DEVELOPMENT OF A DEEP CONVOLUTIONAL NEURAL NETWORK BASED SYSTEM FOR OBJECT RECOGNITION IN VISIBLE LIGHT AND INFRARED IMAGES

BY

YUSUF IBRAHIM

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY,

ZARIA

JANUARY, 2017

DEVELOPMENT OF A DEEP CONVOLUTIONAL NEURAL NETWORK BASED SYSTEM FOR OBJECT RECOGNITION IN VISIBLE LIGHT AND INFRARED IMAGES

BY

Yusuf IBRAHIM, B.Eng (ABU) 2011

MSC/ENG/28389/2012-2013

yusuf2007ee@gmail.com

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE (MSc) DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY,

ZARIA, NIGERIA

JANUARY, 2017

DECLARATION

I declare that the work in this dissertation entitled "DEVELOPMENT OF A DEEP CONVOLUTIONAL NEURAL NETWORK BASED SYSTEM FOR OBJECT RECOGNITION IN VISIBLE LIGHT AND INFRARED IMAGES**"** has been carried out by me in the Department of Electrical and Computer Engineering. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other institution.

Yusuf IBRAHIM _____ _____

Signature Date

CERTIFICATION

This dissertation entitled "DEVELOPMENT OF A DEEP CONVOLUTIONAL NEURAL NETWORK BASED SYSTEM FOR OBJECT RECOGNITION IN VISIBLE LIGHT AND INFRARED IMAGES" by Yusuf IBRAHIM meets the regulations governing the award of degree of Master of Science (M.Sc.) in Computer Engineering of the Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Chairman, Supervisory Committee	_____	_____
(Prof. M.B. Mu'azu)	Signature	Date
Member, Supervisory Committee	_____	_____
(Dr. S.M. Sani)	Signature	Date
Head of Department	_____	_____
(Dr. Y. Jibril)	Signature	Date
Dean, School of Postgraduate Studies	_____	_____
(Prof. K. Bala)	Signature	Date

DEDICATION

This dissertation is dedicated to my beloved mother

ACKNOWLEDGEMENT

All praise to God the Almighty Who gave me the opportunity to successfully complete this MSc program.

I would like to express my most sincere appreciation to my Supervisors in the persons of Prof. M.B. Mua'zu and Dr. S.M. Sani for their immense contributions, guidance, and time throughout this research. May the Almighty God reward them with AljannahFirdaus.

I also thank and appreciate the head of department, Dr. Y. Jibrilfor his encouragement and motivation as well as all members of staff namely: Prof. B.G. Bajoga, Prof. B. Jimoh, Dr. S. Garba, Dr. K. A. Abubilal, Dr. A. M. S. Tekanyi, Dr. A. D. Usman, Dr. T. H. Sikiru, Dr. E. A. Adedokun, Engr. Z. M. Abdullahi, Engr. A. Mohammed, Engr. S. T. Ajayi, Engr. A. A. Saidu, Engr. G. A. Olarinoye, Engr. P. U. Okorie, Engr.B. O. Sadiq, Engr. B. Yahaya, Engr. E. Obi, Engr. Z. M. Abubakar, and all others who contributed in one way or the other to the successful completion of this research work.

I would also like to thank my family for their support, motivation, and patience as well as my friends and colleagueswho made my stay in the university a valuable and memorable experience.

ABSTRACT

This research investigated image recognition frameworks on datasets of visible-light and infrared (IR) imagery using deep convolutional neural networks (CNN). This is due to their recent success on a variety of problems including computer vision which often surpassed the state of the art methods. Three deep learning based object recognition approaches were investigated on a fused version of the images in order to exploit the synergistic integration of the information obtained from varying spectra of same data with a view to improving the overall classification accuracy. Firstly, a simple 3-layer experimental deep network was designed and used to train the datasets for performing recognition. A second experiment was conducted where a pre-trained 16-layer convolutional neural network (Imagenet-vgg-verydeep-16) was used to extract features from the datasets. These features are then used to train a logistic regression classifier for performing the recognition. Finally, an experiment was conducted where another pre-trained model (Imagenet-vgg-f) was fine-tuned to suit the dataset's classes and which is then retrained accordingly on the datasets using back propagation. This research adopted a simple and novel fusion strategy where the IR and visible images were fused by concatenating the IR image as an additional fourth layer to the visible image with a view to enhancing the performance of object recognition systems. Despite its simplicity and the limited size of the training data, the 3-layer network achieved a classification accuracies of 86.8590% on the fused multimodal image, 85.0610% on the visible images and 67.9878% on the Infrared images. These results represent a respective improvement by 4.76%, 3.16% and 13.99% when compared withthose of the CNN architectureofZhang. Also, an improvement by 4.06% on visible images was obtained over that of Zhang's*Gnostic Field + CNN* model. It is also observed that while the pre-trained model performed well, fine-tuning improved the performance to 100% classification accuracy. Results obtained were compared with those ofZhangas a means of validation. This work was implemented using MATLAB programming language, MatConvnet library for CNN and Liblinear library for large linear classification.

TABLE OF CONTENTS

COVER PAGE ..... **i**

TITLE PAGE ..... **ii**

DECLARATION ..... **iii**

CERTIFICATION ..... **iv**

DEDICATION ..... **v**

ACKNOWLEDGEMENT ..... **vi**

ABSTRACT ..... **vii**

TABLE OF CONTENTS **viii**

LIST OF APPENDICES **xi**

LIST OF FIGURES ..... **xii**

LIST OF TABLES ..... **xiii**

LIST OF ABBREVIATIONS ..... **xiv**

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND 1

1.1.1 Traditional Recognition Approaches 1

1.1.2 Multimodal Images 2

1.1.3 Fusion of IR and Visible Imagery 4

1.2 STATEMENT OF RESEARCH PROBLEM 5

1.3 MOTIVATION .6

1.4 AIM AND OBJECTIVES 7

1.5 METHODOLOGY 8

1.6 DISSERTATION ORGANIZATION 9

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION10

2.2 REVIEW OF FUNDAMENTAL CONCEPTS10

2.2.1 Machine Learning10

*2.2.1.1Machine learning categories*10

2.2.2 ComputerVision.....12

2.2.2.1Object recognition 12

2.2.3 Artificial Neural Networks 13

2.2.3.1 Structure of a biological neuron 14

*2.2.3.2 Neuron network topology*16

2.2.3.3 Activation function 16

2.2.3.4 Multi-layer Perceptron 19

2.2.4 Training Algorithm: The Back Propagation Algorithm20

2.2.5 Deep learning22

*2.2.5.1 Deep neural networks (DNN)*23

*2.2.5.2 Convolutional neural networks*24

*2.2.5.3 Advantages of using deep CNN*26

2.3 REVIEW OF SIMILAR WORKS 27

CHAPTER 3 METHODOLOGY

3.1 INTRODUCTION 31

3.2 DATA COLLECTION 31

3.2.1 VAIS Dataset 31

3.2.2 RGB-NIR Scene Dataset 32

3.3 DESIGNING A SIMPLE EXPERIMENTAL NETWORK 33

3.4USING PRE-TRAINED NETWORK FOR CLASSIFICATION 34

3.4.1Extracting Features Using CNN Pre-trained on Imagenet VGGNE 35

3.4.2 VGGNet 36

3.4.3 Finetuning the Pretrained CNN with Backpropagation 37

3.3.4 Alexnet/Imagenet 37

3.5 PERFORMANCE METRIC 39

CHAPTER FOUR: RESEARCH METHODOLOGY

4.1 INTRODUCTION 40

4.2 PRELIMINARY RESULTS 40

4.3 MODIFYING THE CNNs TO ACCEPT 4-LAYERIMAGES 40

4.3.1 Results for Modified 3-layer CNN 41

4.3.2 Results for Modified Pre-trained CNN 43

4.4 EXTENDING EXPERIMENTS ON RGB-NIR SCENE DATASET.....44

4.4.1 Results for Training from Scratch Using RGB-NIR Scene Dataset44

4.4.2 Results for Fine-tuning Using RGB-NIR Scene Dataset.....45

4.5 VALIDATION.....46

CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

5.1 SUMMARY.....48

5.2 SIGNIFICANT CONTRIBUTIONS 48

5.3 CONCLUSION.....49

5.4 LIMITATIONS.....49

5.5 RECOMMENDATIONS FOR FURTHER WORK.....50

REFERENCES 51

APPENDICES 56

LIST OF APPENDICES

APPENDIX A: Implementation of BP Using Sigmoid 56

APPENDIX B1: Matlab Codes: Main M- File 61

APPENDIX B2: Training from Scratch a 3- Layer Network .62

APPENDIX B3: Fine-tuning a Pretrained Model 63

APPENDIX B4: Testing the Trained Model 64

APPENDIX B5: Feature Extraction + Classifier (Logistic Regression) 64

APPENDIX B6: Set Image Dataset 65

APPENDIX B7: Code that Implements SGD for training CNN 67

LIST OF FIGURES

Figure 1.1: A Block Diagram of Deep Learning Architecture 2

Figure 2.1: A Taxonomy of Artificial Neural Network Architectures 14

Figure 2.2: Structure of a Typical Neuron 15

Figure 2.3: Graph of a Sigmoid Function 16

Figure 2.4: A Simple Perceptron with Sigmoid Function.....19

Figure 2.5: Architecture of a Multilayer Perceptron.....20

Figure 2.6: Structure of a Typical Neural Network 24

Figure 2.7: A Typical CNN Architecture 25

Figure 2.8: A Rectified Linear Unit.....26

Figure 3.1: The Experimental Three-Layer CNN.....33

Figure 3.2: The structure of VGGNet 36

Figure 3.3: The structure of AlexNet.....37

Figure 4.1: Visible VAIS Result.....42

Figure 4.2: IR VAIS Results.....42

Figure 4.3: Paired VAIS Results.....42

Figure 4.4: Results for Finetuning a Pretrained Model.....44

Figure 4.5: Response for Training IR (left) and Visible (right).....44

Figure 4.6: Response for Training Paired RGB-NIR Images 45

Figure 4.7: Results for Finetuning Pretrained Model on the RGB-NIR Scene Dataset .46

LIST OF TABLES

Table 3.1: Summary of models' performance on ILSVRC 2012 validation data 38

Table 4.1: First Results for VAIS Datasets.....41

Table 4.2: Second Results for VAIS Datasets 42

Table 4.3: Validation Results.....47

LIST OF SYSMBOLS AND ABBREVIATIONS

Acronym	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Network
BM	Boltzmann machine
BP	Back Propagation
CNN	Convolutional Neural Networks
CNS	Central Nervous System
DBN	Deep Belief Network
DNN	Deep Neural Network
GPU	Graphical Processing Units
HoG	Histogram of Oriented Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IR	Infrared
IVRL	Images and Visual Representation Laboratory
LBP	Local Binary Patterns
LR	Logistic Regression
MLP	Multi-Layer Perceptron
MSER	Maximally Stable Extremal Regions
NIR	Near-Infrared
NN	Neural Network
OCR	Optical Character Recognition

PDP	Parallel Distributed Processing
PNS	Peripheral Nervous System
RBM	Restricted Boltzmann machine
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue
RNN	Recurrent Neural Networks
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-up Robust Features
SVM	Support Vector Machine
VAIS	Visible and Infrared Spectrums
VGG	Visual Geometry Group

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

In the field of computer vision, the area of object recognition which deals with the task of identifying or finding objects in images or video sequence has over the past decades attracted tremendous yet inexhaustible research interests. This is due to its increasing usefulness in several daily life applications such as law enforcement, surveillance systems, information safety, computer security, secured financial transactions amongst others (Kong *et al.*, 2007). Humans naturally recognize multitude of objects with almost no efforts in images. This is regardless of the images' varying viewpoints, sizes, scales or even when the images are rotated or translated. More interestingly is the fact that humans recognize objects even when they are obstructed partially from view. Performing these tasks by a computer still remains a challenging task in computer vision. Object recognition system has wide range of applications in areas related to surveillance, safety and security, access control, information security, identity fraud, etc. These include optical character recognition (OCR), android eyes-object recognition, image panoramas, image watermarking, global robot localization, face detection, manufacturing quality control, content-based image indexing, object counting and monitoring, visual positioning and tracking, automated vehicle parking systems, video stabilization. Hence, researchers over the decades have developed several techniques in order to solve this problem (Chang *et al.*, 2008).

1.1.1 Traditional Recognition Approaches

Over the past decades, many approaches have been implemented to perform object recognition. In particular, the traditional recognition approaches usually use multitude of fixed features such as edges, scale-invariant feature transform (SIFT), histogram of oriented gradients (HoG) etc.

that are usually hand designed or engineered (i.e. fixed kernels) used to extract features from an image which are later fed to a simple trainable classifier (such as an SVM). Today, rather than relying on these hand engineered features, recent research in end-end feature learning makes the predictions using sequences of non-linear processing stages to learn representations of data with multiple level of abstraction so that the resulting intermediate representations are interpreted as being feature hierarchies with the whole system being jointly learned from data. This method is called deep learning, which has proven to surpass the traditional approaches by far. Deep learning has revolutionized the general fields of pattern recognition and machine learning and basically exploits the idea of extracting features automatically and hierarchically (LeCun *et al.*, 2015; Schmidhuber, 2015), this is depicted in Figure 1.1.

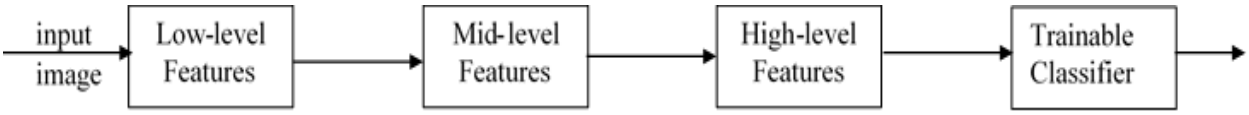


Figure 1.1: A Block Diagram of Deep Learning Architecture

In particular a specialized type of network called convolutional neural network has proven to show a massive power as a high end-to-end learning architecture(Krizhevsky *et al.*, 2012). It is actually considered amongst the most powerful and classic supervised methods widely used today in pattern recognition and machine learning especially in the aspect of object recognition. It is well accepted in the scientific community and have already been deployed in renowned and gigantic corporations such as Facebook and Google in order to solve image auto-tagging (Rosenberg, 2013) and face recognition problems (Taigman *et al.*, 2014).

1.1.2 Multimodal Images

Multimodal images are important since visible images are sensitive to varying illumination conditions which makes the performance of systems built on these images to be degraded especially under poor illumination conditions (Chang *et al.*, 2008). To ameliorate this problem,

the data is complemented by an infrared (IR) spectrum version of the image. This is possible since IR images have little control over lighting conditions as the images are formed from emission that is intrinsic to the objects. Also, despite the massive performance shown by these state-of-the-art techniques, several other factors affect the performance of the object recognition system especially when visible images are involved including the illumination changes, pose variations, occlusions etc. Fortunately, IR sensors unlike visible sensors measure an object's energy radiations which are less sensitive to the illumination changes and are in fact operable in the dark (Zhang *et al.*, 2015). IR images fundamentally provide different view of the scene since illumination or visually reflective information does not generally affect IR measurements and that the structures, shapes and the intensities seen in IR images are formed by a different phenomenon. For example, infrared images also contain gradients and textures which are based on the complicated heat transfer that exist between the internal and the external heat sources and the physical object rather than that of visible illumination. Interestingly, it is possible to correlate features in both the visible and IR spectra since the structures and the composition of the objects captured can easily be recognized in both the visible and IR (Hariharan *et al.*, 2006; Ma *et al.*, 2015;Zhang *et al.*, 2015). However, despite the robustness of the IR images to illumination changes, they also present new challenges which in general allow for the relaxation of many restrictions usually placed in the visible spectrum algorithm which primarily is due to the issues of incident illumination on an object. The IR images also do not provide data with high-resolution like the visible image. An example is face detection in low-light environments where instead of attempting to adjust the detection algorithm due to poor illumination, reliance is shifted on the IR imagery which is guaranteed to consistently perform in poor or nominal illumination (Chang *et al.*, 2008).

1.1.3 Fusion of Infrared and Visible Imagery

In order to exploit the synergistic integration of the information obtained from varying spectrum of same data with a view to improving the overall classification accuracy, fusion approaches are proposed in literature with the expectation that a combination of these data from multiple sources would yield a more informative representation which guarantees better recognition accuracy than the original(Chang *et al.*, 2008; Kong *et al.*, 2007).

Generally, the Information obtained from sensors contains redundancy and may not contribute to actual knowledge of the scene to be analyzed. This information, obtained from different means could provide complementary details of the analyzed object. Therefore, the goal of information fusion approaches is to ultimately exploit these complimentary portions and therefore developing an algorithm capable of visualizing these information is of great importance (Singh *et al.*, 2004). Several literatures reviewed techniques for image fusion for the purpose of visualization and the data fusion from visible and IR sensors has really led to an improved target recognition rates (Zhou & Hecht-Nielsen, 1993). The choice of an appropriate fusion scheme depends on both the application and data. In this research context, the visible cameras capture the reflectance of light of the object while the infrared cameras measure the thermal or heat energy emitted by the object. Now, since the surface of the object and its temperature has nothing in common, information contained in the visible as well as the IR images are hence complimentary and independent.

Pixel by pixel fusion does not generally preserve spatial information in the image unlike multiple-resolution level fusion which allows features with varying spatial extend to be fused at most salient resolutions. Hence, important features which appear at lower resolutions could be preserved by the fusion process. Basically, in multi-resolution fusion, a multiscale transform is

performed on each source image after which a composite multi-scale representation is constructed based on some specific fusion rules. The fused version of the image is obtained by taking inverse of the multiscale transform (Piella, 2003). Some multiscale techniques which are popular include the Fourier transform, Laplacian pyramid, and wavelets. High frequency components are relatively independent of the global illumination changes while low frequencies take the spatial relationships which exist among the pixels into account and are less sensitive to small changes and noise.

The fusion schemes or strategy studied in this work adopts a simple idea of concatenating the infrared image which is a single channel image as a fourth layer to the visible image which is a three channel RGB image because each of the sensing modality has its own benefits and drawbacks as neither of the modality alone can solve completely the problem of robust object recognition alone. This fusion process will therefore attempt to maximize the advantage of each of the modality by fusing intelligently their individual information and at the same time overcoming limitations of each modality when used alone(Kong *et al.*, 2007; Ma *et al.*, 2015).

1.2 STATEMENT OF RESEARCH PROBLEM

Images acquired using visible sensors are sensitive to some factors such as varying illumination conditions, scattering, absorption, etc. which degrade the performance of object recognition systems built on these images especially under poor illumination conditions (Chang *et al.*, 2008). Interestingly, lighting conditions have little or no effect over IR sensors as the images are formed from emissions that are intrinsic to the objects and are in fact operable in the dark (Zhang *et al.*, 2015). Hence, IR images fundamentally provide different view of the scene and the structures, shapes and the intensities seen in IR images are formed by a different phenomenon. According to literatures, (Hariharan *et al.*, 2006; Ma *et al.*, 2015;Zhang *et al.*, 2015), it is possible to correlate

features in both the visible and IR spectra since the structures and the composition of the objects captured can easily be recognized in both the visible and IR. This research therefore exploited the capabilities of deep convolutional neural networks (CNN) architectures as an end-to-end learning architecture in order to recognize objects in multimodal images by synergistically integrating the information obtained in the visible and infrared spectrums of same data with a view to performing a critical role in perception systems and substituting the standard computer vision approaches.

1.3 MOTIVATION

Computer vision applications are getting more important by the day and the technology is far from being matured. For example, our roads, cities and general environment have very unpredictable dynamics where multiple actors such as animals, pedestrians, vehicles, street furniture etc. coexist together. Hence, providing autonomous vehicle having a robust perception system is needed in order to understand the environment correctly to be able to interpret what is happening around the surrounding and consequently act upon it. These applications usually get information from their environment by means of on-board sensors. Today, affordable sensors such as multiple layer laser-scanners or depth and appearance capturing cameras are available in the market and are being incorporated in mobile robots and vehicles. Consequently, new information sources are available to be used in the perception systems. As regards the development of the software, significant advances have been made in the field of machine learning, computer vision and mobile robotics research in recent years. The area of deep learning, specifically advances in convolutional neural networks (CNNs) have shown promising results. This is after the breakthrough research by Krizhevsky *et al.*, (2012) on the ImageNet 2012 competition classification benchmark which brings the state-of the-art error from 26.1% to

15.3% (Krizhevsky *et al.*, 2012). Since then, a lot of the winning performances in the competition are obtained by expansions and modification of the so called AlexNets. In fact, ImageNet 2012 benchmark error rate has been reduced to 6.5% by using an even deeper network (Szegedy *et al.*, 2015). Today CNN based methods are surpassing state of the art solutions on a variety of problems such as document recognition, object classification and detection or semantic segmentation and understanding even beating the human capabilities(He *et al.*, 2015). Therefore, CNNs are of special interest either being used for robust features extraction replacing the traditional hand-crafted ones, or as an end-to-end trainable system. This context is the motivation in carrying out this research where the capabilities of deep CNN architectures are employed to recognize objects in multimodal images with a view to performing a critical role in perception systems and substituting the standard computer vision approaches.

1.4 AIM AND OBJECTIVES

The aim of this research is to perform object recognition in visible-light & infrared (IR) images using deep convolutional neural networks.

In order to achieve this aim, the following objectives were employed:

i) To perform a construction of a representative dataset (adopted)

ii) To design and train a simple experimental deep convolutional neural network architecture and investigate its performance on visible, IR and their fused multimodal versions.

iii) To train a logistic regression classifier using features extracted from standard pre-trained 16-layer CNN called imagenet-vgg-verydeep-16 on visible and IR images.

iv) To fine-tune a standard pre-trained model called imagenet-vgg and retraining accordingly based on the dataset's classes using back propagation algorithm on visible, IR and their fused multimodal versions.

v) To compare the performance of the proposed frameworks with that of Zhang *et al* (2015) on the VAIS dataset as a means of validation using classification accuracy as the performance metric.

1.5 METHODOLOGY

In order to achieve the objectives highlighted in section 1.3, the following research methodology was adopted:

i) Data collection of comprehensively labeled images of predefined categories of visible and infrared images. Two datasets were used for this research which are the VAIS datasets (Zhang *et al.*, 2015) for sea vessel classification which consists of 1623 visible images, 1242 IR images and 1088 visible/IR paired images as well as the RGB-NIR Scene Datasets.

ii) Design of a simple experimental deep CNN architecture for investigating the different recognition frameworks proposed. It had a total of 10 layers having 3 convolutional layers, 2 maxpool layers, 2 dropout layers, 2 ReLU layers and a final loss layer.

iii) Training a network with the datasets using the designed architecture for performing recognition tests. The back propagation algorithm is used for training the designed network.

iv) Training a logistic regression classifier using features extracted from this dataset with the standard pre-trained 16-layer CNN (imagenet-vgg-verydeep-16).

v) Fine-tuning a standard pre-trained model (imagenet-vgg) and retraining accordingly based on the dataset's classes using back propagation algorithm.

vi) Performing steps iii) and v) on the fused combination of visible and IR imagery and comparing the results with those on the VAIS dataset using the % classification accuracy as the metric.

vii) Evaluating performance of the proposed frameworks with those obtained by Zhang *et al* (2015) on the VAIS dataset using the % classification accuracy as the metric

1.6 DISSERTATION ORGANIZATION

The general introduction has been presented in chapter one. The rest of the chapters are structured as follows: Firstly, a detailed review of related literature and relevant fundamental concepts about machine learning and its categories, computer vision, object recognition, artificial neural networks and deep learning are carried out in chapter two. Secondly, chapter three describes an in-depth explanation on designing a simple 3- layer CNN. How a pre-trained model can be used to extract features and eventually used for classification, is also described. The chapter also described how to fine-tune the pre-trained CNN and retrained using backpropagation. The statistics of the dataset used as well as the models and frameworks used in the research are also described in chapter three. Thirdly, the analysis, performance and discussion of the result are shown in chapter Four. Finally, conclusion and recommendations for further work makes up the chapter five. The list of cited references and MATLAB codes are provided in the appendices at the end of this dissertation.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter consists of two parts, review of fundamental concepts which discusses the relevant theoretical background necessary for the proper understanding of this work, and the review of similar works.

2.2 REVIEW OF FUNDAMENTAL CONCEPTS

This part discusses related concepts and theories necessary for proper understanding of this work.

2.2.1 Machine Learning

Machine learning is a technical field that grew out of Artificial intelligence (AI) which deals with the process of giving computers the ability to learn from data without actually being programmed explicitly. In other words, it deals with the process of building computers that automatically improve through experience (Michalski *et al.*, 2013). Hence, systems developed by machine learning algorithms operate by building models from examples so as to make predictions that are data-driven as opposed to writing programs that follow some strict static program instructions. Machine learning has strong ties with mathematical/engineering

optimization that delivers methods and application domains to the field. It is employed widely in computing tasks where explicit programming is not feasible or difficult (Michalski *et al.*, 2013).

2.2.1.1 Machine learning categories

Machine learning is classified into three broad categories depending on learning signal available to the learning system. These are discussed in the following sections (Jordan & Mitchell, 2015; Michalski *et al.*, 2013):

A) **Supervised Learning:** where the computer learns from supervised data. That is, the computer is presented with input examples and their corresponding desired outputs for a portion of the data and the objective is to learn a general rule mapping the inputs to the outputs.

B) **Unsupervised Learning:** where input data with no labels are given to the learning algorithm and the goal is to find structure to the input data. Categorization of unsupervised learning also includes:

i. Clustering where sets of unlabeled data are to be divided into clusters or groups

ii. Density estimation which seeks to find a distribution for the inputs in some space

iii. Dimensionality reductionwhere given input data are simplified and mapped to a lower-dimensional space representation.

C) **Reinforcement Learning:** where the computer program has to interact with the environments dynamically and in which it must perform certain goals without having a teacher explicitly telling it whether it has come close to its goal.

This study falls under the supervised learning which could either be of a classification or regression type. It is a classification type if the inputs are divided into two or more classes (binary or multiclass), and the learning algorithm produces a model which assigns new unseen

inputs to one or more classes. An example of this is the spam filtering where the inputs are messages or email classified as either "spam" or "not spam". On the other hand, the problem is a regression type if the outputs are not discrete but continuous. This thesis falls under the multiclass classification type problem where the aim is to train a model that predicts or recognizes with some acceptable level of accuracy the class of an input image.

2.2.2 Computer Vision

Computer vision is the science by which computers or other machines are endowed with vision i.e. the ability to see. It is a field which deals with methods of image acquisition, processing, analysis and understanding. The interpretation of these images may ultimately lead to other decisions or actions, for example say the navigation of autonomous robots. The discipline aims to elaborate on these images to produce symbolic or numeric information in the form of decisions (Morris, 2004). A fundamental idea which stood behind this field is to duplicate the natural human visual abilities through electronic perception and understanding of images (Sonka *et al.*, 2014). This image understanding can be viewed as disentangling of relevant symbolic information from input image data by the use of models developed with the aid of physics, geometry, statistics, as well as learning theory (Forsyth & Ponce, 2003). Computer vision can hence be seen as the science of integrating and automating a wide range of representations and processes for vision perception. Therefore, as a scientific discipline, computer vision mainly concerns the theory behind the process of extracting information from images through artificial systems. The image data can be in several forms such as a multidimensional data from medical scanners, image sequences, views from multiple cameras, etc. From technological point of view, it is the application of those models and theories to construct computer vision systems. There are

many components to computer vision which include object recognition, video tracking, scene understanding, event detection, motion estimation, object pose estimation, and image restoration.

2.2.2.1 Object recognition

Object recognition is a subfield of computer vision concerned with the process of finding and identifying specific objects in digital images or video sequences. It is considered as the most fundamental and central problem in computer vision (Chen *et al.*, 2010). Humans are able to recognize variety of objects effortlessly despite significant variations of the objects in the images due to for example background clutter, different lighting conditions, viewpoints, many different scales and sizes or different poses. In fact objects can be recognized easily by humans even when the view is obstructed partially (Chen *et al.*, 2010). This task however still remains a challenge for computer vision systems. Over the decades, a variety of approaches and models to achieve this task have been implemented in literatures. Some of these models include: extracted features and boosted learning algorithms, Bag-of-words with features such as maximally stable extremal regions (MSER) and speeded up robust features (SURF), template matching, gradient-based and derivative-based matching approaches, image segmentation and blob analysis amongst others. In this thesis, the task is confronted in-depth using deep learning based algorithm called convolutional neural networks.

2.2.3 Artificial Neural Networks

Artificial neural network (ANN) also called neural network (NN or neural net) is a branch of artificial intelligence which is a statistical learning model belonging to a family of biologically inspired neural network (Russell *et al.*, 1995). ANNs are computational models which are very useful in estimating or approximating functions that depend on large number of inputs data

which are unknown generally. They are presented generally as black box containing layers of "neurons" interconnected in a manner to relay messages to each other. Each of its connection has some numeric weight which are tuned and assigned based on experience making the neural network adaptive to those inputs and capable of learning (Sharma *et al.*, 2012). When modelling these artificial neurons, the biological neuron's complexity is highly abstracted to include components like the inputs (synapses), which get multiplied by the weights or respective signal strength, then a mathematical function computing or determining the activation of the neuron. Also, a function which may be the identity finally computes neuron's output. Hence, ANNs combine these "neurons" to process information. Artificial neural networks have been applied to solve a variety of problem in speech recognition and computer vision which have been considered difficult to solve using the conventional rule based programming approaches. A typical taxonomy of neural network is shown in Figure 2.1.

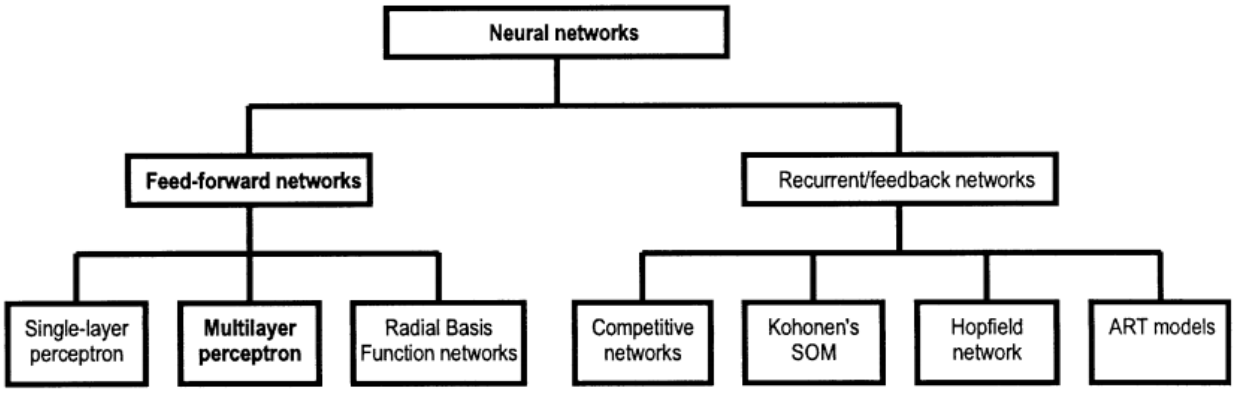


Figure 2.1: A Taxonomy of Artificial Neural Network Architectures (Gardner & Dorling, 1998)

A typical example of applying neural network is in the area of handwriting recognition where set of input neurons are activated by pixels of an input image. The neurons are then activated and passed to other neurons after being weighted and transformed using a predefined activation

function. This process is continuously repeated until the final output of the neural network is activated which determines the written symbol that has been read.

2.2.3.1 Structure of a biological neuron

A neuron is an electrically excitable nerve cell that process and transmits information through the body via chemical and electrical signals (Kandel *et al.*, 2000). They constitute the basic and fundamental information-processing and building block of the nervous system responsible for human cognition. The signals that exist between the neurons occur through synapses via specialized connections with adjacent cells. These neuron collections connect with each other in a systematic way to form neural networks. Biological neurons are the key basic operational components of the brain and the spinal cord in the central nervous system (CNS). It also forms part of the ganglia in the peripheral nervous system (PNS). The biological neurons are of different types including *the sensory neurons* responding to the sense of touch, vision, sound, as well as other stimuli which affect the cells of the mammalian sensory organs which relay the signals to brain via the spinal cord. Others are *themotor neurons* which receive signals emanating from the brain via the spinal cord to bring about muscular contractions, and *interneurons* connecting neurons among one other in the same region of the spinal cord or the brain (Kandel *et al.*, 2000). Figure 2.2 shows the structure of a typical neuron consisting of the cell body, dendrites as well as axon.

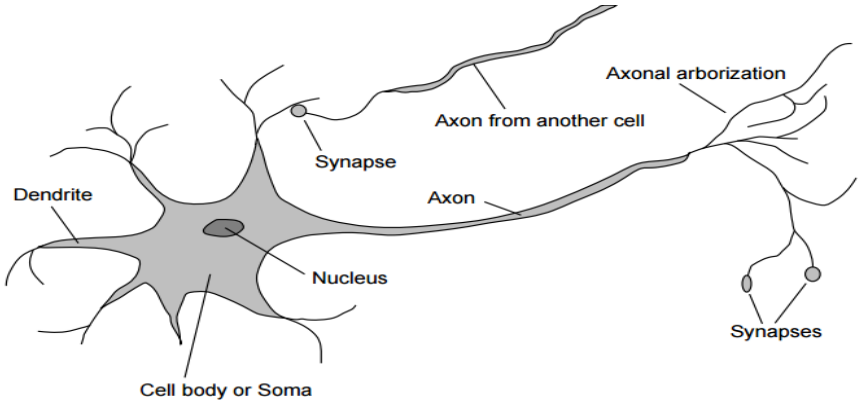


Figure 2.2: Structure of a Typical Neuron (Kandel *et al.*, 2000)

Figure 2.2 shows the structure of a typical neuron consisting of the cell body, dendrites as well as axon. Dendrites are the thin structures extending from the cell body for hundreds of micrometers. They branch a multiple number of times and give rise to a complex dendritic tree. Signals are usually sent via the axon to the dendrite of another neuron after performing some operation by the cell body particularly the nucleus (Kandel *et al.*, 2000).

2.2.3.2 Neuron network topology

The perceptronis a simplified mathematical abstraction or model of the neuron discussed above. It is essentially a binary classifier which when combined with other related counterparts leads to a significantly important results in the field of pattern recognition (Sarle, 1994). It essentially computes linear combination of inputs (with an intercept or bias term). A non-linear activation function is then applied to these non-linear combinations to produce an output. Some of the non-linear activation functions include linear or identity, hyperbolic tangent, logistic, Gaussian etc.

2.2.3.3 Activation function

The most commonly used activation function is the sigmoid activation function (Han & Moraga, 1995) which is shown in Figure 2.3

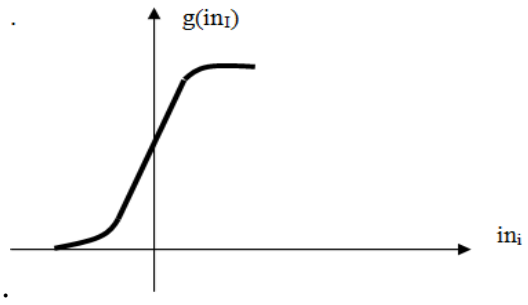


Figure 2.3: Graph of a Sigmoid Function (Han & Moraga, 1995)

It is a mathematical function with an "S" shaped curve called sigmoid or logistic function. The Logistic Sigmoid is used to introduce non-linearity in the model. It shows early exponential growth for negative in_i, which slows to linear growth of slope ¼ near $in_i = 0$, then approaches $g(in_i) = 1$ in an exponentially decaying manner. It is mathematically expressed as:

$$g(v) = \frac{1}{1+e^{-v}} \tag{2.1}$$

The expression for the weighted sum Y(x) can be generalized using a monotonic linear or non-linear Activation Function that acts on a weighted sum as:

$$Y\ x\ = g\left(\sum_{i=0}^{d} x_i w_i\right) \tag{2.2}$$

Where, g(x) = Logistic Sigmoid

A function g: R→R is called a Sigmoid Function, if the following criterion is satisfied:

$$\begin{cases} \lim_{x\to-\infty} g\ \ v\ = 0 \\ \lim_{x\to\infty} g\ \ v\ = 1 \end{cases} \tag{2.3}$$

If $|v|$ is small then f approximates a linear function

The Sigmoid Function is very useful because:

i) It is non-linear,

ii) It is monotonic,

iii) It is bounded above and below (it will never approach infinity no matter the value of "v"), and

It has a simple derivative ($f'(v) = f(v)(1 - f(v))$) (2.4)

The perceptron is a simplified working model of a neuron since it is almost impossible to simulate all the working details of a typical neuron even with modern computers. However, the simplifications is still helpful at discovering the most cognitive relevant characteristics of the neurons (O'Reilly & Munakata, 2000). In this model, the dendrites are thought of as the inputs to the system while the outputs are transferred via the axon to the synapses. The synapses are simply junctions to other neurons. The perceptron is modelled in a manner similar to the received information unit from many different inputs which are integrated to form a single real number which is used to show the level of information match of what the neuron has specialized to detect. This explains the *integrate-and-fire* neural network model (O'Reilly & Munakata, 2000). While the brain simultaneously performs the processing task across billions of distributed neurons throughout the brain called parallel distributed processing (PDP) (McClelland *et al.*, 1987), it actually contrasts the ordinary computers used today in our daily routine since the memory processing units are separate from one other in the standard computer architecture. Neurons are slow and non-deterministic and the output is also random to some extent. Therefore, from the system point of view, the input /output relationship is often accomplished using a sigmoidal function. All the inputs get multiplied by some weights, some minimum threshold for

which a neuron needs to get activated is subtracted and some neuron values fire when they are beyond this threshold or off otherwise (O'Reilly & Munakata, 2000). Figure 2.4 shows a simple perceptron with a sigmoid activation function.

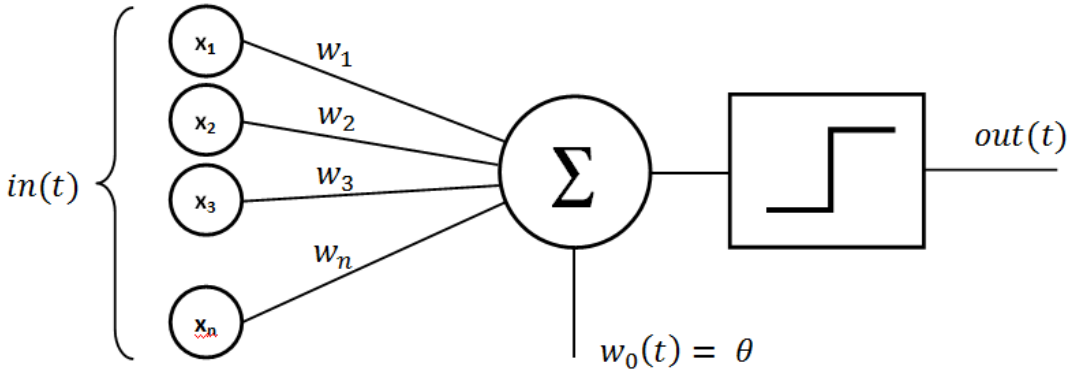


Figure 2.4: A Simple Perceptron with Sigmoid Function (Jain *et al.*, 1996)

The information processing is achieved in two basic steps, a weighted linear addition is first performed and then the result is passed through a nonlinear function (e.g. sigmoid) which produces the final output of the unit (neuron). This is given by (Jain *et al.*, 1996):

$$y = f(b + \sum_i x_i w_i) \tag{2.5}$$

Where b is the bias term, $x_i$is the i^{th} input of the neuron, $w_i$is the weight of the i^{th} input, *f* is the non-linear activation function, and yis the output. The out is the threshold to give a binary class is illustrated as follows:

$$f(x) = \begin{cases} 1 & if \quad \sum_{i=1}^{n} x_i w_i > 0 \\ 0 & if \quad \sum_{i=1}^{n} x_i w_i \leq 0 \end{cases} \tag{2.6}$$

2.2.3.4 Multi-layer Perceptron

A multilayer perceptron is a modification of the single perceptron consisting of several layers of interconnected nodes. It is a feedforward model of an artificial neural network that maps sets of input data to a set of outputs. Each node is a (or neuron) having a nonlinear activation function. MLP uses a supervised learning technique known as backpropagation for training the network (discussed in section 2.2.4). It can classify patterns that are nonlinearly separable and works well for multiclass problems (Jain *et al.*, 1996).

One possible way of classifying a neural network is the nature of their architectures or simply the interconnection between different units and layers of the neurons. Three main neural networks under this classification are the feed-forward neural networks, recurrent neural networks and symmetrically connected neural networks (Jain *et al.*, 1996).

The MLP which has a feed-forward type of architecture is the most common type of artificial neural network and its architecture is shown in Figure 2.5.

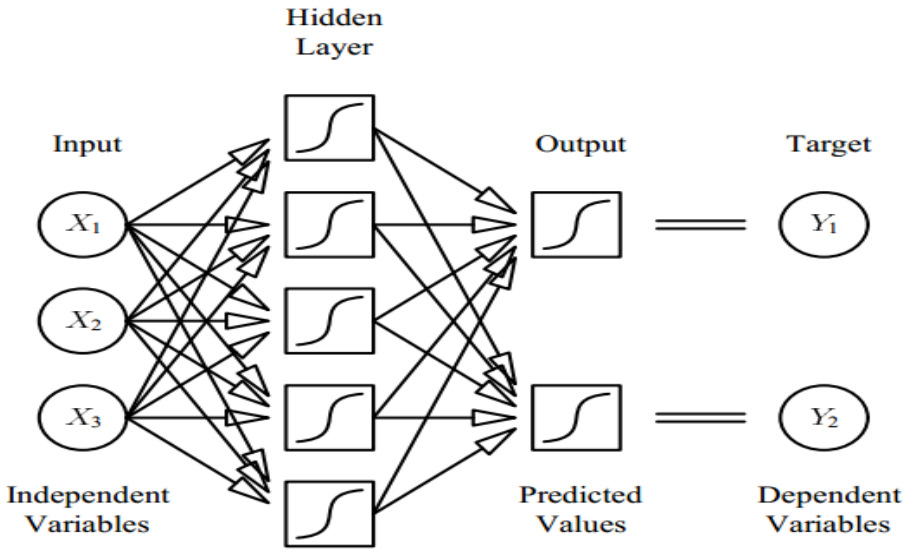


Figure 2.5: Architecture of a Multilayer Perceptron (Sarle, 1994)

It consists of an input layer as the first layer and the last output layer and one or more hidden layers or units in between them which computes series of computations and transformation. The

network is called a deep neural network if there are more than one hidden unit. So at each layer, a new representation of the input is obtained with similar items in the previous layer becoming less similar, or vice versa. This is achieved if the neurons in each layer computes a non-linear function of the activities of the neurons in the preceding layer.

2.2.4 Training Algorithm: Back Propagation (BP) Algorithm

Backpropagation is the most common method used to train neural networks. It is usually applied in conjunction with another optimization method called gradient descent where the gradient of a loss function is calculated with respect to all of the weights in the neural network. This is now fed to the gradient method which is used to update the weights with a view to minimizing the loss function. In order to calculate the gradient of the loss function, a known or the desired output for each input value is required. It is hence a supervised learning method though it may also be applied in some unsupervised networks like auto-encoders. It generalizes the delta rule with the feedforward multi-layered networks iteratively computing the gradient of each layer using chain rule(Chauvin & Rumelhart, 1995).

The backpropagation algorithm can be described briefly in two phases as follows(Chauvin & Rumelhart, 1995):

A) **Propagation Phase:** This phase consists of two steps, forward propagation and the back propagation. The forward propagation involves inputting the training data to the networks and calculating the output. The error of this output is then obtained from the ground truth of the input training data. The back propagation step involves propagating backwards the error of each neuron in each layer.

B) **Weight Update phase:** This phase involves updating the values of the neuron's weights

based on the error.

These two steps are repeated until the error is minimum which concludes the training A four step summary of the backpropagation learning algorithm as presented in (Aleksander & Morton, 1990; Williams & Hinton, 1986) is described as follows:

i) Feed the network with the input data and run the network forward to obtain the network output.

ii) Calculate the error term δ_{pj} for each of the output unit *j* for the particular pair *p*. The error is the difference of the desired output t_{pj} and the current output o_{pj} multiplied by the derivative of the activation function $f'(net_{pj})$ that maps the total input values to an output value:

$$\delta_{pj} = (t_{pj} - o_{pj}) f'(net_{pj}) \tag{2.7}$$

iii) Calculate the error by recursively computing δ for each hidden unit's *j* in the current layer. Where w_{kj} denotes the weights in the *kth* output connections of the hidden unit *j*, δ_{pk} are the error signals obtained from the *k* units in the next layer and $f'(net_{pj})$ denotes the derivative of the activation function. The error signal is propagated backward through all hidden layers until the input layer is reached.

$$\delta_{pj} = \sum_k \delta_{pk} w_{kj} f_j'(net_{pj}) \tag{2.8}$$

Update weights and the biases as follows and repeat steps 1 through 4 repeatedly until the error is minimum or below some acceptably low threshold value. Given:

$$\Delta W = -\eta \delta_l y_{l-1}$$
$$\Delta \theta = -\eta \delta_l \tag{2.9}$$

Apply

$$W \leftarrow W + \Delta W$$
$$\theta \leftarrow \theta + \Delta\theta$$

The parameter η in the above algorithm is the learning rate.

A detailed implementation of the BP algorithm for the sigmoid function is shown in Appendix A.

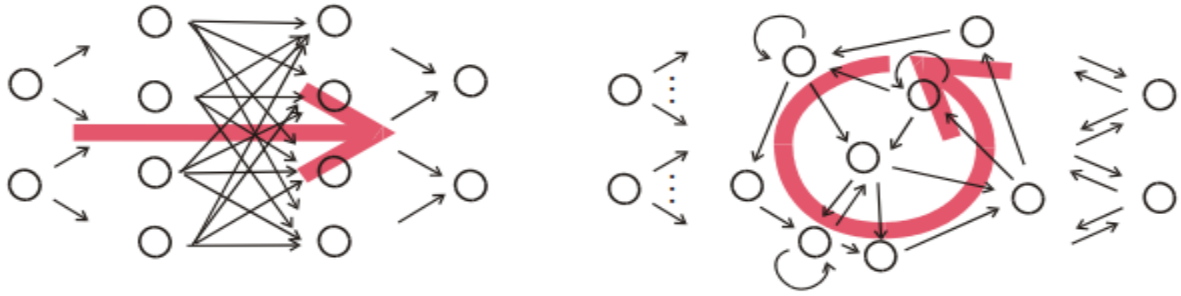
2.2.5 Deep Learning

Deep learning or hierarchical learning has since 2006 emerged as a new research area in the field of machine learning (Bengio, 2009; Hinton *et al.*, 2006)**.** It is based on algorithms that exploit multiple nonlinear layers of information processing for performing pattern analysis and classification by means of a supervised or unsupervised extraction and transformation of signals (Deng & Yu, 2014)**.** The algorithm attempts to learn multiple levels of nonlinear representation with a view to modelling complex relationships amongst data. Thus, higher- level concepts and features are defined in terms of those of lower level with each level corresponding to a different abstraction level. This hierarchy of features is known as a deep architecture (Yu & Deng, 2011). Deep learning methods are wide and are even becoming increasingly richer and richer to encompass those of hierarchical probabilistic models, neural networks and a variety of supervised and unsupervised feature learning algorithms such as: deep belief network (DBN), Boltzmann machine (BM), restricted Boltzmann machine (RBM), deep neural network (DNN), and deep auto-encoder (Yu & Deng, 2011).

2.2.5.1 Deep neural networks (DNN)

DNN is simply the implementation of deep learning in neural networks. Hence, DNNs are multilayer perceptron which has many hidden layers with fully connected weights often initialized using either a supervised or an unsupervised pre-training technique. With, deep learning introduced to neural network, the computational efficiency is improved rapidly but

training becomes more difficult (Jaeger, 2002).Two types of DNNs exist: feedforward and the recurrent DNN. These are shown in Figure 2.6.



(a) Feed forward (b) Recurrent

Figure 2.6: Structure of a Typical Neural Network (Jaeger, 2002)

In the feedforward DNN, the activation in the network is channeled or piped from the input to the output units through the hidden layers. An example is the multilayer perceptron MLP which implements simple input output mappings of functions. On the other hand, the recurrent neural network (RNN) has at least one cyclic path of the synaptic connections. RNNs mathematically implement and approximate dynamical systems with arbitrary precision. Practical and theoretical difficulties have by and large have so far prevented its practical applications (Jaeger, 2002).

2.2.5.2 Convolutional neural network

Convolutional neural networks (CNN) or simply ConvNets are a class of neural networks which are designed to process data such as images that come in form of a multiple arrays. Color images for example are composed of three 2-Dimensional arrays comprising of pixel intensities in three color channels. Other data modalities that exist in this form of multiple arrays include 1D signal and sequences, 2D images or audio spectrograms and 3D volumetric images or videos. Four basic ideas behind CNNs that take advantage of natural signal properties are: local connections, pooling, shared weights and the use of many layers (LeCun *et al.*, 2015). A typical structure of a neural architecture is consists of a series of stages as shown in the Figure 2.7

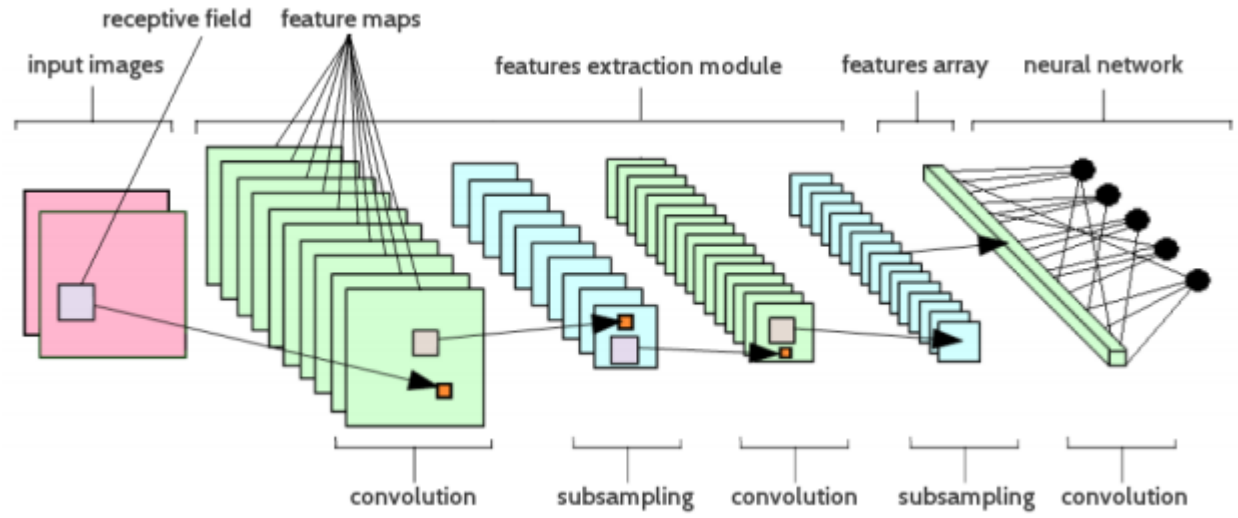


Figure 2.7: A Typical CNN Architecture (Yuan *et al.*, 2014)

As can be seen from Figure 2.7, four main layer types form the CNN architecture (Vedaldi & Lenc, 2015), namely:

A) **Convolutional Layer:** this represents the core building block of the network. Its parameters are a set of learnable filters performing dot product on the input, the CNN then learns filters that activate when a particular type of feature is detected in the input. Hence, units in this layer are organized in feature maps with each unit connected to local patches of the previous layer's feature map through a set of weights known as filter bank. This dot product operation performed by a feature map is mathematically a discrete convolution operation and it role is basically to detect local conjunctions of features from the previous layer.

B) **Pooling Layer:** The role of this layer is to semantically merge similar features into one thereby reducing the spatial size of input with a view to diminishing the number of computation and parameters in the CNN.

C) **Fully Connected Layer:** Neurons in this layer are fully connected to all previous layers' activation like in regular neural networks. Hence, there activations can be computed using

matrix multiplication and followed by a bias offset. They are interpreted as being 1x1 convolutional layers. This layer computes the class score.

D) **Rectified Linear Unit (ReLU) Layer (non-linearity):** The local weighted sum results are usually passed through a layer of non-linearity such as the Rectified Linear Unit (Red in Figure 2.8). The ReLU computes a function which is basically a threshold at zero as shown:

$$(x) = \max(0, x) \tag{2.7}$$

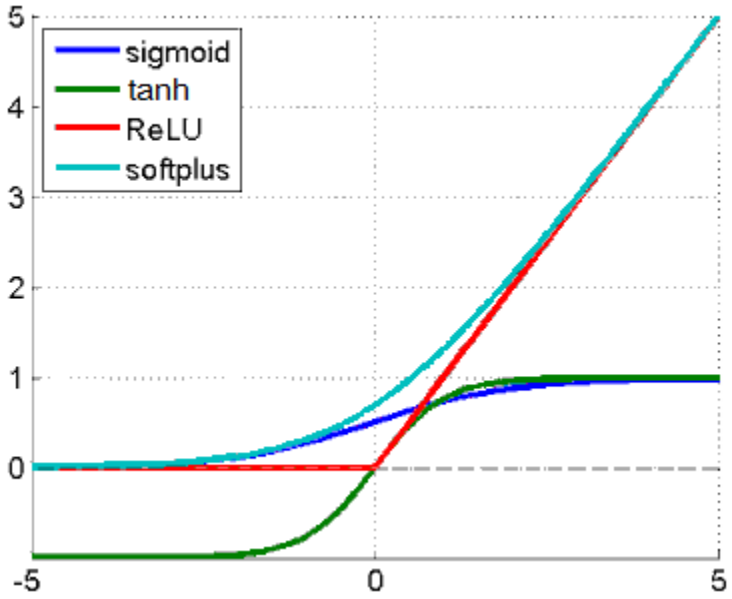


Figure 2.8: Non-Linear Functions

Although several options exist for the nonlinearity like the Tanh and the Sigmoid, the rectified linear unit (ReLU) is often a preferred option because it simplifies backpropagation, it makes learning faster and avoids saturation issues.

2.2.5.3 Advantages of using deep CNNs

The following are the advantages of using the deep CNNs:

A) **Scalability:** Deep neural networks scale to millions of parameters which give them the capacity to learn concepts that are highly complex and thousands of categories. With the

abundance of data and modern hardware larger and more powerful networks are not able to be trained easily.

B) **Speed:** Trained models store their knowledge in learned parameters compactly which makes them easy to be deployed in any environment. Therefore, storing any additional data is not needed to make predictions for new inputs. Hence, they can easily be used on embedded devices while providing responses in milliseconds.

C) **Flexibility:** Models learn to extract features that are discriminative from the input using the provided training data, unlike the traditional computer vision approaches which uses hand-engineered feature extractors such SIFT and local binary patterns (LBP). They can therefore easily adapt to problems in any domain.

2.3 REVIEW OF SIMILAR WORKS

Computer vision researchers have over the years undertaken several approaches to object recognition. This section highlights some of these researches which adopted image fusion strategies and/ or deep learning based methods for performing image recognition which are the approaches jointly used in this research work.

Kong *et al* (2007) described a software-based method for registration and fusion of visible and thermal infrared images for performing face recognition. This data fusion technique finds a combination of multimodal images at a sensor level to obtain an integrated image that is enhanced in terms of its information content for performing pattern classification or recognition. The two modes of the images are aligned by means of a Gaussian criterion for measuring visual similarity and spatial proximity. The pair of co-registered visible and IR images are fused in the wavelet transform domain in order to produce images that are insensitive to variation in

illumination. The combination of these multimodal images was found to significantly improve the accuracy and performance of the face recognition in uncontrolled operating environments. However, a pair of visible and IR images needs to be spatially co-registered before extracting any useful information from the fused images. Hence, the software approach to image registration was adopted since camera systems that produce co-registered images may be expensive or not readily available.

Chang *et al* (2008) carried out a study on multispectral visible and near infrared imaging for face recognition as alternative means to the conventional color or broadband monochrome imaging sensors in order to enhance face recognition performance in an uncontrolled illumination conditions. The work showed that multispectral imaging in both spectra helps in reducing color variations that exists in the face due to the change in source types and directions of the illumination. Several other works followed by adopting their recognition algorithms on multispectral images. However, the work still relies on the traditional non-deep learning approaches which have been shown to be suboptimal.

Bendada and Akhloufi (2010) presented a work which extended the use of non-visible spectra for their face recognition algorithm. The work introduced the use of Local Binary Patterns (LBP) which has been used in visible spectrum face description for efficient multispectral recognition. The proposed texture space was less sensitive to illumination change, noise, and facial expressions which made it a better candidate for effective multispectral face recognition. Results obtained showed that the proposed texture descriptors helped in achieving high performance in the multispectral face recognition. The drawback of this work is that LBP takes high computational time as compared to others like SURF features. Also, this recognition approach is

a non-deep learning method which has been proven to surpass the state of the art recognition methods.

Recently, Krizhevsky *et al.*, (2012) published the historic paper whose technique won the popular ImageNet competition. It is widely seen as the revolution of deep learning by the computer vision community. In the historic work, a large deep convolutional neural network was trained to classify more than a million images of high-resolution in the ImageNet LSVRC contest into the different 1000 classes. The network achieved a top-1 and top-5 error rates of 37.5% and 17.0% on the test data respectively. This result was considerably better than those obtained from previous state-of-the-art methods. The neural network used had 60 million parameters and about 650,000 neurons consisting of five convolutional layers, some of which are followed by max-pooling layers, and then three fully-connected layers with a final 1000-way softmax. However, the network required Graphical Processing Unit (GPU) implementation of the convolutional operations due to slower training and also employed regularization due to overfitting in the fully connected layers. Following this successful work, several other researchers built and improved on the idea to enhance their recognition task.

Howard (2013) published a paper which investigated multiple techniques for improving upon the state of the art in deep convolutional neural networks based on image classification pipeline. The system developed based on the modifications achieved a top-5 classification error rate of 13.55% entry in the Imagenet Large Scale Visual Recognition Challenge 2013 when no external data was used. There was an over 20% improvement as compared to the ILSVRC12 winner. The modification techniques included adding more image transformations to the training data, adding more transformations to generate additional predictions at test time and using complementary

models applied to higher resolution images. However, the possibility of using images in multiple modes was not envisaged in the research and only images of single modality were used.

Szegedy *et al* (2014) proposed a deep convolutional neural network architecture called GoogleNet, which set a new state-of-art technique for detection and classification in the ILSVRC14. An improved utilization of the computing resources inside the network was actualized by a carefully crafted design which allowed for increasing the network's width and depth while the computational budget was kept constant. The paper used a wider and deeper inception network and the architectural decision was based on the intuition of multi-scale processing and the Hebbian principle with a view to optimizing the quality. However, the quality of the network was slightly inferior although its addition to the ensemble seemed to have marginally improved the results. Furthermore, these deep learning approaches were implemented using a single mode of visible image.

Zhang *et al* (2015) introduced visible and infrared spectra (VAIS), the world's first dataset of paired visible and IR ship images to be made publicly available. In the paper, two classification algorithms, namely convolutional neural networks (CNNs) and gnostic fields were used for performing recognition on six classes of ships. A classification accuracy of 56.8%, 81.0%, and 87.4% was obtained on IR, visible and combined IR + Visible images respectively by using a combination of CNN and Gnostic fields. The Gnostic fields were trained with SIFT features in order to complement the CNN based approach. A CNN which has been pre-trained on ImageNet dataset is also used to extract features from the VAIS dataset to avoid overfitting. However, fine-tuning the CNN and re-training using backpropagation with the VAIS's training images and correctly applying regularization techniques such as drop-out could have improved the results as

Gnostic field still relied on applying hand crafted features which have been proven to be suboptimal on their own.

Therefore, this research seeks to maximally exploit the power displayed by deep convolutional neural networks as an end to end learning architecture as highlighted by these reviewed works. This will be achieved by exploiting the synergistic integration of the information of a scene obtained in the visible and infrared spectrums of same data with a view to improving the overall classification accuracy.

CHAPTER THREE

METHODOLOGY

3.1 INTRODUCTION

This research investigated a number of image recognition frameworks on a dataset of visible-light and infrared (IR) imagery approaches enumerated in Chapter Three. However, the overall objective is to investigate same approaches on a fused framework of visible and IR images in

order to exploit the synergistic integration of the information obtained from the varying spectra on same data with a view to improving the overall classification accuracy. A combination of these data from multiple sources yields a more informative representation which guarantees better recognition accuracy than the original. To achieve this, the following research methodology steps listed in section 1.4 was adopted:

3.2 DATA COLLECTION

Two sets of datasets were collected and used for this research. The VAIS (Zhang *et al.*, 2015) Dataset and an RGB-NIR Scene Dataset from the Images and Visual Representation Laboratory (IVRL) of EcolePolytechniqueFédérale de Lausanne – EPFL. The individual dataset statistics are discussed under:

3.2.1 VAIS Dataset

The VAIS dataset is a collection of unregistered thermal and visible images of ships simultaneously acquired from pairs and suitable for performing object recognition or classification research. The main paper (Zhang *et al.*, 2015) which uses the dataset for recognizing maritime imagery claim it to be the world's first publicly released dataset of a paired visible and infrared imagery of ships containing more than a thousand paired visible (RGB) and infrared (IR) images among six categories of ships namely - merchant, passenger, sailing, tug, medium, and small. For the Visible images an ISVI IC-C25 camera was used to capture a 5,056×5,056 bayered RGB pixel images while a Sofradir-EC Atom 1024 camera captures a 1024×768 pixel images

The overall data details are given under:

Total Images = 2865

Total Infrared Images = 1242

Total Visible Images: 1623

Total Visible/Infrared Pairs = 1088

Number of Night IR Images = 154

Number of unique ships = 264

Number of Fine-Grained Categories = 15

Number of Basic Categories = 6

3.2.2 RGB-NIR Scene Dataset

The dataset contains a total of 477 images in nine (9) categories captured in the visible (RGB) and Near-infrared (NIR) spectrums. These images were captured by using separate exposures from a modified SLR cameras by means of visible and NIR filters. The 9 scene categories are: water, urban, street, old building, mountain, indoor, forest, field, and country. The RGB-NIR Scene Data of about 1 GB size has TIFF images at a resolution of 1024x768 which were further Processed and aligned. Also, the images were captured using a Nikon D90 as well as a Canon T1i cameras and using B+W 486 filter for visible and 093 filter for the NIR images. The cutoff between these two filters was approximately 750nm.

3.3 DESIGNING A SIMPLE EXPERIMENTAL NETWORK

In the first experimental setup, a simple deep convolutional neural network was designed so as to train using the dataset, a deep neural network from scratch. An end-to-end learning approach is adopted whereby representations that are highly non-linear are learned from the dataset in order

to maximize the classification objective which is in this case, 6-way or a 9-way classification for the two sets of available datasets. The architecture of the network is shown in Figure 3.1 for the 6-way classification on the VAIS dataset. The same architecture was applied on both datasets.

layer	0	1	2	3	4	5	6	7	8	9	10
type	input	conv	mpool	relu	dropout	conv	mpool	relu	dropout	conv	loss
name	n/a	conv1				conv1				fc1	hinge loss
support	n/a	3	4	1	1	5	3	1	1	12	1
filt dim	n/a	1	n/a	n/a	n/a	10	n/a	n/a	n/a	14	n/a
num filts	n/a	10	n/a	n/a	n/a	14	n/a	n/a	n/a	6	n/a
stride	n/a	1	2	1	1	1	2	1	1	1	1
pad	n/a	0	0	0	0	0	0	0	0	0	0
rf size	n/a	3	6	6	6	14	18	18	18	62	62
rf offset	n/a	2	3.5	3.5	3.5	7.5	9.5	9.5	9.5	31.5	31.5
rf stride	n/a	1	2	2	2	2	4	4	4	4	4
data size	64	62	30	30	30	26	12	12	12	1	1
data depth	3	10	10	10	10	14	14	14	14	6	1
data num	50	50	50	50	50	50	50	50	50	50	1
data mem	2MB	7MB	2MB	2MB	2MB	2MB	394KB	394KB	394KB	1KB	4B
param mem	n/a	400B	0B	0B	0B	14KB	0B	0B	0B	47KB	0B

Figure 3.1: The Experimental Three-Layer CNN

The network consists of convolutional layers (3x3x10, 5x5x14), a max pool layer and a ReLU. The last convolutional layer is called a fully connected layer (12x12x14-> 1x6). This is because, it has an output with a spatial resolution of 1x1 and each unit in its output is a function of the previous layer entirely hence the name fully connected layer. It is, however, mathematically similar to the other convolutional layers and that is why Matconvnet(Vedaldi & Lenc, 2015) specifies them in the same manner. There is also a dropout layer which is used for regularization.

The weights represent the learned filters and they are numbers that are initialized randomly using a Gaussian distribution. From Figure 3.1, it is seen that the filters have a 3x3 spatial resolution and there are 10 of such filters. A constant or a bias offset is also learned by the network so as to associate it to the output of each filter. The next layer which is the max pool layer takes a

maximum over an appropriate sliding window and subsamples the resulting map/image with that a stride of similar size resulting in a corresponding decrease in spatial resolution while the depth remains the same. Other pooling possibilities apart from the max pooling such as the average pooling do exist and may be used. The function of the non-linearity is to set to zero (0) any feature map values from the pooling layer which are negative. Also, other nonlinearity possibilities such as the sigmoid also exist. It should be noted that the ReLU and max pool layers do not have learned parameters associated with them and their behavior are hand specified entirely and so they have no weights to initialize like the convolutional layers. Finally, it is the fully connected layer which has learned filters that operate on the sub-sampled, rectified and max-pooled filter responses from initial or first layer. Its output must be a 1x1 with a filter depth equal to the number of classification categories (say 6 for the VAIS dataset).

One more layer which is used for training called 'loss layer' is added at the top of the network. Various possibilities exist for the loss functions but this experiment makes use of the "softmax loss". According to literature, ways of obtaining the appropriate network architecture is not so obvious and therefore designing the right network takes a lot of trial effort to obtain a training strategy that guarantees optimal performance.

3.4 USING PRE-TRAINED NETWORKS FOR CLASSIFICATION

One impressive and interesting thing about representations that are learned by convolutional neural network is that they surprisingly tend to generalize well to other recognition tasks. This is unexpected since these networks are trained to discriminatively perform well at specific tasks. However, training a neural network on small datasets would lead to overfitting. In order to overcome this, we resort to using pre-trained networks for classification where two sets of experiments were conducted using two different sets of pre-trained models. This works since

these standard models after being trained on more than a million images and from about a thousand classes have learned features which are discriminative for performing object recognition in general.

3.4.1 Extracting Feature Using CNN Pre-trained on Imagenet

In this experiment, a CNN that is pre-trained on imageNet is used to extract features from the datasets. This network has been trained on over a million images from about 1000 classes and guarantees that the network has learned features that are very discriminative for performing object recognition in general. The 16-layer CNN called *imagenet-vgg-verydeep-16*(Simonyan & Zisserman, 2014) which achieves an excellent result on imagenet ILSVRC-2012 dataset was used to extract features from our datasets. This is implemented using the MatConvNet Matlab toolbox for CNN. A Multi-class logistic regression classifier was trained on the output of its fifteenth weight layer using the ReLU non-linearity. This is because, the last layer of the CNN which is specific for recognizing the output of the 1000 categories of the ILSVRC is not used. The network accepts an RGB IMAGE OF 224x224. Therefore, to handle our infrared (IR) images, the single IR channel is duplicated three times so as to create simulated 3–channel images like the RGB. The LIBLINEAR toolbox (Fan *et al.*, 2008) was used to train the multiclass logistic regression model.

3.4.2 VGGNet

The VGGNet(Simonyan & Zisserman, 2014) developed in 2014 was the winning architecture in the ILSVRC-2014 competition. It is a very deep and a more powerful network which has 16-19

layers. The structure is described in Figure 3.2 11. Five structures were initially designed and after some evaluations and experiments, designs D and E emerged as the best structure with E a little better than B in terms of performance. VGGNet has the characteristics that it applies multiple convolutional layers with a small window sizes as opposed to a convolutional layer having large window size and then followed by pooling layer. This makes the network more flexible.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 3.2: The Structure of VGGNet(Simonyan & Zisserman, 2014)

3.4.3 Fine-tuning the Pre-trained CNN with Backpropagation

In another phase of adopting the pre-trained network, the pre-trained CNN was fine-tuned using backpropagation with the training datasets. In this scenario, the final layer of an existing network is replaced with random weights and the entire network is trained again with the given images and their ground truth label for performing the recognition task. We are hence simply treating the pre-trained network as a better and more robust initialization than the randomly generated

weights used when we are training from scratch. In this network, the VGG-F network called *imagenet-vgg-f* which was meant to have a similar architecture with the original AlexNet network was used.

3.4.4 Alexnet/Imagenet

The network was developed by Krizhevsky *et al.*, (2012) and is used widely nowadays. The architecture is shown in Figure 3.3.

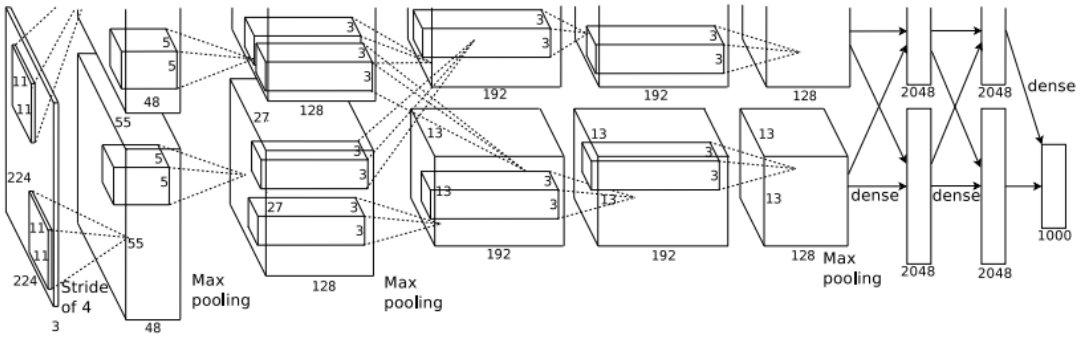


Figure 3.3: The Structure of AlexNet(Krizhevsky *et al.*, 2012)

It consists of five convolutional layers and three fully-connected layers. The structure of the AlexNet network may be observed to be divided into two blocks. This is because two GPUs were used by the authors to train their datasets in parallel. The network is employed widely in large-scale object classification. The last layer consists of 1000 neurons because the architecture was designed originally to identify 1000 objects. In this experiment therefore, the last layer was replaed by fine-tuning the network. The authors performed several experiments to get better result and therefore, the performance of the structure is very stable and this net is used widely in many applications. The performance of these models from 2012- 2014 on the ILSVRC 2012 validation data is summarized in the Table 3.1:

Table 3.1: Summary of Models' Performance on ILSVRC 2012 Validation Data

Model	Introduced	top-1 err.	top-5 err.	images/s
matconvnet-vgg-verydeep-16	2014	28.3	9.5	184.5
vgg-verydeep-19	2014	28.7	9.9	154.5
vgg-verydeep-16	2014	28.5	9.9	183.1
googlenet-dag	2014	34.2	12.9	501.8
matconvnet-vgg-s	2013	37.0	15.8	415.9
matconvnet-vgg-m	2013	36.9	15.5	623.1
matconvnet-vgg-f	2013	41.4	19.1	793.1
vgg-s	2013	36.7	15.3	395.4
vgg-m	2013	37.3	15.9	586.9
vgg-f	2013	41.1	18.8	785.7
vgg-m-128	2013	40.8	18.4	588.7
vgg-m-1024	2013	37.8	16.1	596.8
vgg-m-2048	2013	37.1	15.8	589.4
matconvnet-alex	2012	41.8	19.2	760.3
caffe-ref	2012	42.4	19.6	384.8
caffe-alex	2012	42.6	19.6	382.4

3.5 PERFORMANCE METRIC

The performance metric used in this research is the classification accuracy (%) which is evaluated as (1-classification error) x100%. This is because MatConvnet returns the classification error (*err*) after running a MATLAB file called *ccn_train.m* on the datasets which implements the backpropagation algorithm using stochastic gradient descent (SGD) on the

Convolutional Neural Networks. The code snippet that returns this error from the *cnn_train.m* for a multiclass problem like ours is shown hereunder:

```
% -------------------------------------------------------------------
function err = error_multiclass(opts, labels, res)
% -------------------------------------------------------------------
predictions = gather(res(end-1).x) ;
[~,predictions] = sort(predictions, 3, 'descend');

% be resilient to badly formatted labels
if numel(labels) == size(predictions, 4)
  labels = reshape(labels,1,1,1,[]) ;
end

% skip null labels
mass = single(labels(:,:,1,:) > 0) ;
if size(labels,3) == 2
  % if there is a second channel in labels, used it as weights
  mass = mass .* labels(:,:,2,:) ;
  labels(:,:,2,:) = [] ;
end

error = ~bsxfun(@eq, predictions, labels) ;
err(1,1) = sum(sum(sum(mass .* error(:,:,1,:)))) ;
err(2,1) = sum(sum(sum(mass .* min(error(:,:,1:5,:),[],3)))) ;
```

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 INTRODUCTION

In this section, the performance of the three methods employed was evaluated using accuracy (%) as a performance metric. The results obtained by applying the methods on visible, IR and their fused multimodal versions are also discussed.

4.2 PRELIMINARY RESULTS

In the first preliminary experiments, the three proposed methods were evaluated on visible and infrared images separately and the results obtained are shown in Table 4.1

Table 4.1: First Results for VAIS Datasets

Image recognition framework	Visible Image	IR Image
Previously trained 16 -layer CNN	80.76%	64.60%
Training a 3-layer CNN from scratch	74.27%	58.61%
Fine-tuning , pre- trained 16 -layer CNN	82.53%	68.14%

It can be seen that in all the cases, the accuracy of the algorithms on the visible images are higher than that of the infrared. This suggests that these algorithms learned more discriminative features from the visible images than the infrared images. It is also observed that the pre-trained model performs well and fine-tuning improves this performance, and that the simple model gives promising performance despite its simplicity and the limited size of the training data.

4.3 MODIFYING THE CNNs TO ACCEPT 4-LAYER MULTIMODAL IMAGES

The designed three–layer CNN as well as the standard networks are then modified to accept the four-layer fused multimodal images and the experiments were then extended to the fused version

of the images. To avoid any disparity in the results arising from the difference in image quality, the training and the test data were randomly selected from the entire datasets using different allocations such as 80-20, 70-30, 60-40 and 50-50.

4.3.1 Results for Modified 3-layer CNN

These experiments were then extended to the fused version of the images and the results obtained for the 3-layer CNN are shown in Table 4.2.

Table 4.2: Second Results for VAIS Datasets

Dataset	**50-50**	**60-40**	**70-30**	**80-20**
IR	66.6667%	67.5676%	67.9878%	70.1835%
VISIBLE	81.3187%	82.6577%	85.0610%	86.6972%
Paired	83.7989%	82.9837%	86.8590%	89.4472%

As can be seen from Table 4.2, the experiments were carried out on different train-test partitions. While the results were approximately consistent, the accuracy is higher as the number of training images increased. Most importantly, the accuracy of the networks is higher for the fused images which suggested that networks exploit the synergistic integration of the information obtained from the varying spectra on same data thereby improving the overall classification accuracy.

In Machine learning, typical train-test split for data is 70-30. Therefore, the graphs for the classification when the 70% - 30% dataset partition was used are as shown in Figures 4.1 – 4.3:

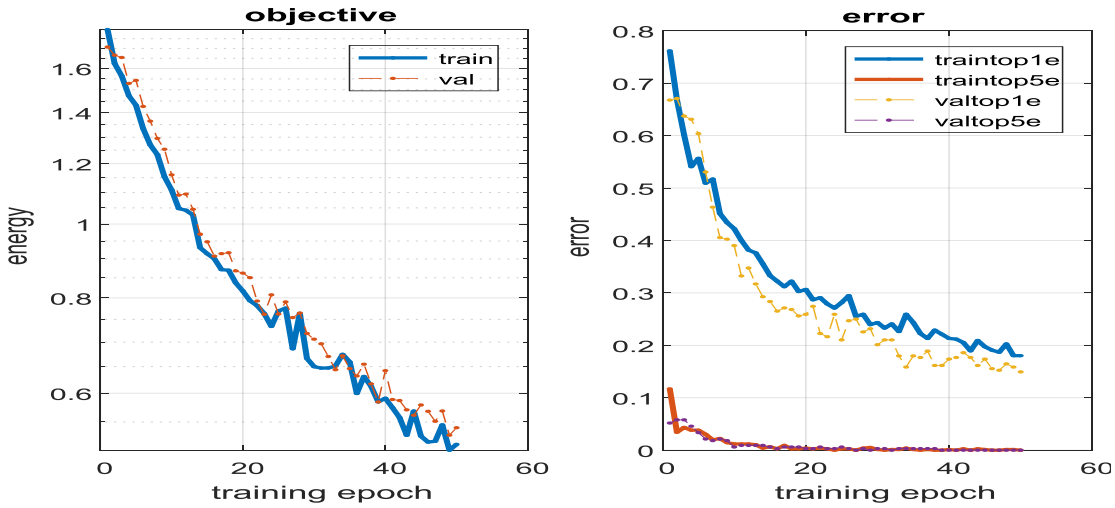


Figure 4.1: Visible VAIS Results

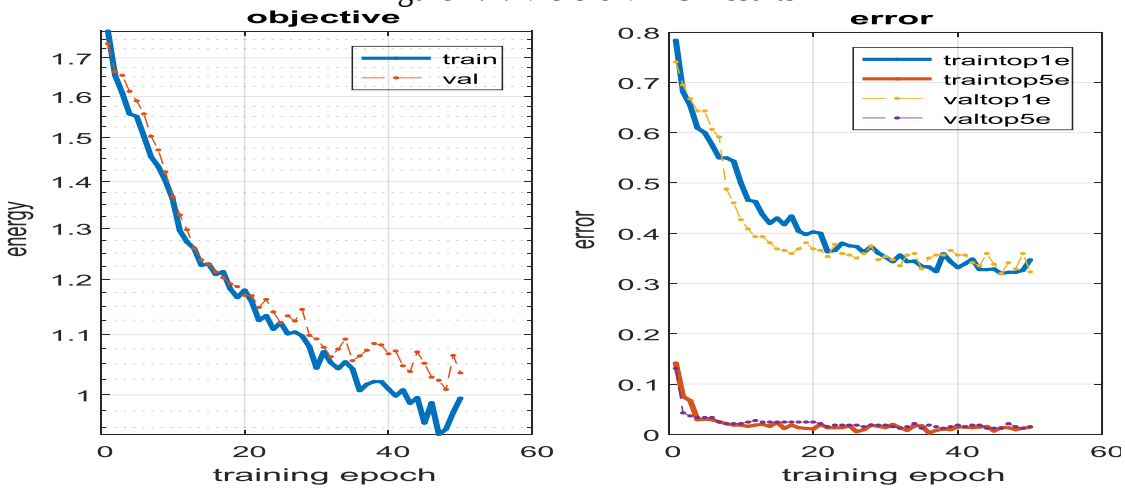


Figure 4.2: IR VAIS Results

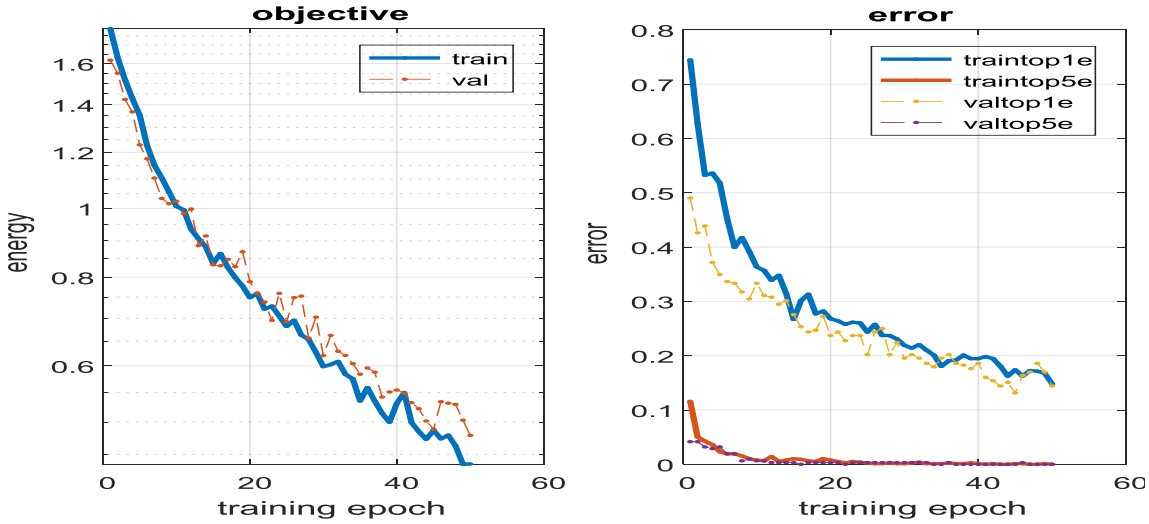


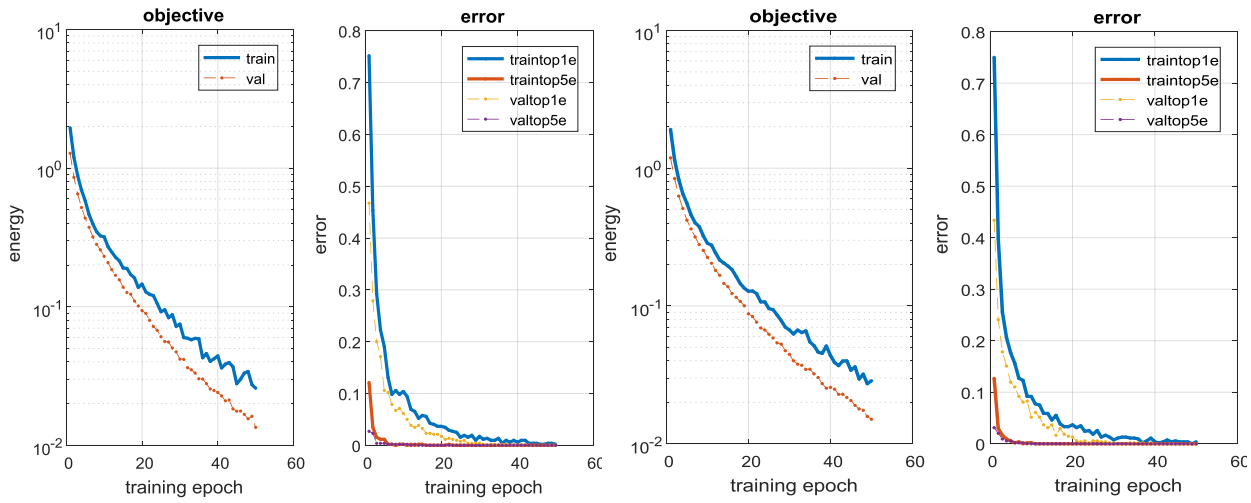
Figure 4.3: Paired VAIS Results

Figures 4.1 through 4.3 showed respectively the performance of the system for visible, IR, and paired after 50 training epochs (i.e. 50 passes through the training set). In the left pane, blue represents the training error and dashed orange the validation error across training epochs. Each training epoch denotes a pass over the entire training data broken up into 50 "batches" of the training instances. A "loss" is usually incurred by the network when it makes mistakes and the weight of the network is updated using backpropagation in such a direction as to decrease the loss. Hence, the blue line is more or less expected to decrease monotonically. The orange dashed line on the other hand is the error that is incurred on the test data. This is referred to as "val" or "validation" in the figures.

The right pane displays the training as well as the testing accuracies on the training and test data across same training epochs. The top 1 error (how often the highest scoring guess is wrong) and top 5 error (how often all of the 5 highest scoring guesses are wrong) are shown. The top 1 error on the test set is the case of interest in this research. It is seen that as the, training progresses and the network initial random weights get updated, the accuracy increased.

4.3.2 Results for Modified Pre-trained CNN

Interestingly, on fine-tuning the pre-trained model on the dataset, and randomly selecting the training and the testing images from the dataset based on 70-30 partition, excellent results are obtained which approaches an accuracy of 100% after about 40 training epochs as shown in Figure 4.4 (a-d) for the visible (left) and the paired (right) respectively.



(a) (b) (c) (d)

Figure 4.4: Results for Fine-tuning a Pre-trained Model

This result surprisingly show that representations that are learned by deep convolutional networks tend to surprisingly generalize well to other recognition tasks. This is seen in figures (a) and (c) where training and the validation objectives closely follow each other. The high accuracy recorded is largely due to the fact that the initial random weight used is more robust as the adopted pre-trained deep convolutional network was treated as a better initialization when compared with the random weights when training from scratch.

4.4 EXTENDING SIMILAR EXPERIMENT ON THE RGB-NIR SCENE DATASET

Extending similar experiment on the RGB-NIR scene dataset, similar behavior as obtained using the VAIS datasets was obtained which shows that the method can be extended to other suitable datasets. Some of the results obtained for the RGB-NIR datasets are shown in Figures 4.5-4.7

4.4.1 Results for Training from Scratch Using RGB-NIR Scene Dataset

When the network was initially trained using infrared and visible images separately, accuracy values of 46.90% and 54.83 % were respectively obtained as shown in Figure 4.5

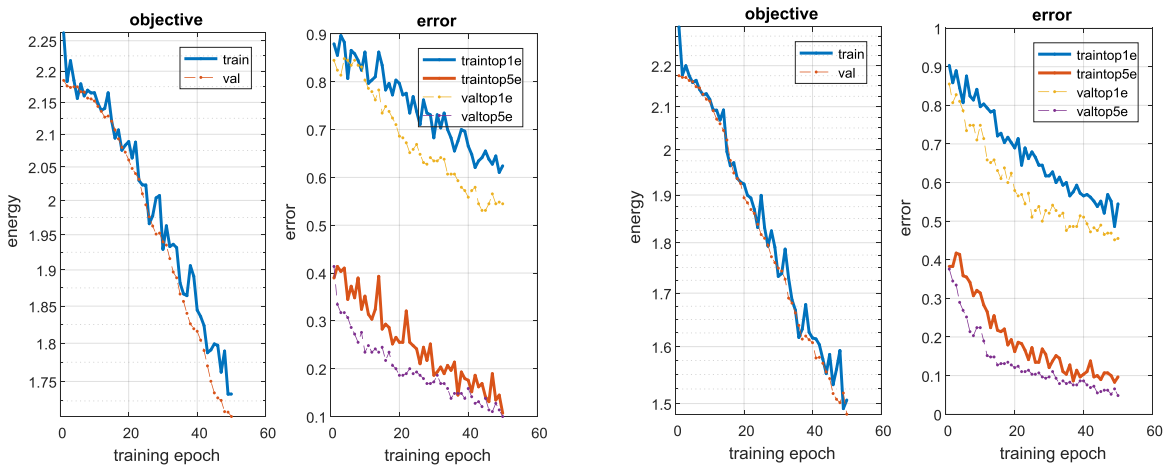


Figure 4.5: Response for Training IR (left) and Visible (right) Images

However, an accuracy of 59.73% was obtained on the paired dataset as shown in Figure 4.6

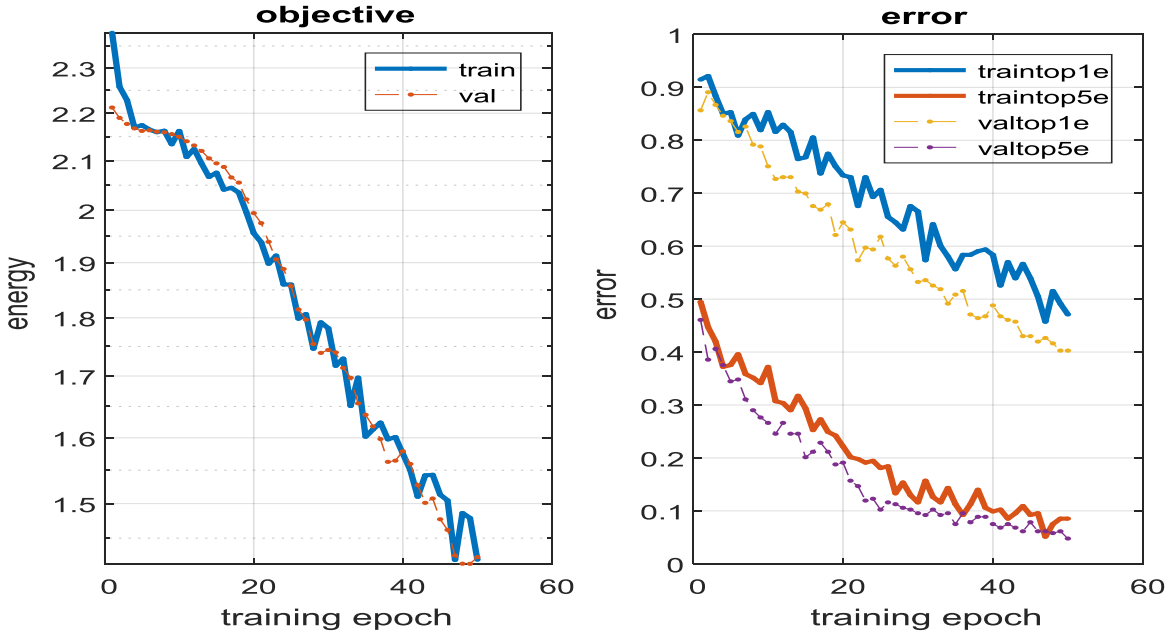


Figure 4.6: Response for Training Paired RGB-NIR Images

4.4.2 Results for Fine-tuning Using RGB-NIR Scene Dataset

The results of fine-tuning a pre-trained network also gave similar result as that of the VAIS datasets. This is shown in Figure 4.7.

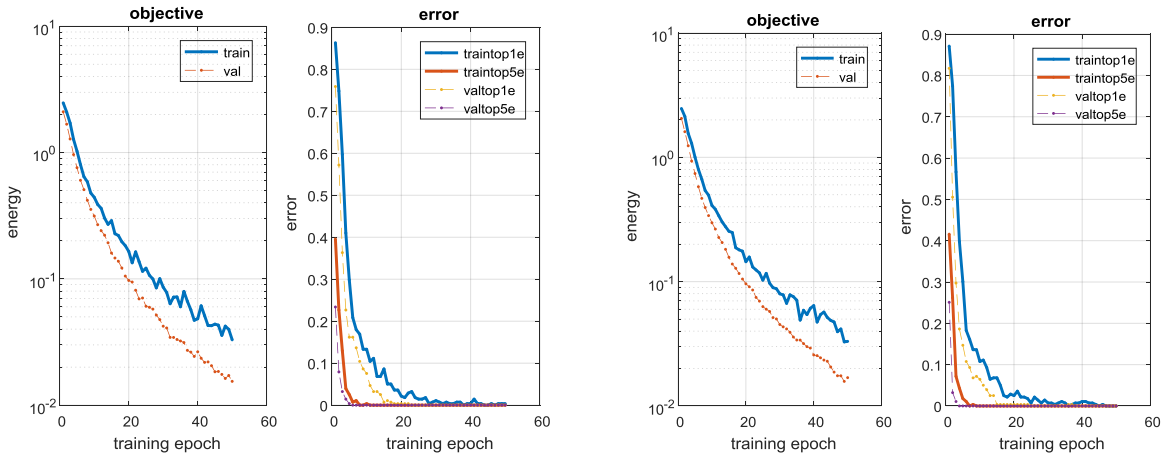


Figure 4.7: Results for Fine-tuning a Pre-trained Model on the RGB-NIR Scene Dataset

As can be seen, exactly similar behavior as obtained using the VAIS datasets which shows the methods can be extended to other suitable datasets.

4.5 VALIDATION

The results obtained are compared with those of Zhang *et al.*, (2015) in order to validate our results. The work used as a benchmark since this research adopted the VAIS dataset published by this group of researchers and also adopted in their work. This is presented in Table 4.3:

Table 4.3: Validation Results

	IR	**Visible**	**Paired**
VAIS_ CNN	54.00%	81.90%	82.1%
Developed CNN	67.99%	85.06%	86.86%
VAIS_Gnostic Field + CNN	56.80%	81.00%	87.40%
Developed Fine-tuned CNN	100.00%	100.00%	100.00%

From Table 4.3, it can be seen that CNN can be tweaked to maximally exploit its recognition power. Furthermore, the adopted fusion technique presented a promising method for improving the classification accuracy. Also, it is observed that while the combination of Gnostic field and CNN performed well for IR images while the developed CNN performed better for both the visible and IR images. Finally, while the developed fusion approach is promising, the fine-tuned developed CNN model performed excellently well.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 SUMMARY

This research work has developed a comprehensive object recognition frameworks using deep convolutional neural networks by jointly learning image representations from fused multimodal images of visible-light and infrared (IR) which was aimed at improving the overall classification accuracy. This chapter summarized the whole dissertation. Conclusion, recommendation and limitation encountered during the course of this research are also presented. Areas of future research have also been suggested.

5.2 SIGNIFICANT CONTRIBUTIONS

A lot of research works have been carried out in the area of object recognition using convolutional neural network. Similarly, many research based on image fusion have been carried out.

The significant contributions of this work are as follows:

i) Adopting a simple and novel fusion strategy where the IR and visible images are fused by concatenating the IR image as an additional fourth layer to the visible (RGB) image.

ii) Improving the classification accuracy of a simple 3-layer using the strategy in (i) to 86.8590% as against the 85.0610% and 67.9878% classification accuracies obtained from the visible and infrared modes respectively. This is despite the simplicity and the limited size of the training data.

iii) Fine-tuning a pre-trained model after feature extraction which yields an accuracy of 100% on a modest dataset by randomly selecting the training images from the entire dataset.

5.3 CONCLUSION

This research work implemented and described a set of comprehensive deep learning based object recognition frameworks using deep convolutional neural networks by jointly learning image representations from fused multimodal images of Visible-Light and Infrared (IR) images. Experimental results show that training a simple convolutional neural networks on a small dataset is suboptimal for object recognition when a single modality of the input image datasets is used. However, results show that the combined use of visible and IR imagery is a tremendously improves the recognition accuracy thereby enhancing the recognition performance of system as opposed to using a single image modality. Furthermore, fine-tuning standard pre-trained networks trained on large datasets and adopting them to a modest datasets achieves excellent results. All experiments in this research were implemented using Matlab programming language, as well as MatConvNet and Liblinear libraries for deeplearning and large linear classification respectively.

5.4 LIMITATIONS

Although the aim of the research work was achieved successfully, some of the limitations of the work are highlighted as follows:

i) The VAIS dataset has only 1088 available paired images. It is worthy to note that sometimes it is more critical to have the required quantity and quality of images in the dataset than to have the best algorithm

ii) Longer computation time on CPU of conventional systems. With increased depth of the inference (learning) layer, parallel computing systems may be required in order to optimize computation time.

5.5 RECOMMENDATIONS FOR FURTHER WORK

The following areas are recommended for further research:

i) Other fusion strategies or image registration techniques can be explored and investigated using same deep learning approaches.

ii) The network can be made deeper and performance investigated on suitable datasets under different scenarios such as with/without regularization, different batch sizes, increasing number of epochs etc.

iii) An investigative research can be carried out on post-training fusion where the contribution of each modality (visible or IR) in arriving at a decision is weighted and analysed

REFERENCES

Aleksander, I., & Morton, H. (1990). *An introduction to neural computing* (Vol. 240): Chapman and Hall London.

Bendada, A., & Akhloufi, M. A. (2010). *Multispectral face recognition in texture space.* Paper presented at the Canadian Conference on Computer and Robot Vision (CRV), 2010

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning, 2*(1), 1-127.

Chang, H., Koschan, A., Abidi, M., Kong, S. G., & Won, C.-H. (2008). *Multispectral visible and infrared imaging for face recognition.* IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08.

Chauvin, Y., & Rumelhart, D. E. (1995). *Backpropagation: theory, architectures, and applications*: Psychology Press.

Chen, C.-h., Pau, L.-F., & Wang, P. S.-p. (2010). *Handbook of pattern recognition and computer vision* (Vol. 27): World Scientific.

Deng, L., & Yu, D. (2014). Deep Learning. *Signal Processing, 7*, 3-4.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of machine learning research, 9*(Aug), 1871-1874.

Forsyth, D. A., & Ponce, J. (2003). A modern approach. *Computer Vision: A Modern Approach*, 88-101.

Gardner, M. W., & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, *32*(14), 2627-2636.

Han, J., & Moraga, C. (1995). *The influence of the sigmoid function parameters on the speed of backpropagation learning.* Paper presented at the International Workshop on Artificial Neural Networks.

Hariharan, H., Koschan, A., Abidi, B., Gribok, A., & Abidi, M. (2006). *Fusion of visible and infrared images using empirical mode decomposition to improve face recognition.* Paper presented at the Image Processing, 2006 IEEE International Conference.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.* Paper presented at the Proceedings of the IEEE International Conference on Computer Vision.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527-1554.

Howard, A. G. (2013). Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*.

Jaeger, H. (2002). *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*: GMD-Forschungszentrum Informationstechnik.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *IEEE computer*, *29*(3), 31-44.

Jordan, M., & Mitchell, T. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255-260.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., & Hudspeth, A. (2000). *Principles of neural science* (Vol. 4): McGraw-hill New York.

Kong, S. G., Heo, J., Boughorbel, F., Zheng, Y., Abidi, B. R., Koschan, A., . . . Abidi, M. A. (2007). Multiscale fusion of visible and thermal IR images for illumination-invariant face recognition. *International Journal of Computer Vision, 71*(2), 215-233.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks.* Paper presented at the Advances in neural information processing systems (NIPS).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521* (7553), 436-444. doi: 10.1038/nature14539

Ma, Z., Wen, J., Liu, Q., & Tuo, G. (2015). Near-infrared and visible light image fusion algorithm for face recognition. *Journal of Modern Optics, 62*(9), 745-753.

McClelland, J. L., Rumelhart, D. E., & Group, P. R. (1987). *Parallel distributed processing* (Vol. 2): MIT press Cambridge, MA.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*: Springer Science & Business Media.

Morris, T. (2004). *Computer vision and image processing*: Palgrave Macmillan.

O'Reilly, R. C., & Munakata, Y. (2000). *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*: MIT press.

Piella, G. (2003). A general framework for multiresolution image fusion: from pixels to regions. *Information fusion, 4*(4), 259-280.

Rosenberg, C. (2013). Improving photo search: A step across the semantic gap: blog.

Russell, S., Norvig, P., & Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, 25*, 27.

Sarle, W. S. (1994). Neural networks and statistical models.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85-117.

Sharma, V., Rai, S., & Dev, A. (2012). A comprehensive study of artificial neural networks. *India (International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10).*

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, S., Gyaourova, A., Bebis, G., & Pavlidis, I. (2004). *Infrared and visible image fusion for face recognition.* Paper presented at the Defense and Security.

Sonka, M., Hlavac, V., & Boyle, R. (2014). *Image processing, analysis, and machine vision*: Cengage Learning.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). *Going deeper with convolutions.* Paper presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). *Deepface: Closing the gap to human-level performance in face verification.* Paper presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Vedaldi, A., & Lenc, K. (2015). *Matconvnet: Convolutional neural networks for matlab.* Paper presented at the Proceedings of the 23rd ACM international conference on Multimedia.

Williams, D., & Hinton, G. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533-536.

Yu, D., & Deng, L. (2011). Deep learning and its applications to signal and information processing [exploratory dsp]. *IEEE Signal Processing Magazine, 28*(1), 145-154.

Yuan, J., Ni, B., & Kassim, A. A. (2014). Half-CNN: A General Framework for Whole-Image Regression. *arXiv preprint arXiv:1412.6885*.

Zhang, M. M., Choi, J., Daniilidis, K., Wolf, M. T., & Kanan, C. (2015). *VAIS: A dataset for recognizing maritime imagery in the visible and infrared spectrums*. 2015 IEEE Conference Paper presented at the Computer Vision and Pattern Recognition Workshops (CVPRW).

Zhou, Y., & Hecht-Nielsen, R. (1993). *Target recognition using multiple sensors.* Paper presented at the Neural Networks for Processing [1993] III. Proceedings of the 1993 IEEE-SP Workshop.

APPENDIX A

An Implementation of Backpropagation Algorithm Using Sigmoid

For convenience of description let us adopt and define the following notation:

x = *training* data input

x_j^l = input to node *j* of layer *l*

x_k^L = linear combination of all the inputs of node *j* in the layer *L* with the weights

W_{ij}^l = weight from node *i* of layer *l*-1 to the node *j* of layer *l*

θ_j^l = bias of node j of layer *l*

y_j^l = output of node *j* of layer *l*

d_j = desired output / ground truth of the training data

$\sigma(\cdot)$ = an activation function (e.g. sigmoid function here)

L = total number of the layers

y_j^L = output of the neural network that corresponds to input *x*

The accuracy is measured in terms of the error *E* which is the difference of the observed output y_j^L and the actual or desired output d_j. The Weights are changed in order to minimize the overall mean square error defined by the cost function given below:

$$E = \frac{1}{2}\sum_{j \in L} \| d_j - y_j^L \|^2 \qquad (1)$$

The goal is to find the minimum and so the partial derivatives of the cost function is computed with respect to the weights and set to zero. That is,

$$\frac{\partial E}{\partial W_{ij}^l} = 0 \qquad (2)$$

Now, two cases are considered: whether the node is an output or a hidden layer node. For the former, the derivative of the difference of the actual and the observed output is computed. We hence have,

$$\frac{\partial E}{\partial W_{jk}^L} = \frac{\partial}{\partial W_{jk}^L} \frac{1}{2} \sum_{j \in L} \left(d_j - y_j^L \right)^2$$
$$= \left(d_k - y_k^L \right) \frac{\partial}{\partial W_{jk}^L} y_k^L \qquad (3)$$

This last equation is based on chain rule of differentiation. Since, only the node *k* has weight W_{jk}^L, the other terms get zero after the differentiation is applied. y_k^L is the output of the activation function with sigmoid used here. The equation becomes therefore becomes:

$$\frac{\partial E}{\partial W_{jk}^L} = \left(d_k - y_k^L \right) \frac{\partial}{\partial W_{jk}^L} \sigma(x_k^L) \qquad (4)$$

The derivative of activation function (sigmoid) has an interesting form:

$$\begin{aligned}\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right)\\ &= \frac{e^{-x}}{(1+e^{-x})^2}\\ &= \frac{1+e^{-x}}{(1+e^{-x})^2} - \left(\frac{1}{1+e^{-x}}\right)^2\\ &= \sigma(x) - \sigma(x)^2\end{aligned} \tag{5}$$

The partial derivative therefore becomes:

$$\frac{\partial E}{\partial W_{jk}^L} = \; d_k - y_k^L \;\; \sigma(x_k^L)(1-\sigma(x_k^L))\frac{\partial}{\partial W_{jk}^L} x_k^L \tag{6}$$

But

$$x_k^L = \sum_{i \in L-1} W_{ik}^L y_i^{L-1}$$

Thus, (9) becomes:

$$\frac{\partial E}{\partial W_{jk}^L} = \; d_k - y_k^L \;\; \sigma(x_k^L)(1-\sigma(x_k^L))y_j^{L-1} \tag{7}$$

Note that $\partial E / \partial W_{jk}^L$ relates to y_j^{L-1} and not y_i^{L-1} where $i \neq j$. therefore, based on this equation, the relation between j node of L-1 layer and k node of L layer is found. Now defining a new notation

$$\delta_k = \; d_k - y_k^L \;\; \sigma(x_k^L)(1-\sigma(x_k^L))$$

to represent node k of L layer term. The equation becomes accordingly:

$$\frac{\partial E}{\partial W_{jk}^L} = \delta_k y_j^{L-1} \tag{8}$$

Now for the *l* hidden layer node, considering layer *L*-1, similar partial derivative is applied over weights on the cost function. We obtain

$$\frac{\partial E}{\partial W_{ij}^{l}} = \frac{\partial}{\partial W_{ij}^{l}} \frac{1}{2} \sum_{k \in L} \left(d_k - y_k^L \right)^2$$
$$= \sum_{k \in L} \left(d_k - y_k^L \right) \frac{\partial}{\partial W_{ij}^{l}} y_k^L \qquad (9)$$

Applying chain rule, we get:

$$\frac{\partial E}{\partial W_{ij}^{l}} = \sum_{k \in L} \left(d_k - y_k^L \right) \sigma(x_k^L)(1 - \sigma(x_k^L)) \frac{\partial}{\partial W_{ij}^{l}} x_k^L \qquad (10)$$

the last derivative term is modified by chain rule:

$$\frac{\partial E}{\partial W_{ij}^{l}} = \sum_{k \in L} \left(d_k - y_k^L \right) \sigma(x_k^L)(1 - \sigma(x_k^L)) \frac{\partial x_k^L}{\partial y_j^l} \frac{\partial y_j^l}{\partial W_{ij}^{l}}$$
$$= \sum_{k \in L} \left(d_k - y_k^L \right) \sigma(x_k^L)(1 - \sigma(x_k^L)) W_{jk} \frac{\partial y_j^l}{\partial W_{ij}^{l}} \qquad (11)$$

The 2nd line of (14) is obtained since the input of x_k^L is a linear combination of outputs of previous layer node with the weight. Now, we observe that the derivative term doesn't relate to *k* node of *L* layer. Chain rule is applied accordingly:

$$\frac{\partial E}{\partial W_{ij}^{l}} = \sum_{k \in L} \left(d_k - y_k^L \right) \sigma(x_k^L)(1 - \sigma(x_k^L)) W_{jk} \sigma(x_j^l)(1 - \sigma(x_j^l)) \frac{\partial x_j^l}{\partial W_{ij}^{l}}$$
$$= \sum_{k \in L} \left(d_k - y_k^L \right) \sigma(x_k^L)(1 - \sigma(x_k^L)) W_{jk} \sigma(x_j^l)(1 - \sigma(x_j^l)) y_i^{l-1}$$
$$= \sigma(x_j^l)(1 - \sigma(x_j^l)) y_i^{l-1} \sum_{k \in L} \delta_k W_{jk} \qquad (12)$$

Now, all terms besides the y_i^{l-1} can be defined to be δ_j. The equation therfore becomes:

$$\frac{\partial E}{\partial W_{ij}^{l}} = \delta_j y_i^{l-1} \tag{13}$$

Now, putting the results of these 2 cases together:

For an output layer node k : $\frac{\partial E}{\partial W_{jk}^{L}} = \delta_k y_j^{L-1}$, where $\delta_k = d_k - y_k^L \ \sigma(x_k)(1-\sigma(x_k))$

For a hidden layer node j: $\frac{\partial E}{\partial W_{ij}^{l}} = \delta_j y_i^{l-1}$, where $\delta_j = \sigma(x_j)(1-\sigma(x_j))\sum_{k\in L}\delta_k W_{jk}$

Similar process is applied also to the bias term. For instance, the partial derivation on bias of *k* node in the last layer L θ_k^L. We get:

$$\frac{\partial E}{\partial \theta_k^L} = d_k - y_k^L \ \sigma(x_k^L)(1-\sigma(x_k^L))\frac{\partial}{\partial \theta_k^L} x_k^L \tag{14}$$

Now since, $x_k^L = \theta_k^L + w_{1k}^L + \cdots + w_{nk}^L$, the last term is 1 and the equation can be updated to:

$$\frac{\partial E}{\partial \theta_k^L} = d_k - y_k^L \ \sigma(x_k^L)(1-\sigma(x_k^L)) \tag{19}$$

the gradient of the cost function over bias is hence given by:

$$\frac{\partial E}{\partial \theta_l} = d_k - y_k^l \ \sigma(x_k^l)(1-\sigma(x_k^l)) = \delta_l \tag{20}$$

The above relation holds for any particular layer *l* of interest.

APPENDIX B1

MATLAB CODES

Main M- File

```
function [net, info] = main_exp(imdb)
DATASET = 1; % set =1,2, or 3 for visible, IR and paired
respectively %run 'vl_setupnn.m';
run  vl_setupnn
%opts.expDir is where trained networks and plots are saved.
opts.expDir = fullfile('..','data','part1') ;
%opts.batchSize is the number of training images in each batch.
opts.batchSize = 50;
opts.learningRate = 0.0001;
opts.numEpochs = 50;
opts.continue = false;
% Thecnn_init function specifies the network architecture.
if DATASET==3
net = cnn_init(4); % d=1 IR.   d=3 EO.d=4 for paired
elseif DATASET==2
net = cnn_init(1);
else
net = cnn_init(3); % DATASET = 1 or 4
end
% Thesetup_data function loads the training and testing images
into MatConvNet'simdb structure.
ifnargin< 1
if DATASET == 3
imdb = VAIS_paired_imdb(1);
elseif DATASET == 4
imdb = VAIS_paired_imdb(0);
else
end
end
%% -------------------------------------------------------------
%                      Training...
% -------------------------------------------------------------
[net, info] = cnn_train(net, imdb, @getBatch, ...
opts, ...
    'val', find(imdb.images.set == 2)) ;
fprintf('Lowest validation erorr is
%f\n',min(info.val.error(1,:)))
```

```
end
% --------------------------------------------------------------------
function [im, labels] = getBatch(imdb, batch)
%getBatch is called by cnn_train.
%'imdb' is the image database.
%'batch' is the indices of the images chosen for this batch.
%'labels' indicates the ground truth category of each image.
% 'jitter' data the data
% --------------------------------------------------------------------
im = imdb.images.data(:,:,:,batch) ;
labels = imdb.images.labels(1,batch) ;
% Add jittering here before returning im
if rand() < 0.5
im = fliplr(im); %# horizontal flip
end
end
```

APPENDIX B2

Training from Scratch a 3 - Layer Network

```
function net = cnn_init(d)
ifnargin< 1
    d = 3;
end
rng('default');
rng(0);
f=1/100;
net.layers = {};
net.layers{end+1} = struct('type', 'conv', 'weights', ...
{{f*randn(3,3,d,10, 'single'), zeros(1, 10, 'single')}}, ...
'stride', 1, 'pad', 0, 'name', 'conv1') ;

net.layers{end+1} = struct('type', 'pool', 'method', 'max', ...
                           'pool', [4 4],'stride', 2,'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;
%added dropout layer
net.layers{end+1} = struct('type', 'dropout', 'rate', 0.5);
%new layer 2
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(5,5,10,14,
'single'), zeros(1, 14, 'single')}}, 'stride', 1, 'pad', 0, ...
                           'name', 'conv1') ;

net.layers{end+1} = struct('type', 'pool', 'method', 'max', ...
                           'pool', [3 3], 'stride', 2, ...
                           'pad', 0) ;
```

```
net.layers{end+1} = struct('type', 'relu') ;
%added dropout layer
net.layers{end+1} = struct('type', 'dropout', ...
                           'rate', 0.5);


net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(12,12,14,6,
'single'), zeros(1, 6, 'single')}}, ...
                           'stride', 1, ...
                           'pad', 0, ...
                           'name', 'fc1') ;
% Loss layer
net.layers{end+1} = struct('name', 'hinge loss', 'type', 'loss',
'loss', 'mhinge'); %struct('type', 'softmaxloss') ;

% Visualize the network
vl_simplenn_display(net, 'inputSize', [64 64 d 50])
```

APPENDIX B3

Fine-tuning a Pretrained Model

```
function net = cnn_init()
ifnargin< 1
    d = 3;
end
net = load('imagenet-vgg-f.mat') ;
% We'll make some modifications to this network to accept a 4th
layer
numfilt = size(net.layers{1}.weights{1}, 4);
fori = 1:numfilt
net.layers{1}.weights{1}(:,:,4,i) = mean
(net.layers{1}.weights{1}(:,:,1:3,i), 3); end
%accepts
f=1/100;
foric = 1:length(net.layers)
ifstrcmp(net.layers{ic}.type,'lrn')
net.layers{ic}.type = 'normalize';
end
end
net.layers{end-1} = struct('type', 'dropout','rate',0.5) ;
net.layers{end} = struct('type', 'conv', ...
                           'weights', {{f*randn(1,1,4096,6,
'single'), zeros(1, 6, 'single')}},   'stride', 1, ...
                           'pad', 0, 'name', 'fc8') ;
```

```
% Loss layer
net.layers{end+1} = struct('type', 'softmaxloss') ;
vl_simplenn_display(net, 'inputSize', [64 64 3 50])
vl_simplenn_display(net, 'inputSize', [64 64 3 50])
```

APPENDIX B4

Testing the Trained Network

```
clear,clc;

load('net.mat');
% run vl_compilenn
runvl_setupnn
net.layers{end}.type = 'softmax';
image_size = [64 64];
im = imread('mo.png') ; % test image
% im = rgb2gray(im);
im_ = single(im) ; % note: 0-255 range
im_ =imresize(im_,image_size);
im_ = im_ - mean(im_(:));
% run the CNN
res = vl_simplenn(net, im_, [], [], 'disableDropout', 1) ;
% show the classification result
scores = squeeze(gather(res(end).x)) ;
[bestScore, best] = max(scores) ;
figure(1) ; clf ; imshow(im) ;
title(sprintf('The class is %d, score %.1f%%',best, bestScore *
100)) ;
```

APPENDIX B5

Feature Extraction + Classifier (Logistic Regression)

```
%The following code was applied to each of the train/test and
clc;
make% Install Liblinear LIBRARY
%Load train/test features/labels
load ('Train_Labels');
load ('Test_Labels');
load('Train_Features_EO.mat');
load('Test_Features_EO.mat');
load('exist_EO_test');
% Convert feature matrix to sparse
Features_sparse= sparse(Train_Features_EO);
Test_Features_sparse= sparse(Test_Features_EO);
% Training a Logistic Regression Classifier ('-s 0')
model = train(Train_Labels, Features_sparse ,['-s 0', 'col']);
% Predict
```

```
Test_Label_New = Test_Labels(exist==1);
Test_Features_sparse= Test_Features_sparse(exist==1,:);
[predicted_label, accuracy, prob_estimates] =
predict(Test_Label_New, Test_Features_sparse, model, '-b 1');
```

APPENDIX B6

Set Image Dataset

```
functionimdb = VAIS_paired_imdb(PAIRED)
ifnargin< 1
    PAIRED = 0; % if 0, only visible. if 1, both (4 layers)
end
total_images = 1088; % IR: 1242
image_size = [64 64]; %downsampling data for speed and because
it hurts
% accuracy surprisingly little. NB. USE [64 64] for scratch and
[224 224]
% for finetuning
imdb.images.data   = zeros(image_size(1), image_size(2),
3+PAIRED, total_images, 'single');
imdb.images.labels = zeros(1, total_images, 'single');
imdb.images.set    = zeros(1, total_images, 'uint8');
cd 'VAIS';
%% prepare paired images
% load vais
[EO_Train, IR_Train, Train_Labels, Train_Night, EO_Test,
IR_Test, Test_Labels, Test_Night, Class_Names] = Load_VAIS('.');
fname = 'annotations.txt';
root = 'C:\Users\yusuf\code\VAIS';
fid = fopen(fullfile(root, fname), 'rt');
C = textscan(fid, '%s %s %s %s %d %d %d');
fclose(fid);
EO_Files = C{1};
IR_Files = C{2};
Basic_Labels = C{4};
%% get the paired images
total = 0;
EO_paired = {};
IR_paired = {};
for k = 1:length( EO_Files)
if ~strcmpi( EO_Files{k}, 'null') && ~strcmpi( IR_Files {k},
'null')
total = total + 1;
EO_paired {k}=EO_Files{k};
IR_paired {k}=IR_Files {k};
end
end
```

```
%% identify the empty entries
DATALIST = IR_paired(:);
exist = ones(length(DATALIST),1);
forimno = 1:length(DATALIST)
ifisempty(DATALIST{imno})
exist(imno) = 0;
end
end
%% obtain the required image sets
EO_paired=  EO_paired (exist==1)';
IR_paired=  IR_paired (exist==1)';
Paired_Labels = Basic_Labels (exist==1);
 %% convert paired_Label to standard numeric class
paired_y = zeros (length(Paired_Labels),1);
for m = 1:length( Paired_Labels)
if  strcmpi(Paired_Labels{m}, 'cargo')
paired_y(m)=1;
elseifstrcmpi(Paired_Labels{m}, 'medium-other')
paired_y(m)=2;
elseifstrcmpi(Paired_Labels{m}, 'passenger')
paired_y(m)=3;
elseifstrcmpi(Paired_Labels{m}, 'sailing')
paired_y(m)=4;
elseifstrcmpi(Paired_Labels{m}, 'small')
paired_y(m)=5;
elseifstrcmpi(Paired_Labels{m}, 'tug')
paired_y(m)=6;
end
end

%% Load images
forimno = 1:length(EO_paired)
    im1 = imread(EO_paired{imno});
    im1 = single(im1);
    im1 = imresize(im1,image_size);
    im1 = im1 - mean(im1(:));
    im2 = imread(IR_paired{imno});
    % farkonlaushin da nakara
im_ir = cat(3,im2,im2,im2);
im_ir = single (im_ir);
im_ir = imresize(im_ir,image_size);
im_ir = im_ir - mean(im_ir(:));
    % karshenlaushin da nakara
    im2 = single(im2);
    im2 = imresize(im2,image_size);
    im2 = im2 - mean(im2(:));
im_ = im1;
```

```
im_(:,:,4) = im2;

if PAIRED==1
imdb.images.data(:,:,:,imno) = im_; %PAIRED
else
imdb.images.data(:,:,:,imno) = im1; %VISIBLE
        %imdb.images.data(:,:,:,imno) = im_ir; % IR
end
imdb.images.labels(  1,imno) = paired_y(imno);
imdb.images.set(     1,imno) = (rand > 0.7)+1; %1 for train, 2
for test (val?)
    %imdb.images.set(     1,imno) = mod(imno,2)+1; %1 for train,
2 for test (val?)
end
cd '..';
```

APPENDIX B7

Code that Implements SGD for Training CNNs

```
function [net, info] = cnn_train(net, imdb, getBatch, varargin)
opts.batchSizes = 50;
opts.numSubBatch = 1;
opts.training = [];
opts.val = [];
opts.numEpochs = 300;
opts.gpus = [];
opts.learning_Rate =0.001;
opts.continue = false;
opts.expDir = fullfile('data','exp') ;
opts.conserve_Memory = false;
opts.backPropDepth = +inf ;
opts.sync = false;
opts.prefetch = false;
opts.cudnn = true;
opts.weightDecay = 0.0005 ;
opts.momentum = 0.9;
opts.errorFunction = 'multiclass';
opts.errorLabels={};
opts.plotDiagnostics = false;
opts.memoryMapFile = fullfile(tempdir, 'matconvnet.bin') ;
opts = vl_argparse(opts, varargin) ;

if ~exist(opts.expDir, 'dir'), mkdir(opts.expDir) ; end
ifisempty(opts.train), opts.train = find(imdb.images.set==1) ;
end
ifisempty(opts.val), opts.val = find(imdb.images.set==2) ; end
ifisnan(opts.train), opts.train = [] ; end
```

```
% Network initialization
evaluateMode = isempty(opts.train) ;
if ~evaluateMode
fori=1:numel(net.layers)
ifisfield(net.layers{i}, 'weights')
      J = numel(net.layers{i}.weights) ;
for j=1:J
net.layers{i}.momentum{j} =
zeros(size(net.layers{i}.weights{j}), 'single') ;
end
if ~isfield(net.layers{i}, 'learningRate')
net.layers{i}.learningRate = ones(1, J, 'single') ;
end
if ~isfield(net.layers{i}, 'weightDecay')
net.layers{i}.weightDecay = ones(1, J, 'single') ;
end
end
ifisfield(net.layers{i}, 'filters')
net.layers{i}.momentum{1} = zeros(size(net.layers{i}.filters),
'single') ;
net.layers{i}.momentum{2} = zeros(size(net.layers{i}.biases),
'single') ;
if ~isfield(net.layers{i}, 'learningRate')
net.layers{i}.learningRate = ones(1, 2, 'single') ;
end
if ~isfield(net.layers{i}, 'weightDecay')
net.layers{i}.weightDecay = single([1 0]) ;
end
end
end
end
% setup GPUs
numGpus = numel(opts.gpus) ;
ifnumGpus> 1
ifisempty(gcp('nocreate')),
parpool('local',numGpus) ;
spmd, gpuDevice(opts.gpus(labindex)), end
end
elseifnumGpus == 1
gpuDevice(opts.gpus)
end
if exist(opts.memoryMapFile), delete(opts.memoryMapFile) ; end
 % setup error calculation function
ifisstr(opts.errorFunction)
switchopts.errorFunction
case 'none'
```

```
opts.errorFunction = @error_none ;
case 'multiclass'
opts.errorFunction = @error_multiclass ;
ifisempty(opts.errorLabels), opts.errorLabels = {'top1e',
'top5e'} ; end
case 'binary'
opts.errorFunction = @error_binary ;
ifisempty(opts.errorLabels), opts.errorLabels = {'bine'} ; end
otherwise
error('Uknown error function ''%s''', opts.errorFunction);
end
end
 % Train and validate
modelPath = @(ep) fullfile(opts.expDir, sprintf('net-epoch-
%d.mat', ep));
modelFigPath = fullfile(opts.expDir, 'net-train.pdf') ;
start = opts.continue * findLastCheckpoint(opts.expDir) ;
if start >= 1
fprintf('resuming by loading epoch %d\n', start) ;
load(modelPath(start), 'net', 'info') ;
end
for epoch=start+1:opts.numEpochs
figure(2) ; clf ; colormap gray ;
  %vl_imarraysc(squeeze(net.layers{1}.weights{1}),'spacing',2)
axis equal ; title('filters in the first layer') ;
  % train one epoch and validate
learningRate = opts.learningRate(min(epoch,
numel(opts.learningRate))) ;
train = opts.train(randperm(numel(opts.train))) ; % shuffle
val = opts.val ;
ifnumGpus<= 1
    [net,stats.train] = process_epoch(opts, getBatch, epoch,
train, learningRate, imdb, net) ;
    [~,stats.val] = process_epoch(opts, getBatch, epoch, val, 0,
imdb, net) ;
else
spmd(numGpus)
      [net_, stats_train_] = process_epoch(opts, getBatch,
epoch, train, learningRate, imdb, net) ;
      [~, stats_val_] = process_epoch(opts, getBatch, epoch,
val, 0, imdb, net_) ;
end
net = net_{1} ;
stats.train = sum([stats_train_{:}],2) ;
stats.val = sum([stats_val_{:}],2) ;
end
  % save
```

```
ifevaluateMode, sets = {'val'} ; else sets = {'train', 'val'} ;
end
for f = sets
    f = char(f) ;
    n = numel(eval(f)) ;
    info.(f).speed(epoch) = n / stats.(f)(1) * max(1, numGpus) ;
    info.(f).objective(epoch) = stats.(f)(2) / n ;
    info.(f).error(:,epoch) = stats.(f)(3:end) / n ;
end
if(mod(epoch, 10)==0)
if ~evaluateMode, save(modelPath(epoch), 'net', 'info') ; end
end
figure(1) ; clf ;
hasError = isa(opts.errorFunction, 'function_handle') ;
subplot(1,1+hasError,1) ;
if ~evaluateMode
semilogy(1:epoch, info.train.objective, '.-', 'linewidth', 2) ;
hold on ;
end
semilogy(1:epoch, info.val.objective, '.--') ;
xlabel('training epoch') ; ylabel('energy') ;
grid on ;
  h=legend(sets) ;
set(h,'color','none');
title('objective') ;
ifhasError
subplot(1,2,2) ; leg = {} ;
if ~evaluateMode
plot(1:epoch, info.train.error', '.-', 'linewidth', 2) ;
hold on ;
leg = horzcat(leg, strcat('train ', opts.errorLabels)) ;
end
plot(1:epoch, info.val.error', '.--') ;
leg = horzcat(leg, strcat('val ', opts.errorLabels)) ;
set(legend(leg{:}),'color','none') ;
grid on ;
xlabel('training epoch') ; ylabel('error') ;
title('error') ;
end
drawnow ;
print(1, modelFigPath, '-dpdf') ;
end
% --------------------------------------------------------------------
function err = error_multiclass(opts, labels, res)
% --------------------------------------------------------------------
predictions = gather(res(end-1).x) ;
[~,predictions] = sort(predictions, 3, 'descend');
```

```
% be resilient to badly formatted labels
ifnumel(labels) == size(predictions, 4)
labels = reshape(labels,1,1,1,[]) ;
end
% skip null labels
mass = single(labels(:,:,1,:) > 0) ;
if size(labels,3) == 2
  % if there is a second channel in labels, used it as weights
mass = mass .* labels(:,:,2,:) ;
labels(:,:,2,:) = [] ;
end

error = ~bsxfun(@eq, predictions, labels) ;
err(1,1) = sum(sum(sum(mass .* error(:,:,1,:)))) ;
err(2,1) = sum(sum(sum(mass .* min(error(:,:,1:5,:),[],3)))) ;

% --------------------------------------------------------------------
function err = error_binaryclass(opts, labels, res)
% --------------------------------------------------------------------
predictions = gather(res(end-1).x) ;
error = bsxfun(@times, predictions, labels) < 0 ;
err = sum(error(:)) ;

% --------------------------------------------------------------------
function err = error_none(opts, labels, res)
% --------------------------------------------------------------------
err = zeros(0,1) ;
function  [net_cpu,stats,prof] = process_epoch(opts, getBatch,
epoch, subset, learningRate, imdb, net_cpu)
% --------------------------------------------------------------------
% move CNN to GPU as needed
numGpus = numel(opts.gpus) ;
ifnumGpus>= 1
net = vl_simplenn_move(net_cpu, 'gpu') ;
else
net = net_cpu ;
net_cpu = [] ;
end
% validation mode if learning rate is zero
training = learningRate> 0 ;
if training, mode = 'training' ; else, mode = 'validation' ; end
ifnargout> 2, mpiprofile on ; end

numGpus = numel(opts.gpus) ;
ifnumGpus>= 1
one = gpuArray(single(1)) ;
else
```

```
one = single(1) ;
end
res = [] ;mmap = [] ;stats = [] ;start = tic ;

for t=1:opts.batchSize:numel(subset)
fprintf('%s: epoch %02d: batch %3d/%3d: ', mode, epoch, ...
fix(t/opts.batchSize)+1, ceil(numel(subset)/opts.batchSize)) ;
batchSize = min(opts.batchSize, numel(subset) - t + 1) ;
numDone = 0 ;
error = [] ;
for s=1:opts.numSubBatches
    % get this image batch and prefetch the next
batchStart = t + (labindex-1) + (s-1) * numlabs ;
batchEnd = min(t+opts.batchSize-1, numel(subset)) ;
batch = subset(batchStart : opts.numSubBatches * numlabs :
batchEnd) ;
    [im, labels] = getBatch(imdb, batch) ;
ifopts.prefetch
if s==opts.numSubBatches
batchStart = t + (labindex-1) + opts.batchSize ;
batchEnd = min(t+2*opts.batchSize-1, numel(subset)) ;
else
batchStart = batchStart + numlabs ;
end
nextBatch = subset(batchStart : opts.numSubBatches * numlabs :
batchEnd) ;
getBatch(imdb, nextBatch) ;
end

ifnumGpus>= 1
im = gpuArray(im) ;
end

    % evaluate CNN
net.layers{end}.class = labels ;
if training, dzdy = one; else, dzdy = [] ; end
res = vl_simplenn(net, im, dzdy, res, ...
                      'accumulate', s ~= 1, ...
                      'disableDropout', ~training, ...
                      'conserveMemory', opts.conserveMemory, ...
                      'backPropDepth', opts.backPropDepth, ...
                      'sync', opts.sync, ...
                      'cudnn', opts.cudnn) ;

    % accumulate training errors
error = sum([error, [...
sum(double(gather(res(end).x))) ;
```

```
reshape(opts.errorFunction(opts, labels, res),[],1) ; ]],2) ;
numDone = numDone + numel(batch) ;
end

  % gather and accumulate gradients across labs
if training
ifnumGpus<= 1
      [net,res] = accumulate_gradients(opts, learningRate,
batchSize, net, res) ;
else
ifisempty(mmap)
mmap = map_gradients(opts.memoryMapFile, net, res, numGpus) ;
end
write_gradients(mmap, net, res) ;
labBarrier() ;
      [net,res] = accumulate_gradients(opts, learningRate,
batchSize, net, res, mmap) ;
end
end
   % print learning statistics
time = toc(start) ;
stats = sum([stats,[0 ; error]],2); % works even when stats=[]
stats(1) = time ;
  n = (t + batchSize - 1) / max(1,numlabs) ;
speed = n/time ;
fprintf('%.1f Hz%s\n', speed) ;

fprintf(' obj:%.3g', stats(2)/n) ;
fori=1:numel(opts.errorLabels)
fprintf(' %s:%.3g', opts.errorLabels{i}, stats(i+2)/n) ;
end
fprintf(' [%d/%d]', numDone, batchSize);
fprintf('\n') ;
 % debug info
ifopts.plotDiagnostics&&numGpus<= 1
figure(2) ; vl_simplenn_diagnose(net,res) ; drawnow ;
end
end
ifnargout> 2
prof = mpiprofile('info');
mpiprofile off ;
end

ifnumGpus>= 1
net_cpu = vl_simplenn_move(net, 'cpu') ;
else
net_cpu = net ;
```

```
end

% ------------------------------------------------------------
function [net,res] = accumulate_gradients(opts, lr, batchSize,
net, res, mmap)
% ------------------------------------------------------------
for l=numel(net.layers):-1:1
for j=1:numel(res(l).dzdw)
thisDecay = opts.weightDecay * net.layers{l}.weightDecay(j) ;
thisLR = lr * net.layers{l}.learningRate(j) ;

%     net.layers{l}
%     net.layers{l}.weights
%     res(l).dzdw
%     pause
% accumulate from multiple labs (GPUs) if needed
ifnargin>= 6
tag = sprintf('l%d_%d',l,j) ;
tmp = zeros(size(mmap.Data(labindex).(tag)), 'single') ;
for g = setdiff(1:numel(mmap.Data), labindex)
tmp = tmp + map.Data(g).(tag) ;
end
res(l).dzdw{j} = res(l).dzdw{j} + tmp ;
end

ifisfield(net.layers{l}, 'weights')
net.layers{l}.momentum{j} = ...
opts.momentum * net.layers{l}.momentum{j} ...
        - thisDecay * net.layers{l}.weights{j} ...
        - (1 / batchSize) * res(l).dzdw{j} ;
net.layers{l}.weights{j} = net.layers{l}.weights{j} + thisLR *
net.layers{l}.momentum{j} ;
else

if j == 1
net.layers{l}.momentum{j} = ...
opts.momentum * net.layers{l}.momentum{j} ...
          - thisDecay * net.layers{l}.filters ...
          - (1 / batchSize) * res(l).dzdw{j} ;
net.layers{l}.filters = net.layers{l}.filters + thisLR *
net.layers{l}.momentum{j} ;
else
net.layers{l}.momentum{j} = ...
opts.momentum * net.layers{l}.momentum{j} ...
          - thisDecay * net.layers{l}.biases ...
          - (1 / batchSize) * res(l).dzdw{j} ;
```

```
net.layers{l}.biases = net.layers{l}.biases + thisLR *
net.layers{l}.momentum{j} ;
end
end
end
end

% ------------------------------------------------------------------
function mmap = map_gradients(fname, net, res, numGpus)
% ------------------------------------------------------------------
format = {} ;
fori=1:numel(net.layers)
for j=1:numel(res(i).dzdw)
    format(end+1,1:3) = {'single', size(res(i).dzdw{j}),
sprintf('l%d_%d',i,j)} ;
end
end
format(end+1,1:3) = {'double', [3 1], 'errors'} ;
if ~exist(fname) && (labindex == 1)
  f = fopen(fname,'wb') ;
for g=1:numGpus
fori=1:size(format,1)
fwrite(f,zeros(format{i,2},format{i,1}),format{i,1}) ;
end
end
fclose(f) ;
end
labBarrier() ;
mmap = memmapfile(fname, 'Format', format, 'Repeat', numGpus,
'Writable', true) ;
% ------------------------------------------------------------------
functionwrite_gradients(mmap, net, res)
% ------------------------------------------------------------------
fori=1:numel(net.layers)
for j=1:numel(res(i).dzdw)
map.Data(labindex).(sprintf('l%d_%d',i,j)) =
gather(res(i).dzdw{j}) ;
end
end
function epoch = findLast-Checkpoint(modelDir)
list = dir(fullfile(modelDir, 'net-epoch*.mat')) ;
tokens = regexp({list.name}, 'net-epoch([\d]+).mat', 'tokens') ;
epoch = celfn(@(x) sscanf(x{1}{1},'%d'), tokens) ;
epoch = max([epoch 0]) ;
```