network. These layers form the processing part of an artificial neural network. Upon passing through this layer, the input is converted to a form from which its class is more predictable.

- **Layer.inputlayer**

  This layer is the bridge between the inputs and the network. This layer reads the input from the necessary format and forwards them into the network for further processing.

- **Layer.outputLayer**

  This is the module that is the final module the processed input has to go through in order to generate the definitive class of the input stage. This module represents the final stage of a fully connected artificial neural network.

CHAPTER 6

IMPLEMENTATION

CHAPTER 6

IMPLEMENTATION

6.1 DATASET COLLECTION

The dataset has been collected for various sources, the primary source being from the MIT website. This dataset is owned by the Computational Visual Cognition lab and the name of the dataset is Urban and Natural Scene Categories. This dataset consists of 8 categories of images, the 8 categories being-

- Coast
- Highway
- Mountains
- Open country
- Inside city
- Tall buildings
- Forest
- Street

Each database is composed of a few hundred images of scenes belonging to the same semantic category. All of the images are in color, in jpeg format, and are 256 x 256 pixels. The sources of the images vary, they have been collected from various commercial databases, websites, and digital cameras.

The given image summarizes the number of picture sample in each category-

CLASS	Number of Images
Opencountry	820
Street	584
Forest	656
Highway	520
Coast	720
Mountain	748
Tallbuilding	712
Inside_city	616

Fig 6.1 Dataset summary

The input dataset split that has been used for this project is 75:25 split i.e., if the database consists of 100 images totally, 75 images have been used for training the network and 25 images have been used for testing the implemented convolutional neural network model.

6.2 PICKLING THE DATASET

The dataset, which contains the training images has been pickled using the makepickle.py module. Pickling is a useful python process. It is used for serializing and de-serializing a Python object structure. Any object in python can be pickled so that it can be saved on disk. What pickle does is that it serializes the object first before writing it to file. Pickling is a way to convert a python object (list, dictionary, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another

python script. The following code converts the input images into the necessary format hence completing the pickling of the dataset-

```
for category,label in dataset.items():
        count=0
        for file in os.listdir('../dataset/'+category):
                count=count+1
                colorImg=Image.open(open('../dataset/'+category+'/'+file,'rb'))
                grayImg=colorImg.convert('L')
                wrapper=[]
                wrapper.append(numpy.array(grayImg))
                wrapper.append(label)
                images.append(numpy.array(wrapper))
                #classes.append(label)
        print(category, ':', count)
shuffle(images)
images2=[]
for im in images :
        classes.append(im[1])
        km=[]
        #im.pop(1)
        km.append(im[0])
        images2.append(km)
images2=numpy.array(images2)
```

```
classes= numpy.array(classes)

print(classes)
```

6.3 LOADING THE DATASET

The python objects, which were generated through the pickling of the dataset have to be loaded onto the convolutional neural network. This is done by the module named Scenes.py. On running this module, the python objects stored for further usage are read into and loaded onto the input layer of the convolutional neural network. The following code shows a function on how the discussed process is carried out-

```
def _print_info(name, data_set):
    x, y = data_set
    print("""{}
  X::
    shape:{}
    min:{} mean:{:5.2f} max:{:5.2f}
  Y::
    shape:{}
    min:{} mean:{:5.2f} max:{}
    """.format(name,
               x.shape, x.min(), x.mean(), x.max(),
               y.shape, y.min(), y.mean(), y.max(),))
def _load_mnist():
    data_dir = os.path.dirname(os.path.abspath(__file__))
```

```
        data_file = os.path.join(data_dir, "scenes4.pkl")
        f = open(data_file,'rb')
        up = pickle.Unpickler(f)

        training_x, training_y, testing_x, testing_y = up.load()
        f.close()
        training_x= training_x.astype('float32')
        training_y=training_y.astype('float32')
        testing_x= testing_x.astype('float32')
        testing_y=testing_y.astype('float32')
        return training_x, training_y, testing_x, testing_y
    training_x, training_y, testing_x, testing_y = _load_mnist()
```

6.4 CONVOLUTIONAL LAYER

This is the layer that distinguishes a convolutional neural network from a regular artificial neural network. It reads in input information in windows sizes and processes then through a filter from. Once the input goes through this stage, the input is ready for the pooling in order to make the input data more abstract than it was when it was delivered to this stage. The convolution layer has been represented by the module named Layer.convlayer. The following code summarizes how the convolution layer has been implemented in this project-

```
    class ConvLayer(Layer):
        def __init__(self, inpt, wts, rand_gen,
                     batch_sz, num_prev_maps, in_sz,
```

```
            num_maps, filter_sz, stride,
            mode='valid',
            actvn='relu50',
            reg=()):
        """

        :param inpt:
        :param wts:
        :param rand_gen:
        :param batch_sz:
        :param num_prev_maps:
        :param in_sz:
        :param num_maps:
        :param filter_sz:
        :param stride:
        :param mode: "valid", "full", "same"
        :param actvn:
        :param reg:
        :raise NotImplementedError:

        TODO: Add support for rectangular input
        """
        assert (wts is not None or rand_gen is not None)
        assert mode in ("valid", "full", "same")
```

```
image_shape = (batch_sz, num_prev_maps, in_sz, in_sz)
filter_shape = (num_maps, num_prev_maps, filter_sz, filter_sz)

# Assign Weights
fan_in = num_prev_maps * filter_sz * filter_sz
fan_out = num_maps * filter_sz * filter_sz
self.W, self.b = init_wb(wts, rand_gen,
                         filter_shape, (filter_shape[0], ),
                         fan_in, fan_out, actvn, 'Conv')

# Add Conv2D Operation to the graph
border_mode = "valid" if mode == "valid" else "full"
conv_out = nnconv.conv2d(inpt, self.W, image_shape, filter_shape,
                         subsample=(stride, stride),
                         border_mode=border_mode)
if mode == 'same':
    assert stride == 1, "For Same mode stride should be 1"
    shift = (filter_sz - 1) // 2
    conv_out = conv_out[:, :, shift:in_sz + shift, shift:in_sz + shift]
    self.out_sz = in_sz

elif mode == "full":
    self.out_sz = in_sz + filter_sz + 1
```

```
elif mode == "valid":
    self.out_sz = in_sz - filter_sz + 1

# TODO: Remove stride support OR make more robust
self.out_sz //= stride
self.output = activation_by_name(actvn)(conv_out +
                                        self.b.dimshuffle('x', 0, 'x', 'x'))

# Store Parameters
self.params = [self.W, self.b]
self.inpt = inpt
self.num_maps = num_maps
self.mode = mode
self.n_out = num_maps * self.out_sz ** 2
self.reg = {"L1": 0, "L2": 0,
            "momentum": .95,
            "rate": 1,
            "maxnorm": 0,}
self.reg.update(reg)

self.args = (batch_sz, num_prev_maps, in_sz, num_maps, filter_sz,
             stride, mode, actvn, reg)
self.representation = (
```

```
            "Conv Maps:{:2d} Filter:{} Stride:{} Mode:{} Output:{:2d} "
            "Act:{}\n\t    L1:{L1}  L2:{L2}  Momentum:{momentum}
Rate:{rate} Max Norm:{maxnorm}"
            "".format(num_maps, filter_sz, stride, mode, self.out_sz,
                actvn, **self.reg))

    def TestVersion(self, inpt):
        return ConvLayer(inpt, (self.W, self.b), None, *self.args)
```

6.5 POOLING LAYER

Pooling is done in order to subsample the data passed to it. When the input data goes through the convolution layer, due to the activation functions, the input gets processed to generate a huge number of feature maps. These feature maps contain too much and too different information which is quiet redundant. In order to reduce the dimensionality of these feature maps, they are passed through the pooling layer, where they are maxed or averaged with surrounding values to an estimate, thus being generalized in the process. Also the dimensions of the feature maps are reduced greatly which further help in faster and easier processing. This layer has been implemented in Layer.poollayer. The following code shows hoe pooling layer has been implemented in this project-

```
class PoolLayer(Layer):
    def __init__(self, inpt, num_maps, in_sz, pool_sz, ignore_border=False):
        """
        Pool Layer to follow Convolutional Layer
        :param inpt:
        :param pool_sz:
```

```
                :param ignore_border: When True, (5,5) input with ds=(2,2)
                   will generate a (2,2) output. (3,3) otherwise.
                """
                self.output = downsample.max_pool_2d(inpt, (pool_sz, pool_sz),
                                                     ignore_border=ignore_border)

                if ignore_border:
                   self.out_sz = in_sz//pool_sz
                else:
                   self.out_sz = math.ceil(in_sz/pool_sz)

                self.params = []
                self.inpt = inpt
                self.num_maps = num_maps
                self.ignore_border = ignore_border
                self.args = (num_maps, in_sz, pool_sz, ignore_border)
                self.n_out = num_maps * self.out_sz ** 2
                self.representation = (
                   "Pool Maps:{:2d} Pool_sz:{} Border:{} Output:{:2d}"
                   "".format(num_maps, pool_sz,
                           "Ignore" if ignore_border else "Keep",
                           self.out_sz))

            def TestVersion(self, inpt):
```

```
return PoolLayer(inpt, *self.args)
```

6.6 FULLY CONNECTED NEURAL NETWORK LAYER

After the input has been made abstract enough for the further processing to be done, the input actually gets classified in the neural network stage, which is the final stage of the implementation of the project. This neural network is different from the previous convolutional neural network in ways that it is fully connected. By this, we can say that a fully connected layer is basically a layer where neurons have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. This fully connected layer has been implemented in the module of NeuralNet.py

6.7 GUI, CLASSIFICATION AND CONFUSION MATRIX

The classification happens in the fully connected layer and it is passed on. For a better user friendly interface, we have designed a Graphical User Interface to display the class of the image. The GUI has been coded using tKinter library. It displays the class of the image sample along with the efficiency of the trained model and the confusion matrix for the considered dataset.

CHAPTER 7

RESULTS AND DISCUSSIONS

CHAPTER 7

RESULTS AND DISCUSSIONS

With the CNN model under discussion, we have performed training and classification on various subsets of the dataset as well as the classes considered.

When the CNN was run for a classification of an image into 8 classes, the classes of coast, forest, highway, inside city, mountain, open country, street, tall building, the model achieved an efficiency of around 54%.

When the CNN was run for a classification of an image into 6 classes, the classes being coast, forest, inside city, mountain, open country, tall building, the model achieved an efficiency of around 57%.

When the CNN was run for a classification of an image into 4 classes, the classes being coast, mountain, forest, tall building, the model achieved an efficiency of around 59%.

The efficiency increases with the decrease in the number of classes as the classes get more generalized. As the real world images include many objects, variations and elements, the classification of images for the machine becomes confusing, For example, if an image contains both the sea and a mountain, a machine might get confused as to what to classify the image as coast or mountain. But when one class, let's say mountain is removed, the image can be classified as coast.

Our results show that a large, deep convolutional neural network is capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning. It is notable that our network's performance degrades if a single convolutional layer is removed. For example, removing any of the middle layers results in a loss of about 2% for the top-1 performance of the network. So the depth really is important for achieving our results. To simplify our experiments, we did not use any unsupervised pre-training even though we expect that it will help, especially if we obtain enough computational power to significantly increase the size of the network without obtaining a corresponding increase in the amount of labeled data. Thus far, our results have improved as we have made our network larger and trained it longer but we still have many orders of magnitude to go in order to match the infero-temporal pathway of the human visual system. Ultimately we would like to use very large and deep

convolutional nets on video sequences where the temporal structure provides very helpful information that is missing or far less obvious in static images.

CHAPTER 8

SNAPSHOTS

CHAPTER 8

SNAPSHOTS

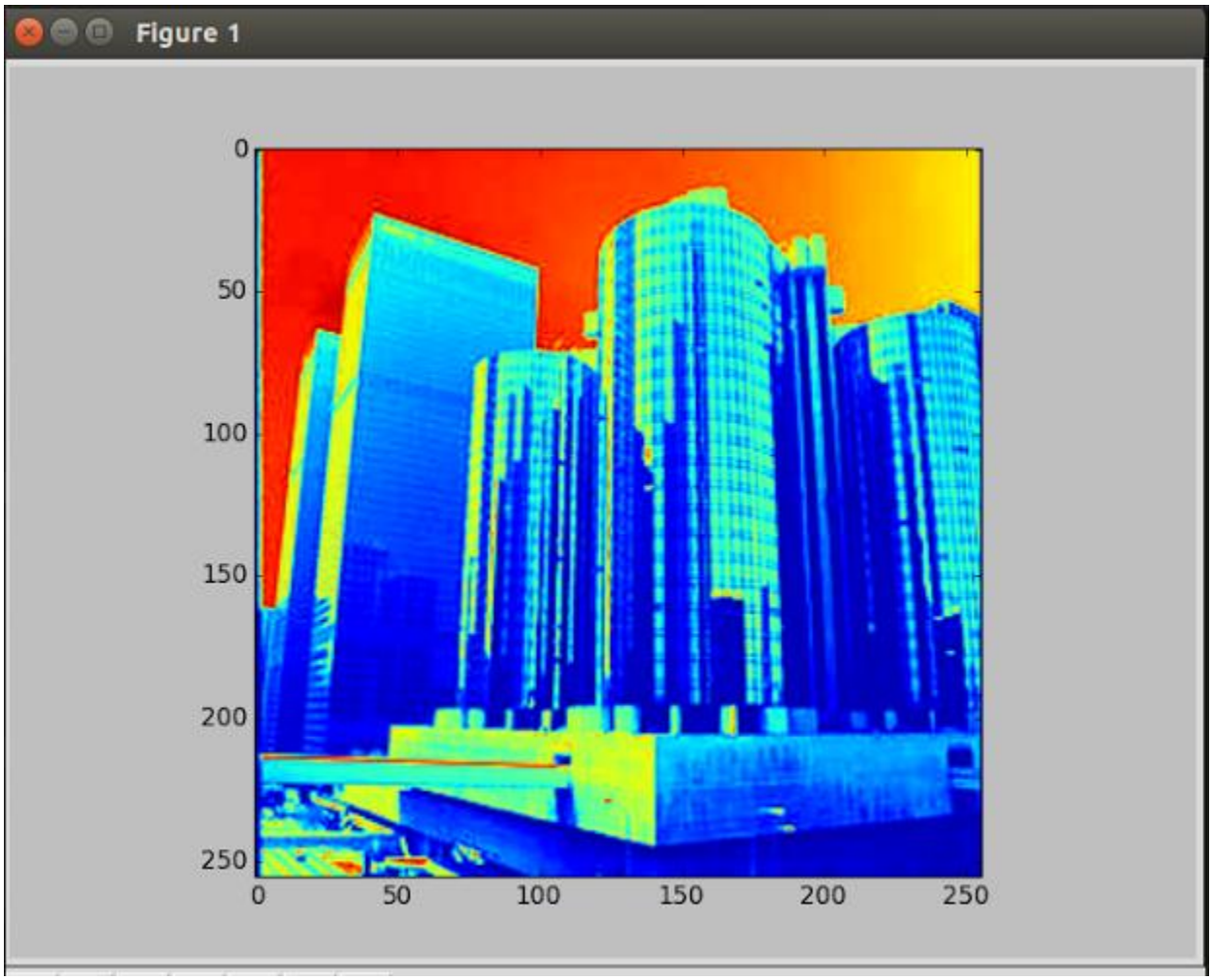

Fig 8.1 Example of image during processing

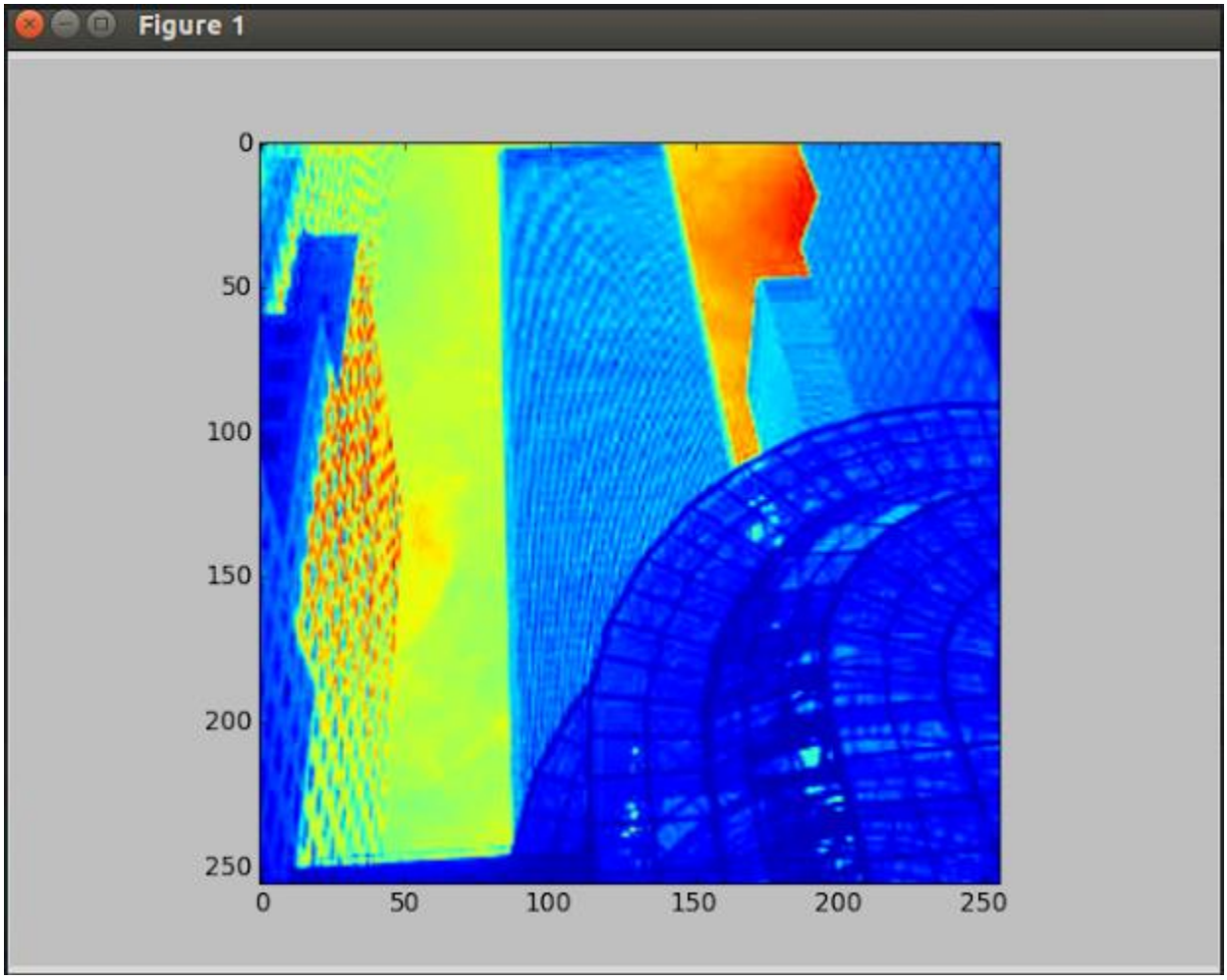

Fig 8.2 Another example of image during processing

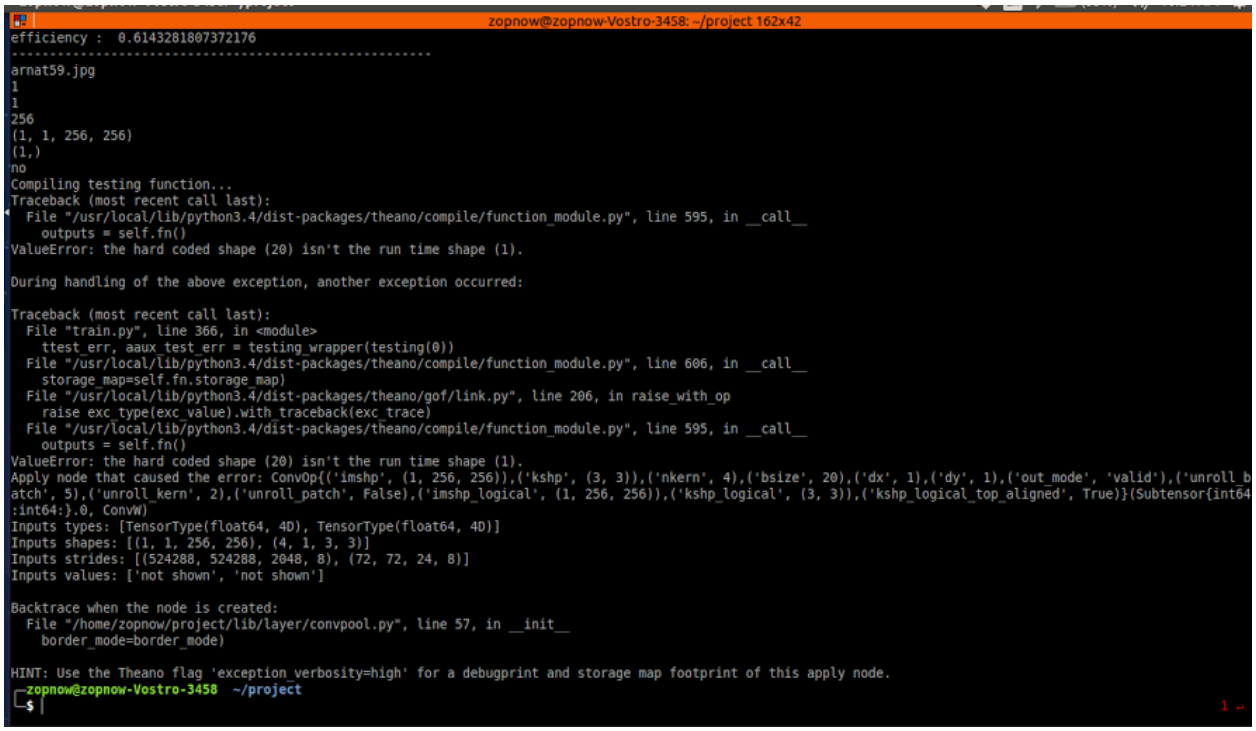

Fig 8.3 results after training

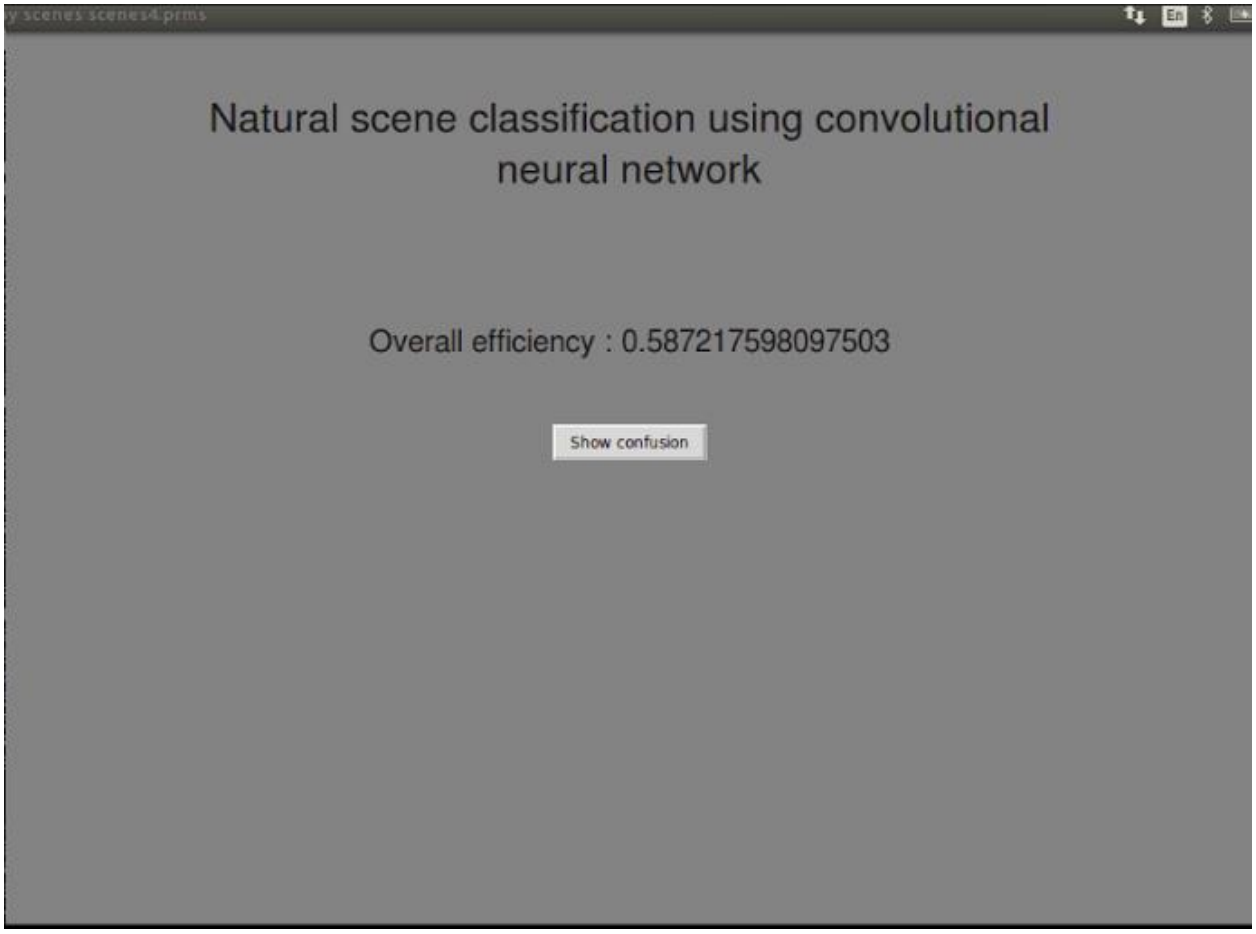

Fig 8.4 The efficiency of the classification model

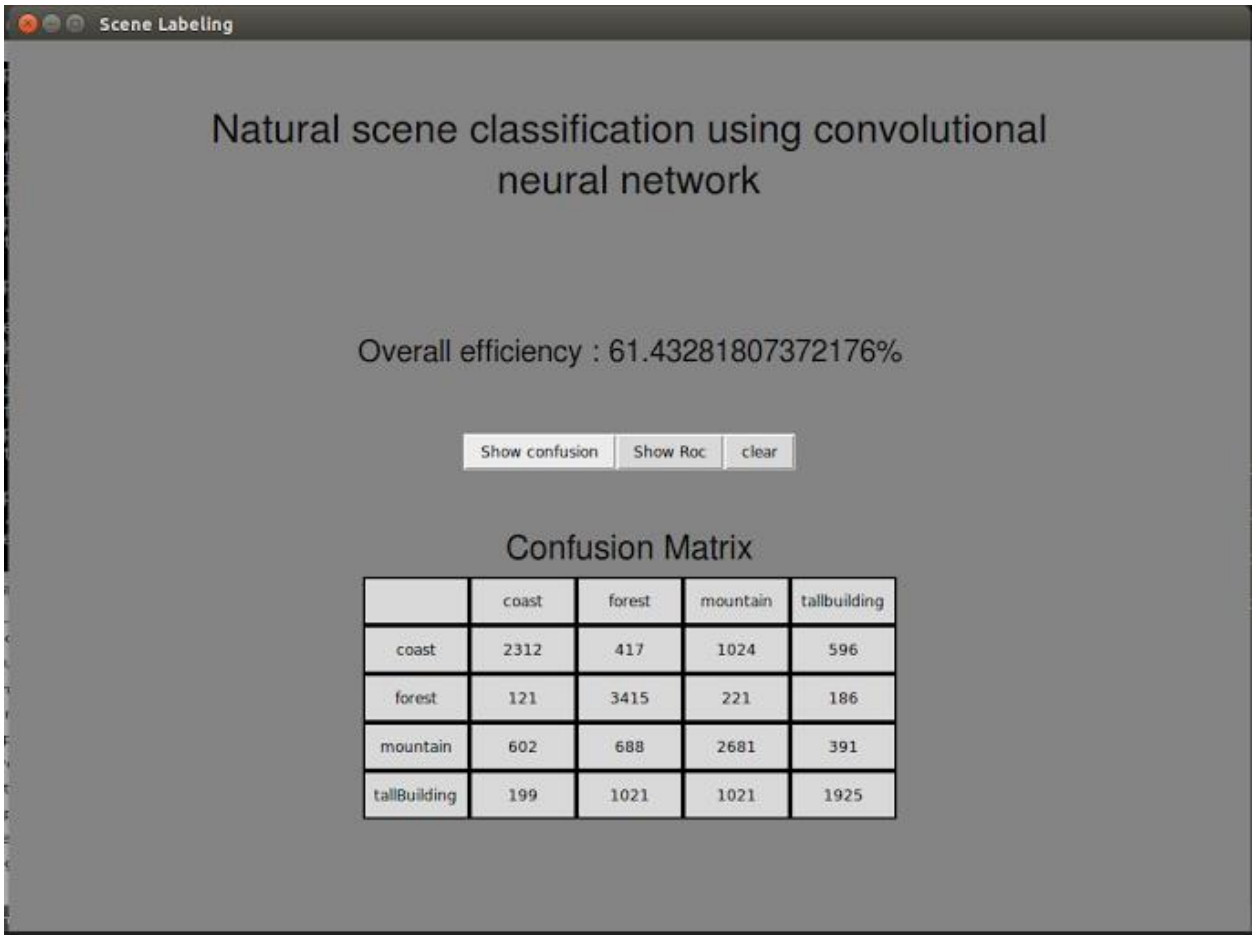

Fig 8.5 Display of confusion matrix after training

CHAPTER 9

CONCLUSION

CHAPTER 9

CONCLUSION

The CNN activation features have a promising outlook towards image classification and object recognition for universal representations. Compared to the many other existing technologies, CNN filters are efficient and activations are fast to extract, even by the application of some special kind of techniques like spatial pyramid structures. At the same time, the important feature of dimensionality reduction have helped the image processing problems with their deep applications. CNN activation features extracted on multiple scales, with the current state-of-the-art performance reported on several datasets, can be considered as an alternative approach for this purpose. On the other hand, combining several CNNs can also result in a performance gain. This has also been observed e.g. in the ImageNet challenges, even when using several networks of identical architecture and trained with the same input data. Even better results can be expected if the variety of the used CNNs would be larger. Overall, Convolutional Neural Networks have proven to be most efficient and well fit tool for image recognition problems as they yield good results compared to other technologies developed till now.

CHAPTER 10

FUTURE ENHANCEMENTS

CHAPTER 10

FUTURE ENHANCEMENTS

This project is a basic working model of Convolutional Neural Networks used for natural Scene Classification. It can be enhanced in the ways mentioned below-

- Trying our CNN model with the aid of a Graphical Processing Unit
- Testing our CNN model with a bigger dataset and comparing results with our existing one
- Including more categories of natural scenes for classification purposes
- Trying to classify images that fall into more than one category
- Try real time scene labelling from a streaming video
- Design a better GUI for better user compatibility
- Testing our method on more datasets to show our generality
- Improving the performance of CNN by constructing/changing layers and parameters per layer
- Training the CNN on better organized dataset
- Training the CNN model better and with possibly higher dimensioned dataset

CHAPTER 11

BIBLIOGRAPHY

CHAPTER 11

BIBLIOGRAPHY

[1] M. Koskela, J. Laaksonen, Convolutional Network Features for Scene Recognition, 2014

[2] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: NIPS, p. 2012.

[3] L.-J. Li and L. Fei-Fei. What, where and who? Classifying events by scene and object recognition. In ICCV, 2007

[4] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, C3d: Generic features for video analysis, arXiv preprint arXiv:1412.0767.

[5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the ACM International Conference on Multimedia, pages 675–678, 2014.

[6] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In CVPR, pages 3367–3375, 2015.

[7] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, C. Bray, Visual categorization with bags of keypoints, in: In Workshop on Statistical Learning in Computer Vision, ECCV, 2004, pp. 1–22.

[8] K. Grauman, T. Darrell, The pyramid match kernel: Discriminative classification with sets of image features, in: IEEE ICCV, Vol. 2, 2005, pp. 1458–1465.

[9] H. J´ egou, M. Douze, C. Schmid, P. P´ erez, Aggregating local descriptors into a compact image representation, in: IEEE CVPR, 2010, pp. 3304–3311.URL "http://lear.inrialpes.fr/pubs/2010/JDSP10"

[10] F. Perronnin, C. Dance, Fisher kernels on visual vocabularies for image categorization, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, 2007, pp. 1–8.

[11] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, Locality-constrained linear coding for image classification, in: IEEE CVPR, 2010, pp. 3360–3367.

[12] C. Schmid, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: IEEE CVPR, 2006.22

[13] Y. Gong, L. Wang, R. Guo, S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, arXiv preprint arXiv:1403.1840.

[14] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, Overfeat: Integrated recognition, localization and detection using convolutional networks, CoRR abs/1312.6229. URL http://arxiv.org/abs/1312.6229

[15] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Computer Vision–ECCV 2014, Springer, 2014, pp. 818–833.

[16] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: A deep convolutional activation feature for generic visual recognition, arXiv preprint arXiv:1310.1531.

[17] A. S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, Cnn features off-the-shelf: an astounding baseline for recognition, arXiv preprint arXiv:1403.6382. 2013, pp. 803–810.

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, 2014.

[19] A. Sharma, O. Tuzel, and M.-Y. Liu. Recursive context propagation network for semantic scene labeling. In NIPS, pages 2447–2455. 2014.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[21] K. G. Derpanis, M. Lecce, K. Daniilidis, R. P. Wildes, Dynamic scene understanding: The role of orientation features in space and time in scene classification, in: IEEE CVPR, 2012, pp. 1306–1313.

[22] C. Feichtenhofer, A. Pinz, R. P. Wildes, Bags of spacetime energies for dynamic scene recognition, in: IEEE, 2014.

[23] Xu, Zhongwen, Yi Yang, and Alexander G. Hauptmann. A Discriminative CNN Video Representation for Event Detection. arXiv preprint arXiv:1411.4006 (2014).