DEVELOPMENT OF NEW SOFT-COMPUTING BASED APPROACH FOR LANDMARK RECOGNITION

Thesis Submitted for the Award of the Degree of

DOCTOR OF PHILOSOPHY

in

Computer Applications

By

Kanishk Bansal

Registration Number: 11915015

Supervised By:

Dr. Amar Singh (23318)

Professor, School of Computer Applications, Lovely Professional University, Phagwara

LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

2023

DECLARATION

I, hereby declared that the presented work in the thesis entitled "Development of a new soft-computing based approach for landmark recognition" in fulfilment of degree of **Doctor of Philosophy (Ph. D.)** is outcome of research work carried out by me under the supervision of Prof. (Dr.) Amar Singh working as Professor, in the School of Computer Applications of Lovely Professional University, Punjab, India. In keeping with general practice of reporting scientific observations, due acknowledgements have been made whenever work described here has been based on findings of other investigator. This work has not been submitted in part or full to any other University or Institute for the award of any degree.

Kanishk Bansal

(Signature of Scholar)

Name of the scholar: Kanishk Bansal

Registration No.: 11915015

Department/school: School of Computer Applications

Lovely Professional University,

Punjab, India

CERTIFICATE

This is to certify that the work reported in the Ph. D. thesis entitled "Development of a new soft-computing based approach for landmark recognition" submitted in fulfillment of the requirement for the reward of degree of **Doctor of Philosophy (Ph.D.)** in the School of Computer Applications, is a research work carried out by Kanishk Bansal, Registration No. 11915015 , is bonafide record of his/her original work carried out under my supervision and that no part of thesis has been submitted for any other degree, diploma or equivalent course.

[illegible]

(Signature of Supervisor)

Name of supervisor: Prof. (Dr.) Amar Singh (UID: 23318)

Designation: Professor

Department/School: School of Computer Applications

University: Lovely Professional University, Punjab

ABBREVIATIONS

AI:	Artificial Intelligence
ML:	Machine Learning
DL:	Deep Learning
DNN:	Deep Neural Network
ANN:	Artificial Neural Network
CNN:	Convolutional Neural Network
LR:	Landmark Recognition
SC:	Soft Computing
NIC:	Nature Inspired Computing
PFA:	Paddy Field Algorithm
GA:	Genetic Algorithm
3PGA:	3 Parent Genetic Algorithm
BA:	Bat Algorithm
PBCOA:	Parallel Bat Colony Optimization Algorithm
NP:	Non-Polynomial
TSP:	Traveling Salesman Problem
CEC:	Congress on Evolutionary Computation
IEEE:	Institute of Electrical and Electronics Engineers

ABSTRACT

The thesis introduces us to the subject of Landmark Recognition in AI. Landmarks, in general, refer to any visible figure in a digital image that may be identified by a human eye. Landmark Recognition is of paramount importance in Artificial Intelligence (AI) since mostly the human eye focuses only on the tangible things visible to it. The tangible things in any digital image can mostly be classified as a Landmark. Landmark Recognition is a vast field and has to deal with extremely heavy sizes of datasets. Landmark classes also tend to be extremely huge because of their imperative nature. In this chapter, we have discussed the Machine Learning (ML) techniques applied to develop Artificial Intelligence models for Landmark Recognition. We have discussed the basic concepts of AI, ML, and DL (Deep Learning). We have discussed how DL models tend to be the best approach for the development of AI models. This thesis is devoted to exploring some new nature-inspired problem-solving approaches, modelling them into computerized programs, and applying these approaches to develop efficient machine-learning models for image classification.

The research work has covered an extensive literature survey of the Machine Learning (ML) techniques that have been applied to Landmark Recognition. We show how Landmark Recognition is a topic of paramount importance in AI and how the Landmarks datasets have emerged as one of the heaviest datasets available in the literature on Image Processing. The techniques that have been used for Landmark Recognition, to date, have been discussed. It is clear from the survey of literature that the best solution currently available for creating highly sophisticated models for landmark recognition tends to be a special class of Artificial Neural Networks (ANNs) called CNNs or Convolutional Neural Networks. It has further been shown how Nature Inspired Computing (NIC) techniques have been used for this purpose. ANNs are a special class of NIC techniques since they are inspired by the working of biological neurons in living organisms.

Also, we present three metaheuristic-based approaches for geographical landmark recognition. The three metaheuristics are namely Paddy Field Algorithm, Genetic Algorithm, and 3 Parent Genetic Algorithm. The metaheuristics are used to search for

the most optimal hyperparameters for the development of Convolutional Neural Networks for Geographical Landmark Recognition. The most optimal hyperparameters are searched for through a search space of more than a million hyperparametric combinations and thus specialized CNN architectures are evolved. The experiments were run successfully, and highly optimized CNN architectures were found to be evolving. The CNN hyperparameters highly different from those usually used were found to be highly competent. Kernels with shapes 5 x 5 and 7 x7 were found to perform better than 3 x 3 of some combinations. Relative accuracies of CNNs increased up to more than 40%, thereby indicating that the evolution of CNNs with evolutionary metaheuristics is highly desirable.

A Parallel Bat Colony Optimization Algorithm based on the introduction of colonies to the usual Bat Algorithm is also proposed. Bat Algorithm is a popular evolutionary metaheuristic that is used to find solutions to NP-Hard problems like the Travelling Salesman Problem (TSP). The concept of colonies was introduced in the regular Bat Algorithm along with phenomena like reproductive crossover and migration. The resultant Parallel Bat Colony Optimization Algorithm (PBCOA) was tested on the standard CEC Benchmark Functions, implemented in MATLAB, and the performance was compared to that of 17 other algorithms. On 6 test functions, the proposed algorithm showed the best performance in comparison with other algorithms. Additionally, it came out to be the sole best performer for CEC functions F9, F14, F15, F16, F17, F19, F22, F24, F25, and F27. The proposed approach performed the best overall across 17 compared algorithms in terms of finding the minimal cost to maximum benchmark CEC-14 functions.

The thesis proposes an evolutionary framework of CNNs with a Parallel Bat Colony Optimization Algorithm (PBCOA) over a geographical landmarks' dataset. We discuss how the application of the PBCOA metaheuristic to CNNs to evolve its hyperparameters works towards the evolution of better hyperparameters out of a search space of more than a million hyperparameters. We evolved initial weights, kernel frame size, number of kernels, and other such parameters. It was found that hyperparametric combinations highly different from the usually used ones seemed to perform better. Also, the performance of CNNs was evaluated using Accuracy and F1 score, and it was found that the evolved architectures performed much better in comparison to the unevolved architectures. The evolved CNN architectures seemed to perform better on the training sets as well as the test sets.

We also describe a comparative analysis of all the developed Soft-Computing approaches. We have discussed all the metaheuristics which were used to evolve CNN architectures and in addition to this, we have discussed the differences in datasets that were used to evolve the CNN architectures. We have discussed how the size of datasets was varied to maintain a healthy specialization and generalization ratio. We discuss the concept of data augmentation and how it was used to develop better datasets. Every ML technique ranging from classical ML techniques to evolved CNN architectures has been discussed. The evolved hyperparametric combinations were used on a landmarks' dataset with 200 classes and the performance of each architecture was measured using accuracy. It was concluded that the CNN hyperparameters evolved with PBCOA seemed to perform the best on the complete dataset.

We have discussed all the research work accomplished and published until now. We have shown how we progressed on the thesis entitled "Development of new Soft-Computing Based Approach for Landmark Recognition". All the publications with their web links have been provided. The list of conferences and attended workshops along with their certificates has also been provided.

ACKNOWLEDGEMENT

I would like to display my utmost gratitude to my supervisor Prof. (Dr.) Amar Singh whose support has been commendable throughout the degree and without whom this thesis could not have been possible. Then, I would like to grab the opportunity to thank all the administrative and teaching staff of the Lovely Professional University who work day and night to make LPU a highly functional and successful workplace.

I would also like to show my gratitude to my colleagues in the Lovely Professional University, who have consistently displayed a supportive environment and have helped me in all my ventures throughout the degree.

I would like to display my utmost gratitude to my family members including my parents, my wife, and my brother. They have provided me immense emotional and spiritual support and there is no doubt that without them today I might have not been writing this acknowledgment. It is said that you can find God in your parents and without a doubt, my parents have given me all the spiritual support required.

TABLE OF CONTENTS

Chapter 1: Introduction 17

1.1. Artificial Intelligence 17

1.2. Machine Learning 18

1.3. Landmark Recognition 18

1.4. Soft Computing 19

1.5. Machine Learning Working 21

1.6. Evaluation metrics for a successful model 21

1.7. Types of Machine Learning 22

1.8. Deep Learning 24

1.9. Artificial Neural Networks 26

1.10. Types of ANNs 26

1.11. Research Objectives 31

1.12. Problem Formulation 31

1.13. Organization of the Thesis 32

Chapter 2: Literature Survey 34

2.1. Introduction 34

2.2. Survey for Landmark Recognition 39

2.3. Survey of Classical Machine Learning Techniques 57

2.4. Survey of Convolutional Neural Networks: Performance Analysis 58

2.4.1. Layers in CNNs 61

2.4.2. Training CNNs 62

2.5. Research Gaps 67

2.6. Conclusion 68

Chapter 3: Geographical Landmark Recognition: Three Metaheuristic Based Approaches 69

3.1. Introduction 69

3.2. CNN Evolution with Paddy Field Algorithm 70

3.2.1. Dataset Pre-Processing 72

3.2.2. Experimentation and Results 73

3.3. CNN evolution with GA and 3PGA 80

3.3.1. CNN Architecture 82

3.3.2. 3 Parent Genetic Algorithm: The Concept 85

3.3.3. Proposed 3PGA-CNN Approach 86

3.3.4. Dataset Pre-Processing 87

3.3.5. Materials and Methodology 89

3.3.6. Experimentation 91

3.3.7. Results and Discussions 93

3.4. Conclusion 101

Chapter 4: Parallel Bat Colony Optimization Algorithm: An improved metaheuristic for global optimization 102

4.1. Introduction 102

4.2. Working example 107

4.3. CEC-14 Benchmark Functions 109

4.4. Results and Discussions 110

4.5. Conclusion 118

Chapter 5: Evolution of CNN over geographical landmark dataset using Parallel Bat Colony Optimization Algorithm 119

5.1. Introduction 119

5.2. Proposed Methodology and Dataset Pre-Processing 120

5.3. Results and Discussions 124

5.4. Conclusion 130

Chapter 6: Comparative Analysis of all the Proposed Approaches 132

6.1. Introduction 131

6.2. Classical Machine Learning Techniques vs CNN 133

6.3. Unevolved CNN vs CNNs evolved with Metaheuristics 136

6.4. Evaluation of CNNs over evolved Hyperparameters 139

6.5. PBCOA-CNN variation with dataset sizes 141

6.6. Conclusions 143

Chapter 7: Conclusions and Future Scope 144

7.1. Conclusions 144

7.2. Future Scope 148

References 150

List of Publications 171

List of Conferences 175

List of Workshops 177

LIST OF TABLES

Table 2.1. Landmark Recognition Literature Survey 40

Table 2.2. Accuracies on Google Landmarks Dataset V2 47

Table 2.3. Accuracies achieved by various teams 48

Table 2.4. Research articles with needs addressed 49

Table 2.5. Research articles for Instance Recognition 52

Table 2.6. Research articles for Robotic Technologies 54

Table 2.7. Research articles with optimizations 55

Table 2.8. Performance analysis of classical ML techniques 58

Table 2.9. Accuracies on a subset of Landmarks dataset 66

Table 3.1. Accuracies evolved with PFA 78

Table 3.2. Best-fit chromosomes in GA and 3PGA 94

Table 3.3. ML accuracies on the Landmarks dataset subset 94

Table 3.4. Training with GA and 3PGA 97

Table 4.1. Possible Decision Variables 105

Table 4.2. Fitness Costs of PBCOA over 30 CEC benchmark functions 111

Table 4.3. Comparative Performance of 18 algorithms 113

Table 4.4. Winning algorithms over 30 benchmark functions 116

Table 5.1. Ranges of evolved hyperparameters 123

Table 5.2. Commonly used hyperparameters 125

Table 5.3. Evolved best-fit architectures 126

Table 5.4. Training CNN with best-fit networks from BA and PBCOA 126

Table 5.5. Evolution of networks with BA and PBCOA 128

Table 6.1. Summary of accuracies for all ML Models 135

Table 6.2. Summary of accuracies from evolved networks...................................... 140

Table 6.3. Performance trends with sizes of datasets 141

LIST OF FIGURES

Figure 1.1. Soft Computing methodologies 19

Figure 1.2. Working of an ML model... 21

Figure 1.3. Architecture of a simple ANN.. 24

Figure 1.4. Types of ANNs 27

Figure 2.1. Landmarks in Google Landmarks Dataset V2 35

Figure 2.2. Landmarks from Weyand, Tobias et al. 38

Figure 2.3. Dataset Size Trends 45

Figure 2.4. Accuracy trends with dataset sizes... 46

Figure 2.5. Accuracies on Google Landmarks Dataset.. 47

Figure 2.6. Team accuracies on Google Landmarks Dataset V2................................. 48

Figure 2.7. Accuracies for various ML approaches... 58

Figure 2.8. Images of Qutub Minar and Alai Darwaza.. 59

Figure 2.9. Architecture of a simple CNN.. 59

Figure 2.10. Summary of the designed CNN.. 60

Figure 2.11. Depiction of convolution operation in CNNs.. 61

Figure 2.12. Depiction of Max-pooling operation in CNNs.. 62

Figure 2.13. Training process of a CNN.. 63

Figure 2.14. Accuracies for Landmark Recognition.. 65

Figure 3.1. Paddy Field Algorithm 70

Figure 3.2. Landmark images from Google Landmarks dataset V2............................ 73

Figure 3.3. Architecture of designed CNN 74

Figure 3.4. Accuracies elapsed as CNNs evolve with PFA... 76

Figure 3.5. Accuracies of evolved seeds 77

Figure 3.6. Accuracy comparison: Default vs Evolved 77

Figure 3.7. Confusion matrix for the trained model 79

Figure 3.8. Best-fit seed after evolution with PFA 80

Figure 3.9. A simple CNN architecture 83

Figure 3.10. Types of CNNs 84

Figure 3.11. Process behind 3 Parent Genetic Algorithm 86

Figure 3.12. Implementation of 3PGA 87

Figure 3.13. Images from landmarks dataset 88

Figure 3.14. Working of 3PGA 92

Figure 3.15. Typical chromosome in the population 92

Figure 3.16. Unevolved vs Evolved chromosomes 93

Figure 3.17. Best-fit chromosomes with accuracies 94

Figure 3.18. Best-fit chromosomes with GA 95

Figure 3.19. Accuracies of various ML techniques 95

Figure 3.20. Best accuracies from all techniques compared 96

Figure 3.21. Variation in accuracy with time 98

Figure 4.1. Bat Echolocation 102

Figure 4.2. Actual bat colonies 103

Figure 4.3. Improvements in Bat Algorithm 104

Figure 4.4. Representation of bat colonies 106

Figure 4.5. Winning functions of candidate algorithms 117

Figure 5.1. Architecture of 5 layered CNN 121

Figure 5.2. Evolution through BA and PBCOA 122

Figure 5.3. Evolved best-fit architectures 124

Figure 5.4. Evolution of networks with BA and PBCOA.. 125

Figure 5.5. Training progress as CNN trains with BA and PBCOA 127

Figure 5.6. Evolution performance as time elapses 128

Figure 6.1. Various ML techniques for Landmark Recognition................................ 133

Figure 6.2. Training process of ML model 134

Figure 6.3. Summary of accuracies for ML models 135

Figure 6.4. Number of classes in LR subset 138

Figure 6.5. Number of images in LR subset 138

Figure 6.6. Number of images after augmentation 138

Figure 6.7. Accuracies by evolved CNN architectures.. 140

Figure 6.8. Performance analysis over larger subsets.. 142

Chapter 1: Introduction

This chapter introduces the subject of Landmark Recognition in AI. Landmarks, in general, refer to any visible figure in a digital image that may be identified by a human eye. Landmark Recognition is of paramount importance in Artificial Intelligence (AI) since mostly the human eye focuses only on the tangible things visible to it. The tangible things in any digital image can mostly be classified as a Landmark. Landmark Recognition is a vast field and has to deal with extremely heavy sizes of datasets. Landmark classes also tend to be extremely huge because of their imperative nature. In this chapter, we have discussed the Machine Learning (ML) techniques applied to develop Artificial Intelligence models for Landmark Recognition. We have discussed the basic concepts of AI, ML, and DL (Deep Learning). We have discussed how DL models tend to be the best approach for the development of AI models. This thesis is devoted to exploring some new nature-inspired problem-solving approaches, modeling them into computerized programs, and applying these approaches to develop efficient machine-learning models for image classification.

1.1. Artificial Intelligence

In recent years, computing powers have been increasing manifolds in very short time intervals. It would not be wrong on our part to claim that the growth is exponential in nature. According to a law given by computer scientist and mathematician Gordon E. Moore, computer power doubles every two years. Though it was thought to have achieved a saturation point a few years ago, with the onset of Artificial Intelligence, we can safely claim that computer power is again increasing at an exponential rate.

Artificial Intelligence (AI) is a term coined by emeritus Stanford Professor John McCarthy in 1955, that was defined by him as “the science and engineering of making intelligent machines”. There are many computing devices in the market ranging from small calculators to supercomputers that can be intelligent. However, not all computing devices we see are intelligent. Smartphones are mostly intelligent. For example, the cameras of smartphones today can detect faces in a photograph. This is nothing but an example of an intelligent operating system of our smartphone.

Artificial Intelligence is a very broad area of computer sciences. One of the applications of Artificial Intelligence is Machine Learning. Machine Learning refers to the field of AI which deals with making intelligent machines. In other words, machines that are capable of doing work intelligently are dealt with in the area of Machine Learning. Machine Learning is an integral part of most of our lives, today.

1.2. Machine Learning

According to Tom Mitchell, “Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience.” Most of us, at least once in our lives, might have done, or at least been witness to, online shopping. On these online shopping websites or platforms, when we look out for any product we want or need, the website saves our preferences and shows us similar products in the future. This is nothing but an example of an artificially intelligent computer server or machine. Hence, in this example, we have made the machine learn our preferences. What is even better is that most of the time, our data is kept private using encryption mechanisms that do not allow any person except machines to process the data.

1.3. Landmark Recognition

The use of machine learning to identify landmarks in photographs has been the subject of much research [1]. An essential area of computer vision is landmark recognition. In this area, machine learning models are trained to find and recognise the closed, clearly identifiable objects in an image. A landmark is often believed to be included in that closed polygon produced by the pixels that may be considered to be a distinct and discernible entity in one sense or the other, if we consider a digital image to be a set of coordinates of various pixels [2].

Since landmark recognition is regarded as one of the foundational steps towards achieving a full computer vision, it is a crucial study issue in the field of image processing. Landmark Recognition is an important research area in the field of image classification since it is considered to be one of the first steps toward reaching a complete computer vision. An identifiable figure contained by a set of coordinates on an image is referred to as a landmark. Since they serve as the foundation for the discipline of image classification, landmarks are a crucial research area in the broad field of computer vision.

A landmark can be anything, from a structure to people to an X-ray of their brains. Yet, given the vast, intricate, and noisy datasets that are currently available in the literature, landmark recognition is a challenging task. The landmark image datasets that we receive typically have a lot of noise. Even though, since real-life images too are very noisy even at very high resolutions, noises are an essential component of image classification. The most common example of landmark recognition is the computerized verification of robots vs humans. Currently, only humans can distinguish between difficult landmark images.

1.4. Soft Computing

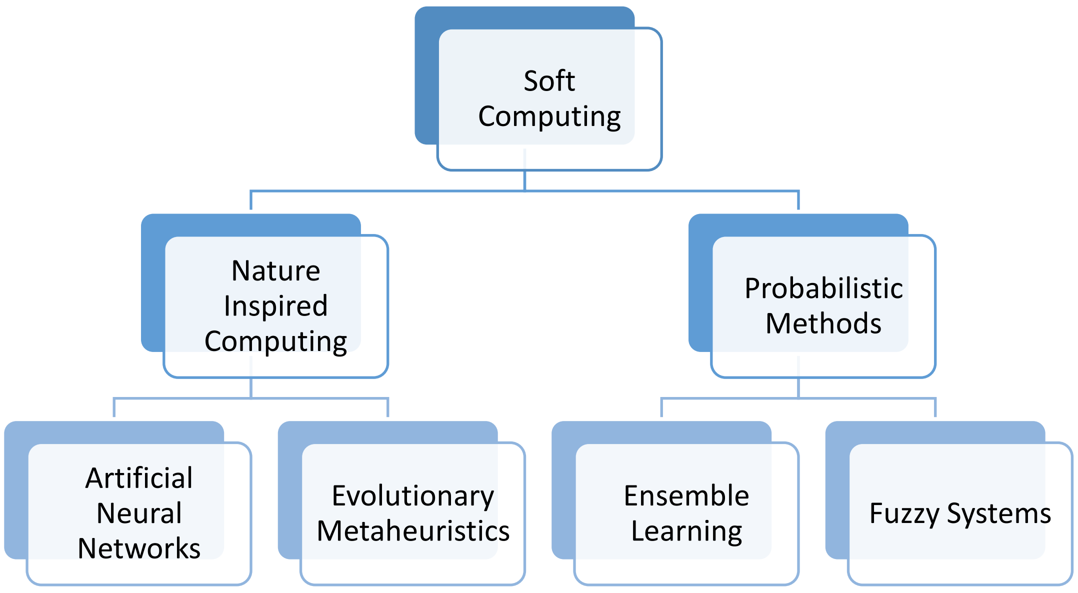

Soft Computing
Nature Inspired Computing
Probabilistic Methods
Artificial Neural Networks
Evolutionary Metaheuristics
Ensemble Learning
Fuzzy Systems

Figure 1.1: Soft computing methodologies.

The solution to NP-Hard problems is a million-dollar problem in computing. The Clay Institute of Mathematics in the United States has offered this reward-cum-appreciation to anyone who can tell whether NP is equal to P or not. All the current encryption mechanisms are based on this problem with the currently accepted answer as NP $\neq$ P.

This is arguable since the scientists remain divided over the issue till acceptable proof is provided.

This problem from the Clay Institute of Mathematics boils down to the answer to the query of whether any specified amount of looping can guarantee an exact answer to any mathematical problem using a computer system. There exist typical problems in computing like the Traveling Salesman Problem (TSP) that cannot be solved to the exact solution in some acceptable amount of time. A brute force logic for such problems can result in thousands of years for a computer program to run for getting the exact solution. Here, the concept of soft computing comes in.

Soft Computing is the science of the development of algorithms and techniques to find an optimal solution to NP-Hard problems instead of the exact solution. These algorithms tend to find highly useful answers to queries that can be difficult for a brute hard logic system. Routing and pattern finding are some of the applications where soft computing can help.

Figure 1.1 describes that soft computing broadly consists of nature-inspired-computing (NIC) approaches and probabilistic methods. Further, NIC approaches include Artificial Neural Networks (ANNs) and Evolutionary metaheuristics like Genetic Algorithms, Ant Colony Algorithm, Bat algorithms, etc. Parallelly, probabilistic methods include ensemble-based machine-learning approaches and fuzzy logic systems.

The main components of Soft Computing are:

i. **Artificial Neural Networks** – Artificial neural networks or ANNs are a Deep Learning paradigm that mimics human brain function to create artificially intelligent models for various tasks.
ii. **Evolutionary metaheuristics** – This soft computing paradigm handles search and optimization problems by modeling searches based on nature-inspired techniques. Examples include the Genetic Algorithm, Bat Algorithm, Ant Colony Algorithm, etc.
iii. **Fuzzy Logic** – Fuzzy logic is based on the probabilistic method of how decisions are taken. Instead of the hard brute logic, whereby the decision may

either be yes or no, the fuzzy logic presents itself such as: Mostly yes, probably yes, can't say, probably no, mostly no, etc.

iv. **Ensemble learning** – These methods are developed using statistical tools that include probability. Ensemble-based machine learning models exist that include regression, classification, clustering, etc.

1.5. Machine Learning Working

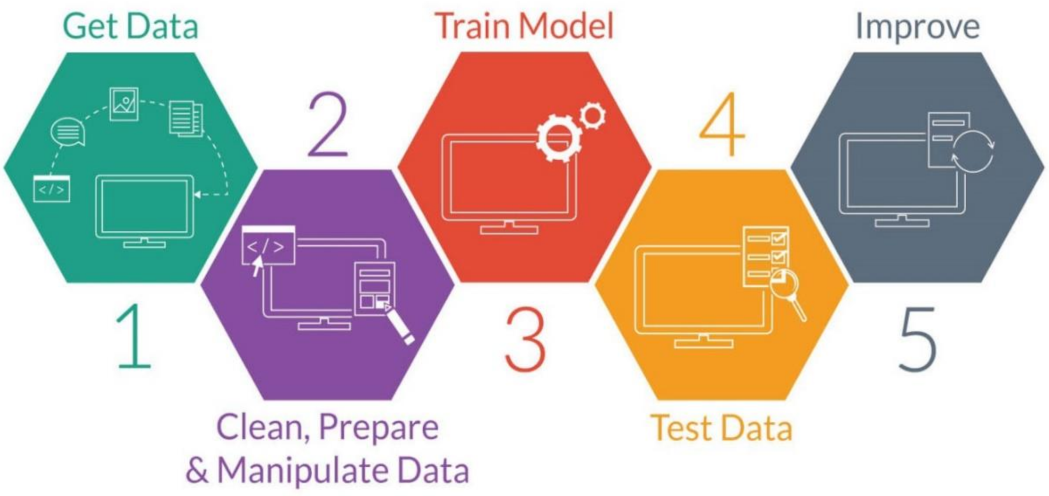


Figure 1.2: The designing and working of a machine learning model.

Machine Learning models are designed in many ways. However, some steps are similar. The common working steps are: getting relevant input data and processing the input data to get the training data. Split training data to test the validity of the trained ML model. Train an AI model from the training data. Save the machine learning model. Test the ML model on test data. Get evaluation results in terms of Accuracy, Precision, Recall, F1 score, etc. Compare training, test, and validation results. If performance is fine, pass the model. If performance is low, consider re-training with different hyperparameters.

1.6. Evaluation metrics for a successful model

For evaluating the performance of an AI model, we may use one of the many parameters based on TP - True Positive, TN - True Negative, FP - False Positive, and FN - False Negative. True and false refer to the true and false classification of a class and positive and negative represent two binary classes that can be extrapolated.

Some of the available performance metrics are Accuracy, Precision, Recall, and F1 score. These can be explained as follows:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}} \qquad \ldots(1.1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad \ldots(1.2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \qquad \ldots(1.3)$$

$$\text{F1 Score} = \frac{2*(\text{Recall} * \text{Precision})}{\text{Recall} + \text{Precision}} \qquad \ldots(1.4)$$

Accuracy is the most intuitive parameter for the evaluation of an AI model. Each parameter has its benefits while assessing performance. However, we mostly use accuracy to evaluate the model since it is the most acceptable metric for the evaluation of any AI model and its questioning is highly improbable.

1.7. Types of Machine Learning

There are mainly three types of machine learning namely:

a. Supervised Learning

b. Unsupervised Learning

c. Reinforcement Learning

a. Supervised Learning

Supervised learning is a sort of machine learning in which the output is predicted by the machines using well-labeled training data that has been used to train the machines. The term "labeled data" refers to input data that has already been assigned the appropriate output. In supervised learning, the training data that is given to the computers serve as the supervisor, instructing them on how to correctly predict the output. It employs the same idea that a pupil would learn under a teacher's guidance.

The method of supervised learning involves giving the machine learning model the right input data as well as the output data. Finding a mapping function to link the input variable (x) with the output variable is the goal of a supervised learning algorithm (y). Supervised learning has applications in the real world such as risk assessment, image categorization, fraud detection, spam filtering, etc.

Regression and Classification are the common types of Supervised Learning. The machine learning techniques that perform supervised learning include Decision Tree Classifier, Random Forest Classifier, K Nearest Neighbours (KNN), and Support Vector Machines (SVM). Most deep-learning techniques also fall under this category.

b. Unsupervised Learning

Unsupervised learning is a type of machine learning in which models are not supervised using training datasets, as the name implies. Instead, models themselves decipher the provided data to reveal hidden patterns and insights. It is comparable to the learning process that occurs in the human brain while learning something new.

Because, unlike supervised learning, we have the input data but no corresponding output data, unsupervised learning cannot be used to solve a regression or classification problem directly. Finding the underlying structure of a dataset, classifying the data based on similarities, and representing the dataset in a compressed format are the objectives of unsupervised learning. Clustering is the most common type of unsupervised learning. K means clustering is the best example of an unsupervised learning technique.

c. Reinforcement Learning

In reinforcement learning, the model learns by executing actions and observing the outcomes of those actions. Hence, an agent learns how to behave in a given environment via reinforcement learning, a feedback-based machine-learning technique. The agent receives a reward for each correct outcome and is penalized or given negative feedback for each negative action.

In contrast to supervised learning, reinforcement learning uses feedback to autonomously train the agent without the use of labeled data. The agent can only learn from their experience because there is no labeled data. A certain class of problems, such as those in robotics, gaming, and other time-consuming tasks, are solved using RL.

The agent engages with the environment and independently explores it. In reinforcement learning, an agent's main objective is to maximize the reward while doing better. The agent learns through hit-and-trial, and based on its experience, it trains itself to carry out the task more effectively.

1.8. Deep Learning

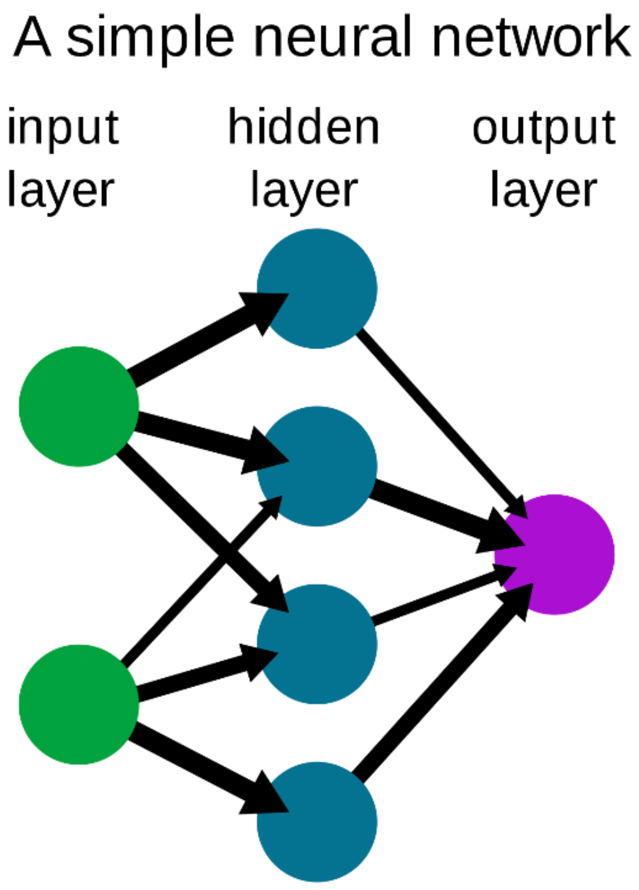


Figure 1.3: Architecture of a simple ANN.

Artificial Neural Networks (ANNs), also known as Deep Neural Networks (DNNs) serve as the foundation to the subset of Machine Learning (ML) paradigm known as Deep Learning (DL). The Artificial Neural Networks are structured similar to the neural networks existent in a human brain. They serve the purpose of learning patterns artificially from heavy volumes of data. This may be achieved in supervised, semi-supervised or non-supervised fashion.

ANNs perform the tasks like image identification, speech recognition, and natural language processing. Deep learning models must learn efficient patterns in the provided data to build reliable models. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Feedforward neural networks are the three most popular deep learning designs.

Although the ANNs employ a condensed set of ideas from biological brain systems, these networks mimic biological neural networks [3]. Particularly, ANN models mimic the electrical activity of the nerve system and brain. Since these are similar to different kinds of monitoring components are processing elements, they are also referred to as neurons or perceptrons. The perceptrons are typically stacked in layers or vectors, with the output from one layer acting as the input for the following layer and maybe others.

A neuron may be connected to every neuron in the following layer or just some of them, replicating the synaptic connections seen in the brain. Weighted data signals mimic information transit within a network or brain by simulating the electrical excitement of a nerve cell when it enters a neuron. A connection weight that mimics the strengthening of neural connections in the brain is multiplied by the input values to a processing element. Learning is simulated in ANNs by varying the connection strengths or weights.

The late nineteenth and early twentieth century witnessed the background research in the paradigm of Deep Learning related to Artificial Neural Networks. This mainly involved interdisciplinary study in the fields of neurophysiology, psychology, and physics. There were no detailed mathematical models of how neurons functioned in this early study; instead, it focused on generic theories of learning, perception, conditioning, etc. These fresh findings gave neural networks new life.

Over the past 20 years, a large number of articles have been published and various ANN types have been studied. Neural networks have been employed in a wide range of industries, including aerospace, automotives, banking, defence, electronics, entertainment, financial, insurance, manufacturing, medical, oil and gas, speech, securities, telecommunications, transportation, and environment.

1.9. Artificial Neural Networks

A neural network consists of three layers. The first layer, called the input layer. There are the input neurons that send information to the hidden layers. The hidden layers perform the calculations on the input data before sending the results to the output layer. The output layer includes the classes, labels and the data that must be linked to input in a regularised manner. Other things included in an ANN are weights, activation functions, and cost functions.

Weight, which is the numerical value, is the term used to describe the connection strength between neurons. The weights between neurons govern how well a neural network can learn. Artificial neural networks learn by adjusting the weights between the neurons.

The information is initially delivered into the input layer. Then, it is passed on to the hidden layers, where a subsequent interaction between these two layers first randomly chooses the weights for each input. After bias is added to each input neuron, the weight total—a combination of weights and bias—is then sent through the activation function. The activation function is responsible for selecting which node to fire for feature extraction and final output calculation.

Therefore, the term "forward propagation" is used to describe the entire process. After getting the output model to compare with the original output and knowing the error, weights are modified in backward propagation to reduce the error. This procedure lasts for a predetermined number of iterations known as epochs and finally, the model weights get updated so that predictions can be made.

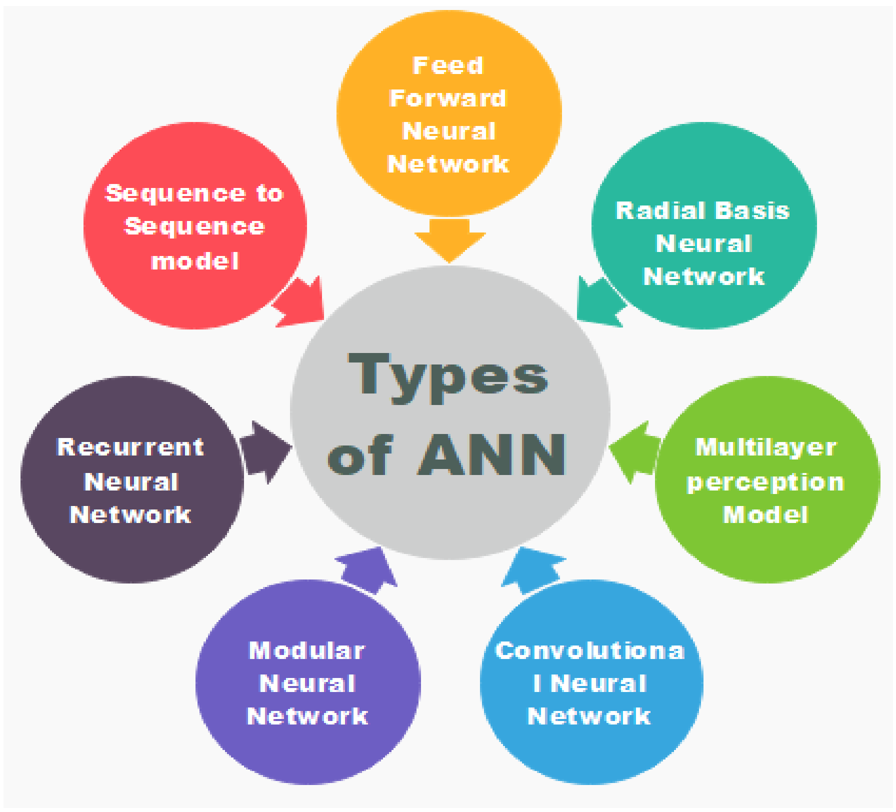
1.10. Types of ANNs

Artificial Neural Networks are constructed using a set of parameters and mathematical procedures that determine the output. Based on these criteria, let's examine a few neural networks:

1. Feedforward Neural Network – Artificial Neuron

2. Radial basis function Neural Network

3. Kohonen Self Organizing Neural Network
4. Recurrent Neural Network (RNN) – Long Short Term Memory
5. Convolutional Neural Network
6. Modular Neural Network



Feed Forward Neural Network
Sequence to Sequence model
Radial Basis Neural Network
Types of ANN
Recurrent Neural Network
Multilayer perception Model
Modular Neural Network
Convolutiona l Neural Network

Figure 1.4: Types of ANN

1. **Feedforward Neural Network – Artificial Neuron:** One of the most basic types of ANN, this neural network only transmits data in one direction. Data travels through input nodes before leaving on output nodes. The hidden layers of this neural network might or might not exist. Using a classifying activation function, it typically has a front propagated wave and no backpropagation.

2. **Radial basis function Neural Network:** The separation between a point and the center is taken into account by radial basis functions. RBF functions have two layers: the inner layer, which combines the Radial Basis Function and the features, and the outer layer, which works as a kind of memory by accounting for the output of the features while computing the same output in the next time-step.

3. **Kohonen Self Organizing Neural Network:** A Kohonen map's goal is to input vectors of any dimension to a discrete map made up of neurons. To organize the training data in a way unique to the map, it must be trained. Either one or two dimensions are present. The neuron's position is constant during training, but the weights change depending on the value. The first phase of this self-organization process involves initializing each neuron value with a tiny weight and the input vector.

4. **Recurrent Neural Network (RNN) – Long Short Term Memory:** The Recurrent Neural Network is based on the principle that by reusing a layer's output as input, it will be possible to predict the layer's outcome. The first layer is formed in this scenario similarly to the feed-forward neural network using the product of the sum of the weights and the features. When this is computed, the recurrent neural network process starts, which means that each neuron will carry over some of the knowledge it had in the previous time step from one time step to the next.

5. **Convolutional Neural Network:** Tens or even hundreds of layers can be present in a convolutional neural network, and each layer can be trained to recognise various aspects of an image. Each training image is subjected to filters at various resolutions, and the result of each convolved image is utilised as the input to the following layer. Beginning with relatively basic properties like brightness and borders, the filters can get more complicated until they reach characteristics that specifically identify the object.

6. **Modular Neural Network:** Multiple independent networks that work together to produce results make up modular neural networks. Each neural network's set of inputs is distinct from those of other networks building and carrying out subtasks. To complete the tasks, these networks do not communicate with one another or exchange signals.

ANNs are a network of many activation functions while data processing. Networks are trained in a manner so that the appropriate function gets activated from a function in the previous layer. The architecture includes many layers and performance is usually better with an increase in layers. This has made ANNs one of the best choices for Machine Learning (ML).

However, not all types of data can be handled by all types of ANNs. The digital images need to be handled by a type of ANNs called Convolutional Neural Networks (CNNs). A digital image contains 2 dimensions of pixelated information where each pixel itself contains one color out of millions of colors. An 8-bit RGB encoding scheme contains around 16M colors and each pixel contains one color out of these. To handle this data in tensor form CNNs were developed.

A typical CNN takes in parameters in the tensor form of:

$$x \in \mathbb{R}^{(C * W * H * n)} \qquad \ldots(1.5)$$

Where x designates the input in the network, C designates the color scheme in vector form, W defines the width of an image in pixels, H defines the height of the image and n is the number of images provided to the network.

When the network is trained, the activation functions that are mostly used are the Softmax activation functions:

$$\sigma(z)_i = \frac{e^{z_i}}{\Sigma_{j=1}^{k} e^{z_j}} \qquad \ldots(1.6)$$

where σ is the softmax activation function, z_i is the input vector, e^{z_i} is the standard exponential function for input vector, k is the number of classes in the multi-class classifier and e^{z_j} standard exponential function for output vector.

The network trains itself in the backpropagation phase where the information about correctness is fed back and appropriate measures have to be taken. These measures are known as in-built optimizers and currently there exist many optimizers like Gradient descent, Stochastic Gradient Descent, Adagrad, AdaDelta, and Adam. We have used AdaDelta optimization to improve the network. The equation for AdaDelta optimization is:

$$\eta_t' = \frac{\eta}{\sqrt{S_{dw_t+\varepsilon}}} \; where \; S_{dw_t} = \beta S_{dw_{t-1}} + (1-\beta)(\frac{\partial L}{\partial w_{t-1}})^2 \qquad \ldots(1.7)$$

Here η is the learning rate which changes with respect to S_{dw_t} which is the varied gradient descent, and it changes as the equation is shown on the right. Each layer L and weight w affect the optimization.

Optimization gives us a hint that more layers can give us more performance. However, it has been seen after a certain number of layers, the performances tend to remain the same or even worsen. For this purpose, various architectures of CNNs were developed which include ResNets (Residual Networks) [4], LeNets [5], GoogLeNets [6], VGGNets (Visual Geometry Group Networks) [7], Inception Nets [8], DELF Nets (Deep Local Feature Networks) [9], and many more.

In all these architectures, it was seen that as the sizes and variety of data increased, the techniques failed miserably due to their heavy resource consumption. It demanded a newer way of looking at the problem and today we have evolutionary computing to evolve the architectures of ANNs in an automated fashion. This is known as Neural Architecture Search (NAS).

Image Processing is extremely important in areas like Entertainment, Health, Defence Technologies, etc. In Defence, Artificial Intelligence (AI) is used for target identification, Computer Vision, etc. Computer Vision has become an indispensable necessity in the field of Artificial Intelligence and Computing. We see instances of Computer Vision almost daily. The AI cameras in our smartphones, and the identification of objects in an image, are all instances of Computer Vision. Large tech-giants like Google provide us with facilities like these, even for free.

Since AI works much like a living being trying to comprehend nature, we intend to use Nature-Inspired Computing Algorithms for the process of Computer Vision. Nature-Inspired Computing (NIC), as the name suggests, proposes algorithms inspired by natural phenomena, to achieve specific goals. We have used NIC algorithms for

Landmark Recognition in digital images. Landmark recognition is important for things ranging from recognizing places from old photographs to training robot vision etc. Google, the well-known provider of search engines, regularly announces prizes for the best accurate AI models for landmark recognition. We discovered from the literature review that the algorithms used for landmark recognition do not provide good accuracy for the emergence of increasingly large datasets. Hence, methods that can accurately execute landmark recognition while also being time-efficient are the need of the hour.

1.11. Problem Formulation

Landmark Recognition is an essential task in the field of Computer Vision. Landmark recognition is important for things like recognizing places from old photographs to training robot vision, etc. Image Processing is extremely important in the fields of medicine, defence, automobiles, etc. From the literature, we observed that the algorithms that have been applied to landmark recognition don’t give high accuracies for the increasingly vast datasets that are emerging. Landmark recognition datasets are highly noisy, unregulated, and complex even for a normal human to identify. Landmark recognition is performed mostly with the help of sophisticated Machine Learning paradigms known as Deep Neural Networks. Literature in the field of Soft Computing describes that Nature-Inspired-Computing including evolutionary algorithms help evolve the architectures of Artificial Neural Networks and this can further be applied for Landmark Recognition. These automated Deep Learning paradigms tend to outperform the regular unevolved ANNs on various evaluation metrics. Also, these are expected to outperform the best of ANNs developed till date. Hence there is a need for approaches that can accurately perform Landmark Recognition by exploring all the possible architectures of an AI model.

1.12. Research Objectives

We aimed to work on finding the best models for landmark recognition using soft computing algorithms and neural networks since these approaches are assumed to be both accurate and fast.

The agreed-upon objectives are underlined as follows:

a. To study, analyze and evaluate various classical and soft-computing-based algorithms applied to landmark recognition available in the literature.

b. To propose one nature-inspired-computing-based approach and evaluate its performance on standard benchmark problems.

c. To propose a new landmark recognition approach based upon proposed/existing nature-inspired computing algorithms.

d. To evaluate and compare the performances of the proposed landmark recognition approach with existing landmark recognition approaches using Python along with Google Colab.

1.13. Organization of the Thesis

The thesis has been organized into 7 chapters which can be underlined as follows:

- The first chapter introduces us to the research topic of Landmark Recognition in AI and the techniques that are used in the area. It introduces AI, ML, Soft Computing, DL, and Artificial Neural Networks
- The second chapter is an extensive survey of the Machine Learning and Deep Learning techniques that have been applied in the area of Landmark Recognition. The literature survey suggests that landmark recognition is an extremely broad research area with many possibilities for improvement.
- The third chapter presents the evolution of convolutional neural networks with three metaheuristics namely the Paddy Field Algorithm (PFA), Genetic Algorithm (GA), and 3 Parent Genetic Algorithm (3PGA).
- The fourth chapter describes the benchmarking of the metaheuristic proposed by us i.e. the Parallel Bat Colony Optimization Algorithm (PBCOA).
- The fifth chapter is about the application of the Parallel Bat Colony Optimization Algorithm (PBCOA) for evolving CNNs.
- The sixth chapter is the comparative analysis of all the approaches. A comparison is drawn between various approaches linked to landmark recognition and the important feats of PBCOA evolved CNN has been highlighted.

- The seventh chapter concludes the thesis with all the work that has been done till now. Our research works like the publications and other areas have been provided.

Chapter 2: Literature Survey

This chapter has covered an extensive literature survey of the Machine Learning (ML) techniques that have been applied to Landmark Recognition. We show how Landmark Recognition is a topic of paramount importance in AI and how landmark datasets have emerged as one of the heaviest datasets available in the literature on Image Processing. The techniques that have been used for Landmark Recognition, to date, have been discussed. It is clear from the survey of literature that the best solution currently available for creating highly sophisticated models for landmark recognition tends to be a special class of Artificial Neural Networks (ANNs) called CNNs or Convolutional Neural Networks. It has further been shown how Nature Inspired Computing (NIC) techniques have been used for this purpose. ANNs are a special class of NIC techniques since they are inspired by the working of biological neurons in living organisms.

2.1. Introduction

A human brain can correctly classify visually identifiable items known as landmarks in a digital image. Landmark recognition is a crucial subject in the science of computer vision since it deals with the fundamental capability of computers to recognize things in an image. A machine must first be trained on the most fundamental landmarks to recognize everything in an image. However, recognizing landmarks is not a simple task. The extensively used datasets in the field of landmark identification are not only very big, which harms training, but also exceedingly noisy and complex. Even the greatest photographs taken tend to have a lot of noise in the data after processing. To solve this issue, we trained a Convolutional Neural Network (CNN) on a dataset of geographical landmarks that was made available to the public. We discovered that the training and test accuracies of CNNs were higher than those of traditional machine learning techniques (98% and 73%, respectively). Additionally, it was observed that CNNs had reduced model overfitting.

A landmark in a digital image is defined as a distinguishable figure enclosed by a set of coordinates on the image. Landmarks are an important concept in computer vision because they set the basis for the field of Image Classification. A landmark in an image can be anything ranging from a building to humans to the X-ray image of their brains.

But landmark Recognition is currently a challenging task at hand due to the large, complex, and noisy available datasets. The datasets of images of landmarks that we get mostly include lots of noise. Noise is an integral part of image classification because real-life images are extremely noisy even at very high resolutions.

Landmark recognition is important because it is one of the first steps in imparting artificially intelligent (AI) systems a complete computer vision. It is also one of the first steps in giving robots some sort of vision. The current research is consistently trying to better the accuracy in the field of landmark recognition. If high accuracies are possible in this field, it could be very much possible that robotic systems can visualize and comprehend the normal world. Then they may also be able to make decisions that could be physical in nature. This also leads to another subject of cyber-physical systems. However, the current AI systems are not able to verify objects in an image accurately and many times this concept is used to verify humans vs. computers [10]. If high accuracies become possible, this could become a thing of the past soon.

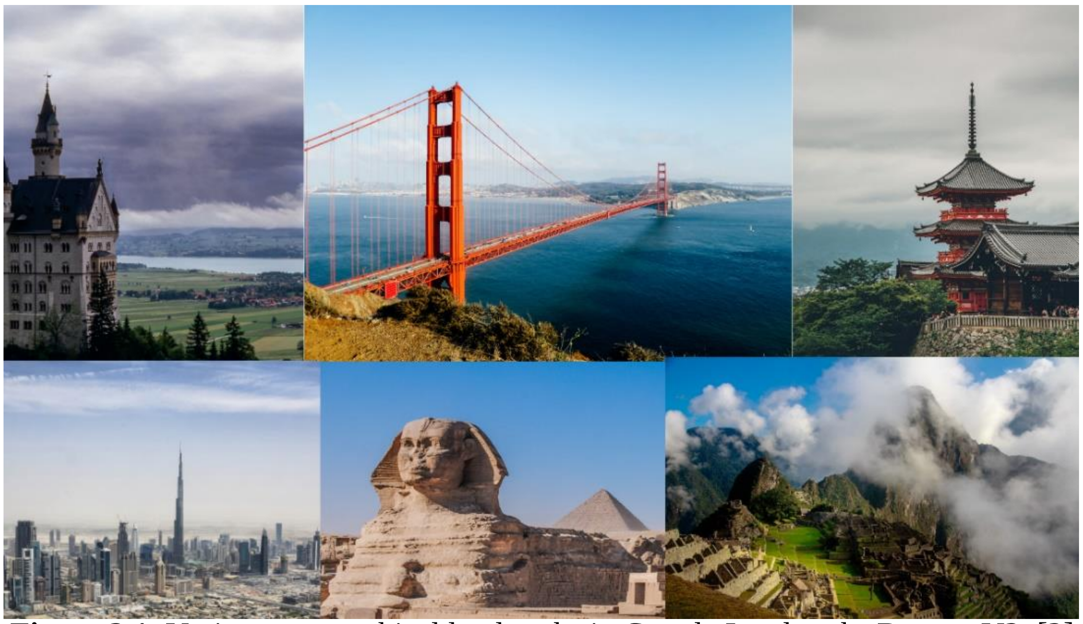


Figure 2.1: Various geographical landmarks in Google Landmarks Dataset V2. [2]

Artificial Intelligence (AI) has become an indispensable part of today's life. We use instances of AI in many spheres of everyday life. AI has also been termed the future of Computing and most computer systems will have to adapt to infuse the power of AI within them. The most common example of AI being used in everyday life is the cameras of our mobile phones being able to detect our faces and them being able to beautify our photographs accordingly [2], [10].

In the field of AI, computer vision is an integral part. Computer vision refers to the ability of a computer system being able to comprehend visual data in the form of digital images or videos and give results after processing them in novel ways. Image processing, deep fakes, etc. are examples of some of the topics included in computer vision [2].

We have focused on the task of image classification on geographical landmarks with the very popular convolutional neural networks (CNNs). For this task, we have taken a publicly available dataset from Kaggle. The geographical landmarks that we have used are the famous monuments from Delhi, India. We have recognized the famous geographical landmark places in images using different machine learning techniques of AI.

Many companies invest hefty amounts in this research area. Google regularly announces competitions on Kaggle (which is a famous website that organizes AI competitions) to reward those who come up with good techniques to recognize landmarks in an image [11]. Rewards of hefty amounts are given to those who come up with the best techniques for this task. Figure 1 shows some images from Google Landmarks Dataset V2.

It has been seen that as the sizes of datasets of landmark images increase, the current techniques for Landmark Recognition fail miserably [12]. Though high accuracies have been achieved on datasets of very large sizes, however, it is estimated that these techniques, too, will underperform when larger datasets come to the fore.

Landmark Recognition from images is a challenging issue in the field of Computer Vision due to the large and complex available datasets. Thus, we present a summary of different AI (Artificial Intelligence) based approaches for Landmark Recognition. To understand the scope of improvement, we have done a comparative analysis of the different approaches and datasets available in the State of the Art. From this study, we observed that the sizes and complexities of Landmark Recognition datasets are increasing with time. As the sizes are increasing, the accuracies of the available approaches are decreasing. In this paper, we observed that up to now, Google Landmarks Dataset V2 is the largest available dataset and the SE RestNetXT152 approach achieved the best training and validation accuracy on this dataset as compared to the other approaches. In the Future Scope section, we proposed that we can optimize the performance of different AI-based approaches for Landmark Recognition by using Nature Inspired Computing (NIC) based Search and Optimization Approaches.

Artificial Intelligence is the branch of computer science that deals with making computers efficient so that they can make good, well-informed decisions on their own. An artificially intelligent computer system learns from the existing knowledge very fast and can make decisions (sometimes) better than humans. AI learns from existing knowledge, builds models for a better understanding of a scenario, and delivers intelligent decisions. AI systems mostly work on statistical procedures to learn from the data provided to them in the form of knowledge, build models for a better understanding, and then deliver good, well-informed decisions based on the models formed from the data provided. For a good AI model, not only high-tech hardware is required but better AI models would have to be created. This requires both, good machinery and a good human brain for building the AI system. If developed aptly, AI models can outperform humans in delivering insightful answers and decisions.

Out of many uses of AI, one of the uses includes processing media. Recently, AI has been used to identify objects from an image. This area is known as Image Processing by AI. This area in AI is extremely important in the long run if we want machines to learn and understand the process of Visioning and comprehending what exists in the environment or even in a 2D image. In this, we can present an image to a computer system and ask it to identify what exists in that image. This field of Intelligence Systems is known as Computer Vision. An intelligent computer system can identify most of the objects in the image. A lot of research has been done and is ongoing in this field. AI systems can identify many things in photos, images, and even videos. These things include plants, fruits and vegetables, crops, animals, colors, and even human emotions.

Today, there exist many application software that can even process images and videos to the extent that the software recognizes any face in an image and uses it on some other face that exists in a video. This concept is known as Video Morphing. However, there are things which still most computer systems are not able to figure out. One of these is the important Landmarks in a picture which may include historical monuments, buildings, wonders of the world as well as simple immovable objects, etc. An AI system has been trained for single Landmarks but as dataset size increases, the AI systems also tend to suffer and fail.

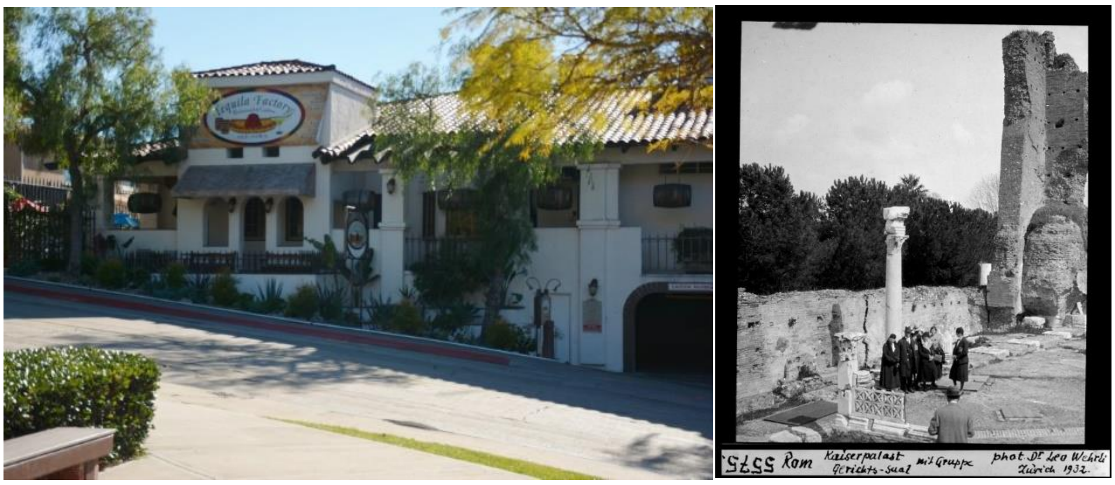


Figure 2.2: Examples of Landmarks from Weyand, Tobias, et al. (2015) [9]

However, Landmark Recognition is an important task in the field of Computer Vision because it is one of the first steps in enabling full-fledged Computer Vision. A properly enabled AI system will be able to identify almost everything with human-like precision or more. Work is ongoing to identify such Landmarks with high accuracy by a computer system. However, we still have a great deal to accomplish. A dataset of landmarks can be extremely large. Therefore, training a computer system to learn and identify these landmarks is a difficult task. A great deal of work has been already done in this area and a great deal more is required. Training accuracies of 90% and more have been achieved on small datasets but on larger ones, training accuracies don't cross 70% [13].

Google routinely announces new competitions to improve accuracy in this field [14]. However, as the datasets become more exhaustive, the out-of-the-sample accuracies are no more than 40%. We need much better techniques to handle such large datasets and improving validation accuracy i.e. accuracy for data outside training data, is needed to a very large extent [15]. The work in this field of Computer Vision is ongoing and a lot more is required yet. We, still, have a long way to reach an era when AI systems or robots will be able to identify everything in our surroundings just by Artificial Computer Vision.

A lot of research in Landmark Recognition focuses particularly on robot navigation only. Literature tells us how landmarks have been used continually for developing a Robot or Computer Vision. If we wish to see a machine being able to identify things with precision close to humans, training the AI systems on Landmarks shall be the first and foremost step. Landmarks are immovable or inanimate objects in our surroundings that do not usually change shapes too frequently. Due to such properties of Landmarks, AI systems, at a younger stage, should be trained with Landmarks only.

2.2. Survey for Landmark Recognition

Work on Landmark Recognition is progressing, and a lot of success has been accomplished. The research articles like Ozaki, Kohei, and Shuhei Yokoo. (2019), have tried to better the accuracies for large datasets available for Landmark Recognition [3]. The dataset used in the article is noisy and diverse. The approach in this research article is based on Deep Convolutional Neural Networks with Metric Learning, trained using cosine-softmax-based losses. These methods are sensitive to noise, and this could hinder in development of a reliable metric. To address the issue, the authors have developed an automated data-cleaning system. Similarly, more works that are ongoing, and which are accomplished have been discussed. The articles have been surveyed in terms of their datasets, approaches used, accomplishments, and the technologies they produced.

In this section, we survey trends in features of available datasets in the field of Landmark Recognition. As we can see in Table 2.1, in the article, Torii, Akihiko, et al. (2015), there was a dataset of just 125 landmarks and still, the accuracy was only 85% [16]. This is very less accurate considering the size of the dataset. In this research paper, 125 landmarks were used although the images totaled up to 1,000 out of which 315 were used as test images. Though this dataset was very small, however, the dataset in it could be used at any time meaning that the dataset was stable.

Table 2.1. The table shows some research papers along with the datasets they used for landmark recognition and their accuracies.

S. No.	Year	Research Article	Authors	Dataset Used	Landmarks	Technique	Training Accuracy	Validation Accuracy
1	2019	Google Landmarks Dataset v2-A Large-Scale Benchmark for Instance-Level Recognition and Retrieval [1]	T. Weyand, A. Araujo, B. Cao, J. Sim	Google Landmarks Dataset v2	200000	ResNet101 + ArcFace	70%	40%
2	2019	2nd Place and 2nd Place Solution to Kaggle Landmark Recognition and Retrieval Competition 2019 [14]	Chen, Kaibing, et al.	Google Landmarks Dataset v2	200000	SE ResNeXt152	81%	60%
3	2017	LargeScale Image Retrieval with Attentive Deep Local Features [17]	H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han	Google Landmarks	30000	DELF	81%	62%
4	2015	Visual landmark recognition from Internet photo collections: A large-scale evaluation [18]	T. Weyand and B. Leibe	Paris500k	13000	Voting	61%	43%
5	2016	Fine-tuning CNN Image Retrieval with No Human Annotation [19]	F. Radenovic, G. Tolias, and O. Chum.	Flickr-SfM	713	CNN	90%	65%

6	2016	Deep Image Retrieval: Learning Global Representations for Image Search [20]	A. Gordo, J. Almazan, J. Revaud, and D. Larlus	Landmark URLs	586	Deep Nets	83%	63%
7	2018	Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking [21]	F. Radenovic, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum	Revisited Oxford	11	HesAff–rSIFT–SMK	95%	82%
8	2015	24/7 place recognition by view synthesis [16]	A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla.	24/7 Tokyo	125	View Synthesis	85%	62%
9	2012	Worldwide Pose Estimation using 3D Point Clouds [22]	Y. Li, N. Snavely, D. Huttenlocher, and P. Fua	Landmarks-PointCloud	1k	3D Point Clouds	73%	45%
10	2011	Building Rome in a Day [23]	S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. Seitz, and R. Szeliski	San Francisco	15k	ANN SIFT	70%	44%
11	2010	City-Scale Landmark Identification on Mobile Devices [24]	D. Chen, G. Baatz, K. Koser, S. Tsai, R. Vedantham,	European Cities 50k	20	PFI Pipeline	95%	67%

			T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk					
12	2010	Feature Map Hashing: Sub-linear Indexing of Appearance and Global Geometry [25]	Y. Avrithis, G. Tolias, and Y. Kalantidis	Geotagged StreetView	---	FMH	---	---
13	2010	Avoiding Confusing Features in Place Recognition [26]	J. Knopp, J. Sivic, and T. Pajdla	Rome 16k	69	Confuser suppression + Query expansion	48%	---
14	2008	Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases [27]	J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman	Paris	11	Soft Assignment	72%	48%
15	2008	Hamming Embedding Â´and Weak Geometric Consistency for Large Scale Image Search [28]	H. Jegou, M. Douze, and C. Schmid	Holidays	500	HE + WGC	53%	40%
16	2015	Landmark recognition with sparse representation classification and	Cao, Jiuwen, et al.	NTU Campus	1	Sparse Representation Classification	93%	83%

		extreme learning machine [29]						
17	2019	Large scale landmark recognition via deep metric learning [30]	Boiarov, Andrei, and Eduard Tyantov	Scenes	4	Deep Metric Learning	90%	80%
18	2018	An accurate retrieval through R-MAC+ descriptors for landmark recognition [31]	Magliani, Federico, and Andrea Prati	Holidays	500	R-MAC+ with retrieval	91%	75%
19	2017	A location-aware embedding technique for accurate landmark recognition [32]	Magliani, Federico, Navid Mahmoudian Bidgoli, and Andrea Prati.	ZuBuD	460	locVLAD	86%	70%
20	2018	Efficient nearest neighbors search for large-scale landmark recognition [33]	Magliani, Federico, Tomaso Fontanini, and Andrea Prati	Holidays	500	BoI adaptive multi-probe LSH	86%	63%
21	2018	Landmark detection in 2D bioimages for geometric morphometrics: a multi-resolution tree-based approach [34]	Vandaele, Rémy, et al.	Geometric Morphometrics	20	Tree Based	82%	70%

22	2019	Fast Landmark Recognition in Photo Albums [35]	S. Pierre, and C. Oprean	Film Simulation HALDCLUT	---	---	---	---
23	2016	Bee-inspired landmark recognition in robotic navigation [36]	CA Kodi Cumbo, et al.	Robot Camera	---	---	---	---
24	2008	Multiscale Unsupervised Segmentation of SAR Imagery Using the Genetic Algorithm [37]	Xian Bin Wen, Hua Zhang and Ze Tao Jiang	---	---	Genetic Algorithm	93%	---
25	2008	Perceptual Image Segmentation Using Fuzzy-Based Hierarchical Algorithm and Its Application to Dermoscopy Images[38]	J. Maeda, A. Kawano, S. Yamauchi, Y. Suzuki A. R. S. Marcal and T. Mendonc	---	---	Fuzzy Algorithm	90%	---
26	2007	A QuantumInspired Evolutionary Algorithm for Multiobjective Image Segmentation [39]	Hichem Talbi, Mohamed Batouche and Amer Draa	---	---	---	---	---
27	1996	Neural-based color image segmentation and classification using self-organizing maps [40]	Jander Moreira and Luciano Da Fontoura Costa	---	---	---	---	---

In contrast to this, Weyand, Tobias, and Bastian Leibe. (2015), has a dataset of 501k images and landmarks amounting to 13k [18]. However, in this dataset accuracy was drastically reduced to just 61% and the dataset was unstable meaning that it could not be used for further uses. This article, published in 2015, showed that in that year neither the large datasets could be stable, nor could we get high accuracy on such datasets. Therefore, it was implicit that we required better AI models for competent Computer Vision.

In Gordo, Albert, et al. (2016), authors analyze how making a computer learn Global Representations could be used for image search [20]. An AI model can be trained using Global Representations by which they can learn how to distinguish different things from others. In this article, there were 586 landmarks, and still, the accuracy was just about 83% on testing data.

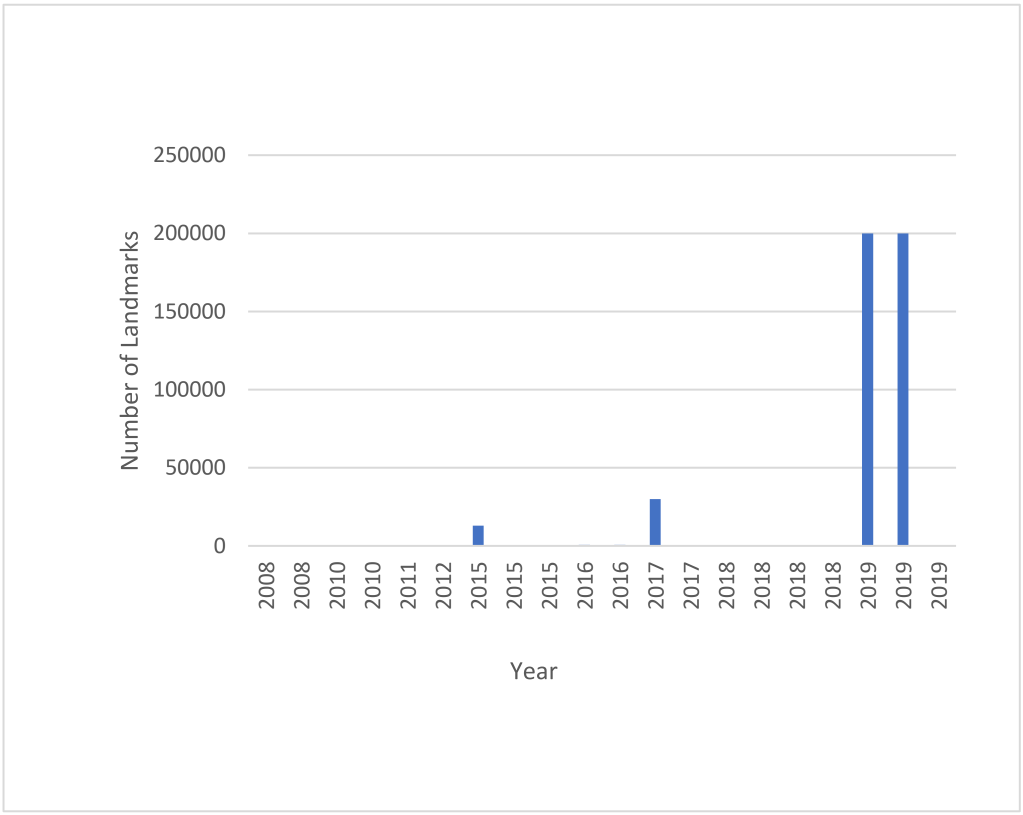


Figure 2.3: Dataset Size trends with Time

Then, Radenović, Filip, Giorgos Tolias, and Ondřej Chum. (2018) was published which came up with an accuracy of 90% on a dataset with 713 landmarks. Although the accuracy is considerable, however, in comparison to [18], the number of landmarks is too small and so was the dataset with just 120k images which included training and test images. Noh, Hyeonwoo, et al. (2017) published a paper published taking the Google dataset as a training dataset [17].

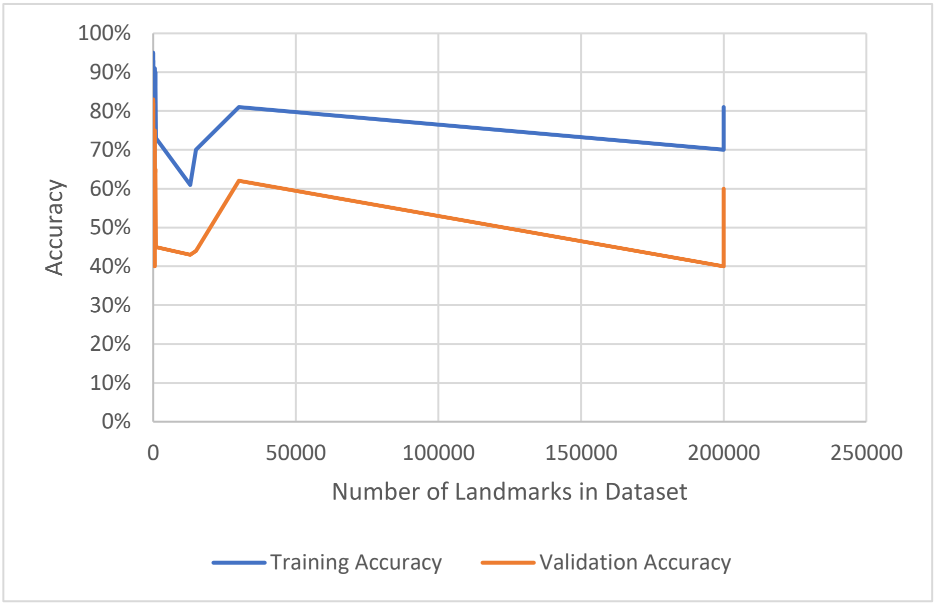


Figure 2.4: Accuracy trends with Dataset Sizes

This was an unstable dataset but a large one. It had about 30k landmarks and 1.2M images. This research article came up with an accuracy of approximately 81% which is not considerable but taking into consideration, the size of the dataset, this dataset was able to achieve some accomplishments at least. Then, in 2018, an article named Radenović, Filip, et al. (2018) was published [21]. This article had only 11 test landmarks and was able to achieve 95% accuracy. It was not a big achievement, but the article was very important because introduced new methods of benchmarking for areas like Landmark Recognition. This paper had 70 images and all of them were used as test images.

From Figure 2.3, we observe that the trends show that the dataset sizes have, continuously, been increasing with each passing year. Though the datasets might not be robust, however, sizes are increasing which means variability in the data is increasing and thus machine learning algorithms are becoming more powerful with each passing year. The year 2019 shows a dataset of 200,000 landmarks which amounted to more than 4.1M images meaning that an average landmark class had over 200 images. Also, we observed that there were articles with very few landmarks that were not that relevant since dataset sizes have seen an increasing trend after any number of articles with small datasets.

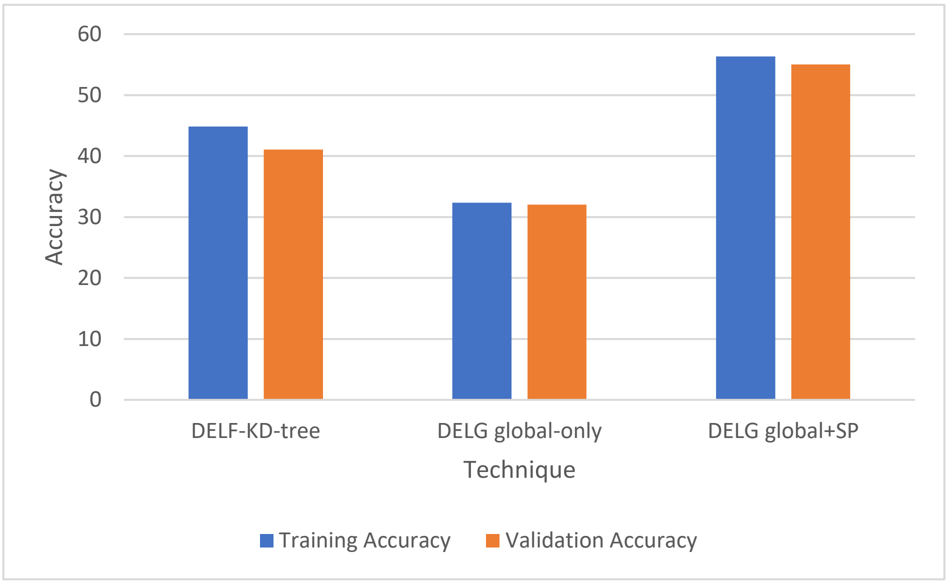


Figure 2.5: Accuracies on Google Landmarks Dataset

Table 2.2. Training and Validation Accuracies of various techniques on Google Landmarks Datasets [9]

Technique	Training Accuracy	Validation Accuracy
DELF-KD-tree	44.84	41.07
DELG global-only	32.37	32.02
DELG global+SP	56.35	55.01

Fig. 2.4 shows the trends in accuracies achieved in datasets as their sizes increase. As we notice, when first, the size of datasets increases considerably, the accuracy also falls by considerable numbers. However, when years pass by and the dataset size remains almost the same or does not increase by a considerable level, the accuracy increases by substantial amounts. We can infer from the figure that though, accuracy decreases with drastically increasing dataset sizes, however, substantial accuracies are achieved as years pass by.

Then the next article that introduced very large-scale and stable datasets in 2019. This article named Weyand, Tobias, et al. (2019) uses more than 4.1M images as training and test data [1]. Just like [17], the dataset used in this article has been published by

Google but unlike [20], the dataset was stable, meaning the data could be used for later uses. Also, the accuracy was reduced considerably to 70%. This was the testing accuracy using training data itself, whereas validation accuracy was below 40% which is extremely low accuracy. Since then, Google has regularly announced competitions to reward those who come up with better accuracies on such large datasets.

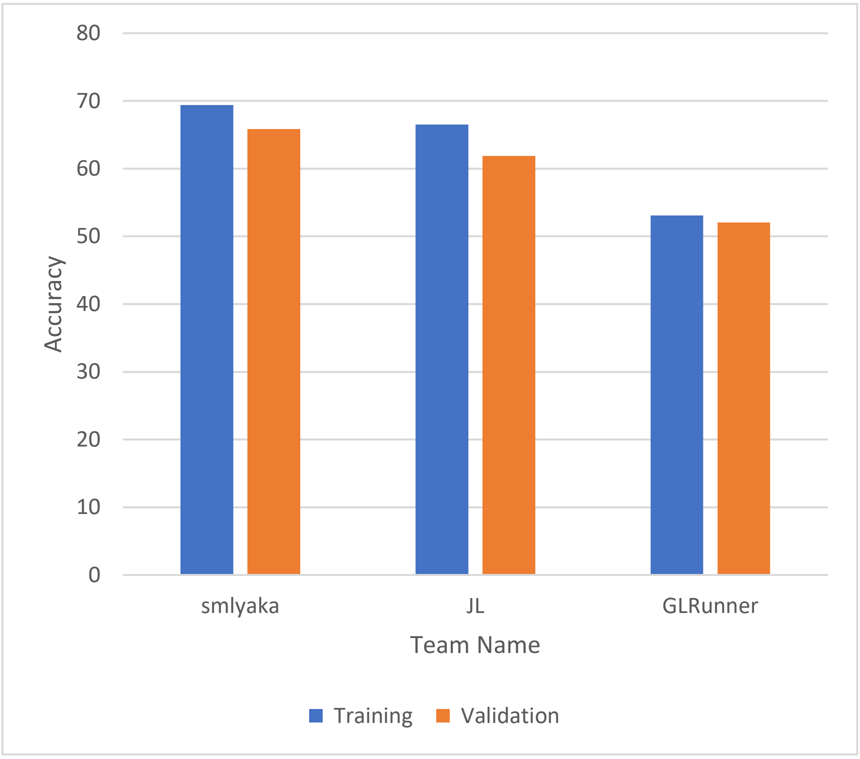


Figure 2.6: Accuracies on Google Landmarks Dataset V2

Table 2.3 Training and Validation Accuracies on Google Landmarks Datasets V2 for various teams along with the techniques used by them [1].

Team Name	Technique	Training	Validation
smlyaka	GF ensemble -> LF -> category filter	69.39	65.85
JL	GF ensemble -> LF -> non-landmark filter	66.53	61.86
GLRunner	GF -> non-landmark detector -> GF+classifier	53.08	52.07

Figures 2.5 and 2.6 show the training and validation accuracies achieved by various techniques and teams on Google Landmarks Dataset and Google Landmarks Dataset V2 [1]. As we can see Google Landmarks Dataset had lower accuracies as compared to Google Landmarks Dataset V2, thereby meaning that, Dataset V2 was more robust as compared to the previous version. Though the dataset size of version 2 was larger, getting better results meant that AI techniques bettered manifolds.

Table 2.4 Research articles with the need addressed in them and work accomplished.

Research Article	Need Addressed	Work Accomplished
Babenko, Artem, et al. (2014) [41]	Image Recognition and Retrieval	Built Convolutional Neural Networks
Suau, Pablo. (2005) [42]	Robust Landmark Recognition	Developed Polar Histograms
Chen, Yuntao, et al. (2019) [43]	Deal Large Datasets	Design Distributed Frameworks
Crall, Jonathan P., et al. (2013) [44]	Instance Recognition	Neural Nets for Species Recognition
Scheirer, Walter J., et al. (2011) [45]	Assigning scores for Image Recognition	Theory and Practice Methods Discussed
Teichmann, Marvin, et al. (2019) [46]	Image Classification	Efficient Regional Aggregation
Zitová, Barbara, and Jan Flusser (1999) [47]	Better Landmark Recognition	Landmark Recognition using invariant features
Cao, Jiuwen, et al. (2015) [29]	Better Landmark Recognition	Extreme Learning Machines
Srinivasan, Sridhar, and Laveen Kanal. (1997) [48]	Qualitative Landmark Recognition	Visual cues for image recognition
Cao, Jiuwen, Tao Chen, and Jiayuan Fan. (2014) [49]	Feature Extraction for Landmark Recognition	SLFNs
Chen, Tao, and Kim-Hui Yap. (2013) [50]	Qualitative Landmark Recognition	Mobile Landmark Recognition
Chen, Tao, Kim-Hui Yap, and Dajiang Zhang. (2012) [51]	Qualitative Landmark Recognition	BoW Framework
Li, Xiaowei, et al. (2008) [52]	Landmark Recognition and Modelling	Iconic scene graphs

The article Babenko, Artem, et al. (2014) tries to make neural networks for image recognition and retrieval [41]. It shows that simple machine learning techniques cannot work on image retrieval and we need more complex techniques. So, authors build neural networks including Convolutional Neural Networks (CNNs) which help us in image retrieval and classification. Suau, Pablo. (2005), had developed polar histograms for robust Landmark Recognition [42].

Chen, Yuntao, et al. (2019) discuss how distributed frameworks help us in dealing with large datasets such as those for instance recognition [43]. We can design simple and versatile distributed frameworks for this matter. One of the developed frameworks: SimpleDet; has been discussed in the research article. According to “Chen, Yuntao, et al.”, “SimpleDet supports up-to-date detection models with best practice. SimpleDet also supports distributed training with near-linear scaling out of the box.”

Crall, Jonathan P., et al. (2013) deal with one of the instance recognition frameworks namely species recognition [44]. This article discusses neural nets built for animal and plant species recognition. Though the article has little to do with Landmark Recognition, it was also one of the good articles in establishing procedures related to instant recognition. Using a labeled database and a quick, accurate algorithm, HotSpotter can detect particular animals. It makes use of an algorithm for competitive scoring called the Local Naive Bayes Nearest Neighbour Algorithm.

Scheirer, Walter J., et al. (2011) discuss the important theory and practice methods followed while analyzing scores for image recognition [45]. The AI systems are assigned scores according to how well they fare at a task. For the task of Image Processing and Instance Recognition, this research article discussed how the scores are created, assigned, analyzed, and interpreted. Both theory and practice methods have been discussed conveniently.

Teichmann, Marvin, et al. (2019) discuss how Efficient Regional Aggregation can be used to identify images in an image classification dataset [46]. This is yet another technique used for image search. “Teichmann, Marvin, et al.” demonstrate how a trained landmark detector, using their new dataset, could be leveraged to index image regions and improve retrieval accuracy while being much more efficient than existing regional methods. Zitová, Barbara, and Jan Flusser (1999) discussed landmark recognition with features that remain invariant in data images [47]. This technique helped to further the research on Landmark Recognition by using similar features in different images.

The research article by Cao, Jiuwen, et al. (2015) discusses more complex techniques namely sparse representation classification and extreme learning machines [29]. “Cao, Jiuwen, et al." showed how these techniques, combined, produced good results for the field of Landmark Recognition. Srinivasan, Sridhar, and Laveen Kanal. (1997) discusses visual cues for image recognition [48]. The article explains how there are some visual cues in an image that may be used for Qualitative Landmark Recognition.

Cao, Jiuwen, Tao Chen, and Jiayuan Fan. (2014) article proposed a fast online learning framework based on the Single Hidden Layer Feedforward Neural Networks (SLFNs) [49]. This technique is also based on the bag-of-words (BoW) method which is used for feature extraction for Landmark Recognition. Chen, Tao, and Kim-Hui Yap. (2013), discusses a discriminative BoW framework for mobile Landmark Recognition [50].

Chen, Tao, Kim-Hui Yap, and Dajiang Zhang. (2012) was a research article published in 2012, which used the technique of Discriminative bag-of-visual phrase learning for Landmark Recognition [53]. Li, Xiaowei, et al. (2008), published in 2008, developed Landmark Recognition and Modelling using iconic scene graphs [52].

Chen, Tao, et al. (2009) was another survey article published in 2009 for information retrieval for mobile landmark recognition [54]. Chen, Wei-Chao, et al., (2009) was a research article published in 2009 that developed benchmarks for summarising Landmarks from community photo collections [55]. Li, Hao, and Simon X. Yang. (2002) discussed the evolution of visual Landmark Recognition systems [56].

Chen, Tao, Kim-Hui Yap, and Dajiang Zhang. (2014), published in 2014, discussed the technique named Discriminative soft bag-of-visual phrase which has been used for mobile Landmark Recognition [51]. Chen, Tao, Kim-Hui Yap, and Lap-Pui Chau. (2011) published in 2011, uses integrated content and content analysis for mobile Landmark Recognition [57].

Stefani, Pierre, and Cristina Oprean. (2019) discussed landmark recognition in photo albums which were both fast and comprehensive [35]. In this article, a DELF network is used with a VLAD layer in a CNN network. Guo, Jianya, Xi Mei, and Kun Tang. (2013) discussed 3D techniques for the identification of 3D things [58]. This article used landmark annotations and a technique named dense correspondence registration for building 3D human facial images. The article describes how landmarks can be used in other fields.

Table 2.5 Research articles in the area of Instance Recognition Addressed

Research Article	Area Addressed
Chen, Tao, et al. (2009) [57]	Mobile landmark recognition
Li, Hao, and Simon X. Yang. (2002) [56]	Visual Landmark Recognition systems
Chen, Tao, Kim-Hui Yap, and Dajiang Zhang. (2014) [51]	Mobile landmark recognition
Hui Yap, and Lap-Pui Chau. (2011) [57]	Mobile landmark recognition
Stefani, Pierre, and Cristina Oprean. (2019) [35]	Landmark recognition in photo albums
Guo, Jianya, Xi Mei, and Kun Tang. (2013) [58]	Identification of 3D things
Kragskov, Jens, et al. (1997) [59]	Landmark identification in craniofacial items
Déniz, Oscar, et al. (2011) [60]	Face recognition
Van Asselen, Marieke, Eva Fritschy, and Albert Postma. (2006) [61]	Better navigations
Epstein, Russell A., and Lindsay K. Vass. (2014) [62]	Landmarks Based wayfinding
Magliani, Federico, Tomaso Fontanini, and Andrea Prati. (2019) [63]	Small Scale to Large Scale Datasets
Zheng, Yan-Tao, et al. (2009) [64]	Web-scale Landmark Recognition
Wu, Jianixn, and James M. Rehg. (2008) [65]	Spatial Navigation

Kragskov, Jens, et al. (1997) discussed important landmark identification in craniofacial items using cephalometric radiographs and CT scans [59]. This article, just like [58], is a case of landmarks other than inanimate objects. Déniz, Oscar, et al. (2011) discuss face recognition using oriented gradient technology in histograms [60]. This article, again like [58], [59], is another instance of recognition of landmarks for animate objects.

Van Asselen, Marieke, Eva Fritschy, and Albert Postma. (2006) discusses the psychological behaviors of humans in how they acquire spatial knowledge for navigation and how machines can be developed for better navigation using this learning [61]. Similarly, Epstein, Russell A., and Lindsay K. Vass. (2014) discuss neural systems for landmark-based wayfinding in humans [62]. This is a technique that is used not only by animate organisms but using this technique, we may even be able to train an Artificial Intelligent computer System.

Magliani, Federico, Tomaso Fontanini, and Andrea Prati. (2019) discusses the datasets ranging from scale to large scale that have been used to date for Landmark Recognition and Retrieval [63]. This was also a comprehensive study that discussed all the simple and difficult traits of Landmark Recognition. Also, it discussed the achievements and research gaps of various datasets ranging from very small ones to ones having a million images. Zheng, Yan-Tao, et al. (2009) published in 2009, also aimed to develop a web-scale Landmark Recognition engine that could be used for later uses as well [64]. Wu, Jianixn, and James M. Rehg. (2008) used PACT technology to determine the whereabouts of a person or robot [65]. The technique uses histograms to know where a person is in space.

Li, Hao, and Simon X. Yang (2003). was a research article published in 2003 that built a fully autonomous mobile robot [66]. It developed various modules that could implement different levels of competence and behaviors in the robots. Similarly, Nasr, Hatem, and Bir Bhanu. (1988) also worked on developing autonomous mobile robots [67]. This research article, however, was published in 1988, when not too much hardware existed. This article was more dealt with more theoretical aspects of building an autonomous mobile robot. The authors didn't practically implement anything, but the article paved the way for a better understanding of future aspects. Trahanias, Panos E., Savvas Velissaris, and Thodoris Garavelos. (1997) published in 1997 had more success in Landmark extraction and recognition [68]. This article was one of the early articles published for autonomous robot navigation development. It was quite successful in bettering autonomous robot navigation.

Lee, Jung-Sub, et al. (2009) published in 2009, developed In-pipe robot navigation based on Landmark Recognition using shadow images [69]. This article, also for navigation of robots, dealt with developing robot navigation using a technique named In-pipe robot navigation. The technique used images of shadows of landmarks in an image to navigate through the surroundings. Elmogy, Mohammed, and Jianwei Zhang. (2008) was a research article published in 2008, which developed robust and real-time techniques for Landmark Recognition that were used for humanoid robot navigation [70]. The authors discuss humanoid robot navigation which intends to make robots navigate their environment like humans. Mata, Mario, et al. (2001) published in 2001 discussed visual Landmark Recognition systems for the topological navigation of mobile robots [71]. It discussed how robots can be trained to deal with all the 3-dimensional obstacles that enter their pathways. Liu, Tingting, et al. (2019) discuss how Infrared Spectral Imaging can be used to sense objects by robots [72]. This would lead to better robotic vision.

Table 2.6: Research articles in the area of Robotic Technologies

Research Article	**Work Accomplished**
Li, Hao, and Simon X. Yang (2003) [73]	Fully autonomous mobile robot
Nasr, Hatem, and Bir Bhanu. (1988) [67]	Autonomous mobile robots
Trahanias, Panos E., Savvas Velissaris, and Thodoris Garavelos. (1997) [68]	Autonomous robot navigation
Lee, Jung-Sub, et al. (2009) [69]	In-pipe robot navigation
Elmogy, Mohammed, and Jianwei Zhang. (2008) [70]	Humanoid robot navigation
Mata, Mario, et al. (2001) [71]	Topological navigation of mobile robots
Liu, Tingting, et al. (2019) [72]	Infrared (IR) spectral imaging sensing
Yang, Simon X., and Hao Li. (2002) [73]	Mobile robots
Do, Quoc V., and Lakhmi C. Jain. (2006) [74]	Autonomous robot navigation
Luo, Ren C., Harsh Potlapalli, and David W. Hislop (1992) [75]	Robot navigation

Yang, Simon X., and Hao Li. (2002) published 2002, succeeded in developing mobile robots that not only were autonomous but could avoid obstacles in their path with a fuzzy logic behavior. This article was also used as a visual Landmark Recognition System [73].

Do, Quoc V., and Lakhmi C. Jain. (2006), published in 2006, was yet another attempt to develop landmark recognition systems for autonomous robot navigations [74]. Luo, Ren C., Harsh Potlapalli, and David W. Hislop (1992) discussed how neural networks could be used for robot navigation [75]. This was developed using Landmark Recognition for robotic technologies developed using neural networks.

Table 2.7: Various types of optimizations discussed by various research articles.

Year	Research Article	Authors	Optimization
2005	Ga-facilitated knn classifier optimization with varying similarity measures [76]	Peterson, Michael R., Travis E. Doom, and Michael L. Raymer	Optimized KNN with Genetic Algorithm
2008	Particle swarm optimization for parameter determination and feature selection of support vector [77]	Lin, Shih-Wei, et al.	Optimized SVM with Particle Swarm Optimization
2012	Simultaneous feature selection and SVM parameter determination in classification of hyperspectral imagery using ant colony optimization [78]	Samadzadegan, Farhad. Hadiseh Hasani, and Toni Schenk.	Optimized SVM with Ant Colony Optimization
2017	A hybrid approach from ant Colony optimization and K-nearest neighbor for classifying datasets using selected features [79]	El Houby, Enas MF, Nisreen IR. Yassin, and Shaimaa Omran	Optimized KNN with Ant Colony Optimization
2017	An improved grey wolf optimization strategy enhanced SVM and its application in predicting the second major [80]	Wei Yan, et al.	Optimized SVM with Grey Wolf Optimization
2019	Optimization of K-nearest neighbor using particle swarm optimization for face recognition [81]	Sasirekha, K., and K. Thangavel	Optimized KNN with Particle Swarm Optimization
2019	GA-SVM based feature selection and parameter Optimized SVM with optimization in hospitalization expense modelling [82]	Tao, Zhou, et al.	Optimized SVM with Genetic Algorithm
2020	Binary Grey Wolf Optimizer with K-Nearest Neighbor classifier for Feature Selection [83]	Al-wajih, Ranya, et al.	Optimized KNN with Grey Wolf Optimization

We have seen that a lot of work has been done in the field of Landmark Recognition. We see that with time, the sizes of datasets are growing as they might be expected to. However, as the dataset sizes increase, accuracies seem to go down. The techniques that seem to do well on smaller datasets fail miserably on larger ones. Hence, there is a need for developing techniques that do well not only on datasets of smaller sizes but on datasets of relatively all sizes.

Today, there exist many Nature Inspired Computing (NIC) Algorithms [76]–[83]. These algorithms are based on the working of natural phenomena in the environment and can be used for different areas like routing in computer networks, test case generations, image processing, etc [84]–[91]. The NIC algorithms can also be used to optimize the existing AI techniques like KNN, SVMs, etc.

As we are trying to develop AI models using and for natural elements in the surroundings, it might not be a thing of surprise that using a touch of nature should better the AI models currently being used for Image Processing for Landmark Recognition. No wonder, most of the algorithms and techniques developed today, for instance, recognition, is based on the natural processes and phenomena occurring in our environment.

The processes and phenomena on which they are based include how organisms reproduce [88], how bats see [89] and how our universe started [90]. These techniques provide a strong basis for the development of better AI models since AI is expected to work like natural processes. No wonder, Nature-Inspired Computing is widely used to optimize AI techniques today [76]–[116]. NIC algorithms work more like phenomena inspired by nature and hence are closer to functioning in a way that delivers nature-like results. Viewing is a natural phenomenon that we aim to pass on to machines. So, Nature Inspired Techniques can go a long way in developing Artificially Intelligent models.

We predict that Nature Inspired Computing techniques which include Neural Networks will supposedly be used mostly in all future developments. They would be used to manufacture machines that would have abilities like those of living beings. An AI model that might be able to work close to the abilities that nature has provided should be inspired by nature.

We have surveyed the research articles ranging from 1988 till the present date related to Landmark Recognition and Instance Recognition. From the survey, we observed that since the sizes of datasets in the field of Landmark recognition have been growing (as they are expected to), the accuracies have been decreasing and there is a constant need for better technologies. Though better AI techniques have continuously been developing, however, to expect robotic vision, we either need even better AI techniques better hardware, or both. From the survey, we have concluded that the field of Landmark Recognition is given high importance in AI but it still has a long way to go due to various issues including the large size of datasets. Datasets in this field are extremely large and AI computer systems fail to train themselves in a proper manner and limited time. To make an AI system visualize things close to human-like precision, we need much better AI models and/or better hardware configurations. We also presented an implementation of classical Machine Learning Techniques on a part of Google Landmarks Dataset V2 and we could infer that these techniques will not work on datasets of such large sizes. A lot of research has already been done in this field and a lot more has yet to be done. If researchers reach appropriate results, this field can change the future of AI and robotic technologies.

2.3. Survey of Classical Machine Learning Techniques

A Survey of the Literature reveals that many approaches are being developed for the process of Landmark Recognition which includes RESNETs, DELF networks, VGGNETS, etc. This section focuses on the implementation of a simple CNN architecture for geographical landmark recognition and compares it with classical machine learning algorithms like KNN, Decision trees, and SVM.

In this section, we analyze the performance of SVM, KNN, Decision Tree, and Random Forest, the four most popular classical Machine Learning Approaches. We analyzed the performance of these approaches on 1000 images out of Google Landmarks Dataset V2. The performance results are shown in Table 2.8 and Fig. 2.7.

Table 2.8: Performance analysis of classical ML approaches on 1000 images out of Google Landmarks Dataset V2.

ML Approach	Training Accuracy (in%)	Test Accuracy (in %)
SVM	51	11
KNN	45	11
Decision Tree	42	10
Random Forest	58	12

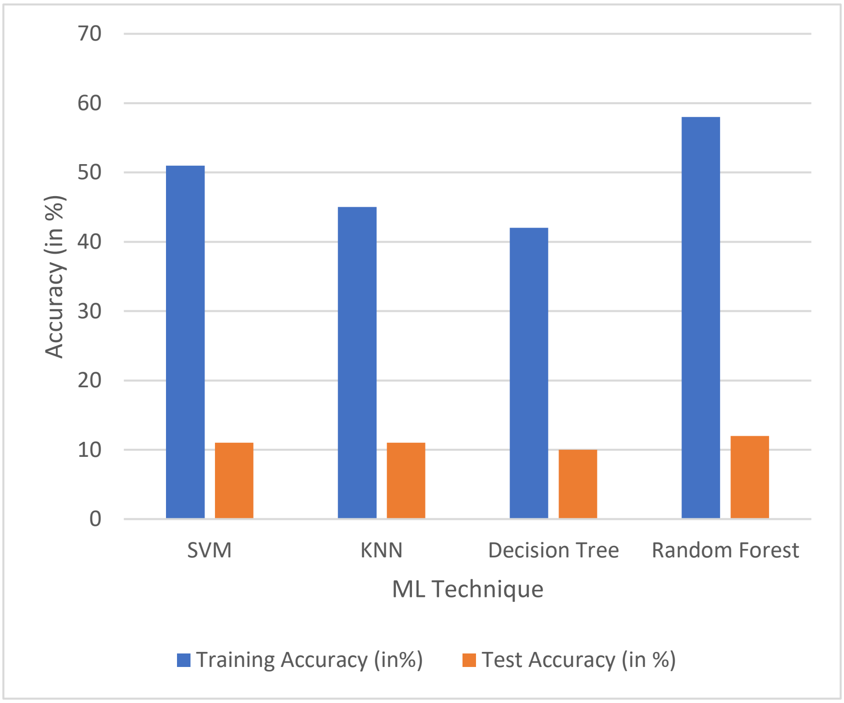


Figure 2.7: Training and Test Accuracies for various ML Approaches.

2.4. Survey of Convolutional Neural Networks: Performance Analysis

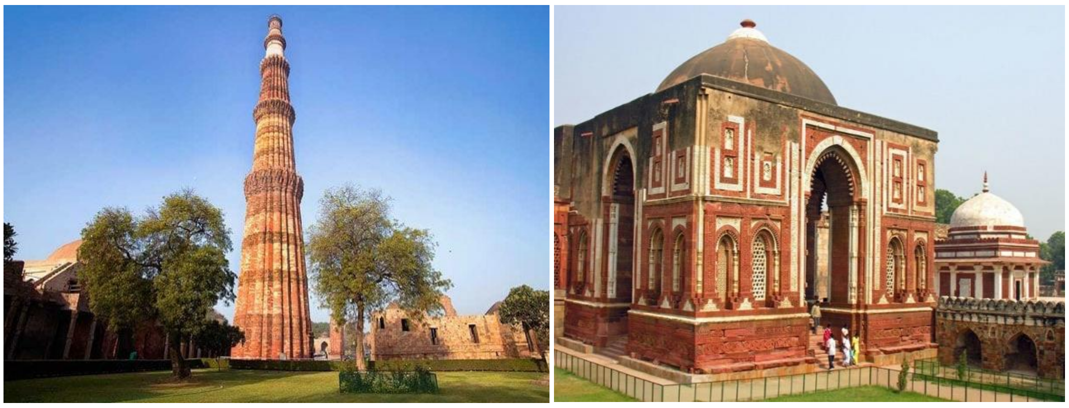


Figure 2.8: Images of Qutub Minar (left) and Alai Darwaza (right) from our dataset.

CNNs can be designed in plenty of ways to make them perform better on a particular dataset. Today there exist complex CNN architectures namely ResNets, VGGNets, Inception Nets, etc [117]. We take a very simple CNN architecture with just a single Convolutional Layer which is singularly able to perform better than all the classical ML approaches that we discuss later in the section.

We took our dataset from a Kaggle Competition named Qutub Complex Monuments' Images Dataset (https://www.kaggle.com/varunsharmaml/Qutub-complex-monuments-images-dataset). The dataset was published in 2018 as a public dataset and consisted of five famous Landmarks from the National Capital Region, Delhi, India. The dataset consisted of Landmark places namely Alai Darwaza, Alai Minar, Iron Pillar, Jamali Kamali Tomb, and Qutub Minar. Figure 2.8 shows two images from this dataset.

The original dataset consisted of 1286 image files which were not very uniform in nature. Some image files were in JPEG format and some were in PNG format. A few image files were in grayscale and others were colored in nature. Out of the colored images, some files used only three color channels, namely Red Green Blue (RGB) while a few used four color channels, namely Cyan Magenta Yellow Key (CMYK).

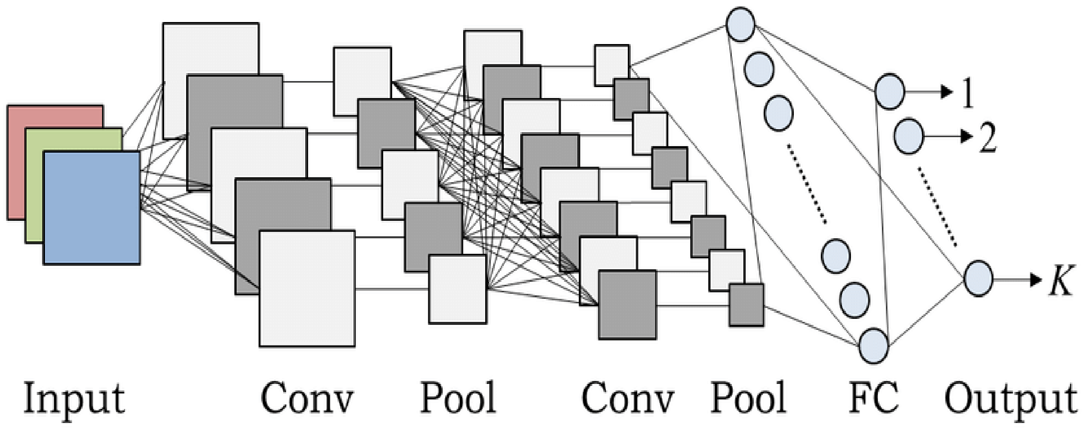


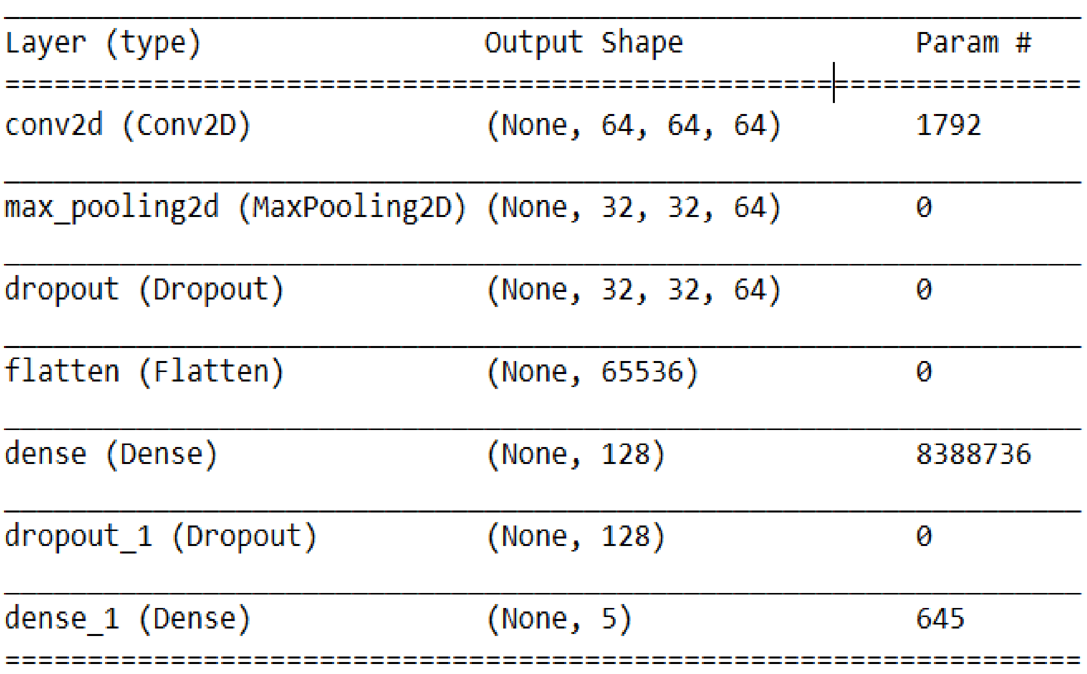
Figure 2.9: Architecture of a Convolutional Neural Network with two convolutional layers [13]

It is seen that CNNs perform best when all the images that are used for training have the same shape (dimensionality) throughout. So, we decided to make the data uniform to run our codes efficiently and without hassle. We decided to run our codes on Jupyter

Python Notebooks. So, the shape function from the NumPy library was run to identify the shape of each image and images that could have caused trouble were identified.

We could make the data uniform by identifying the images that were causing non-uniformity and then removing them. We identified 16 images that were either Grayscale images or with 4 Color channels and removed them from our dataset. The remaining dataset consisted of 1270 files which consisted of only the colored images with 3 color channels namely RGB. This was a good size for a dataset for comparing various ML techniques.

Also, since the image files had varying image dimensions of 64 x 64. So, one image file had a shape of (64,64,3) which included 64 pixels of height, 64 pixels of width, and 3 channels of colors namely RGB (Red Green Blue). Every channel of color could take 256 (0-255) shades of each color. Figure 2.9 shows the architecture of the neural network developed by us.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 128)               8388736
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
```

Figure 2.10: Summary of the Convolutional Neural Network model designed by us.

We divided our dataset into training and test files with a ratio of 75:25 for classical ML Approaches and 70:30 for CNN. We provided more training data to classical machine learning techniques and less training data to CNN and still, CNN could outperform classical ML techniques like SVM, Decision Tree, and KNN even with fewer training files. We shall discuss this, in detail, in the next sections.

2.4.1. Layers in CNNs

Figure 2.9 describes how a simple Convolutional Neural Network can be designed [5], although the CNN designed by us has only one Convolutional Layer. The above architecture exemplifies that there are two Convolutional Layers, two pooling layers, and one fully connected layer in the network.

In the Convolutional Neural Network designed by us, there is a single Convolutional Layer, one max-pooling layer, and two fully connected layers. The model designed by us can be summarized in the manner shown in Figure 2.10. The working of the convolution operation is described in Figure 2.11.

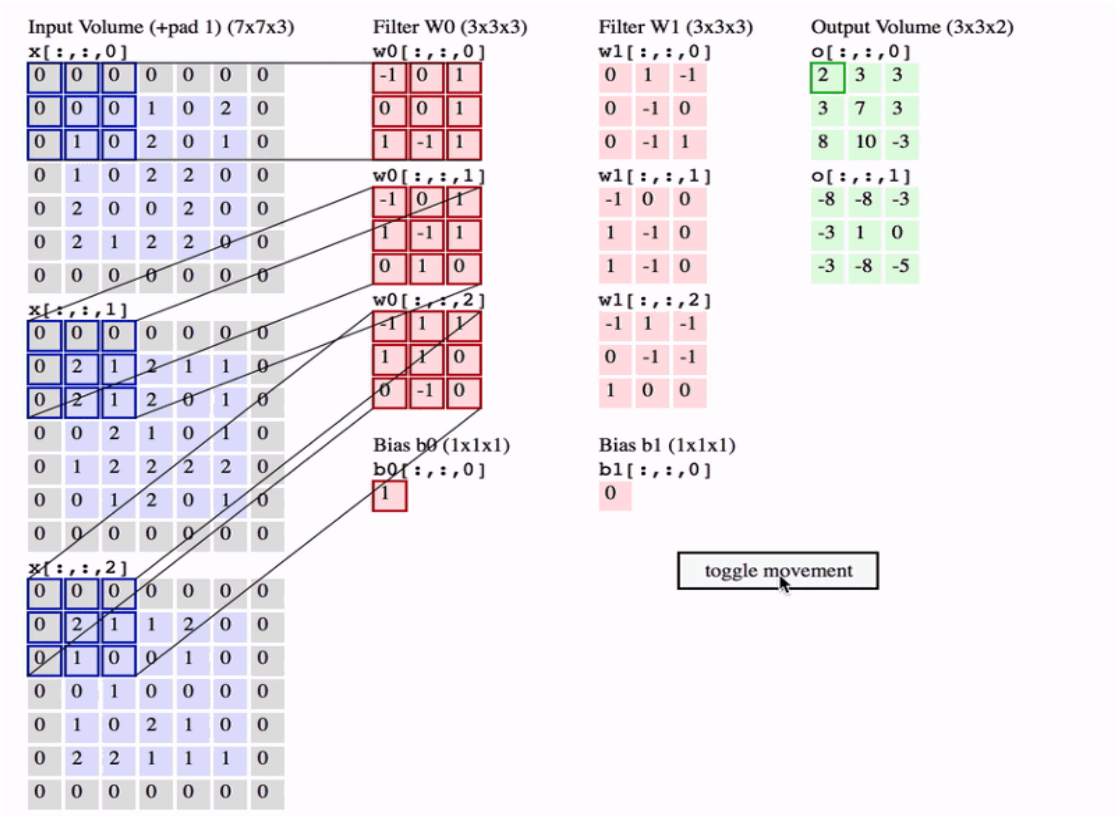


Figure 2.11: A depiction of Convolution operation in a Convolutional Neural Network

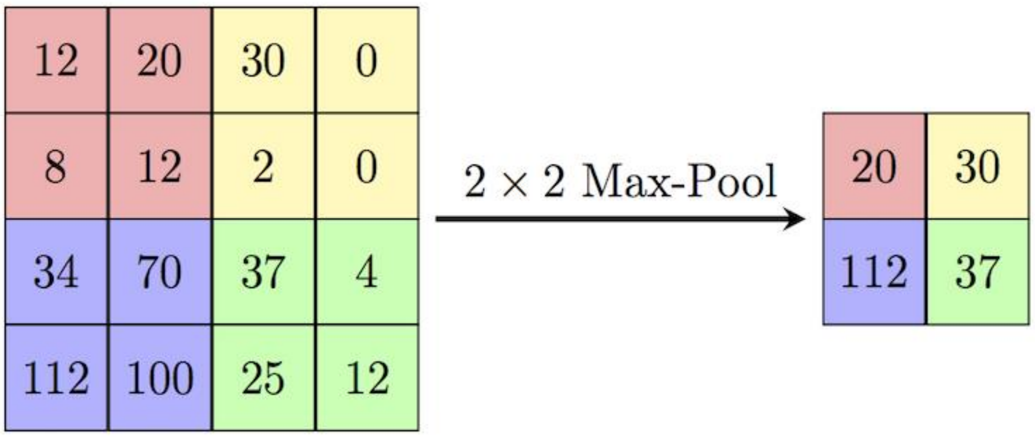


Figure 2.12: The depiction of the Max-Pooling Operation

First, a convolutional layer is initialized. This layer will train the weights and biases for developing the model. Then we apply the Max Pooling Layer to downsample the excessive number of images that have been created. Then a dropout function is applied to reduce the overfitting of the model. Then, a flatten layer converts high-dimensional data into single-dimensional data. Figure 2.12 shows how the computer achieves max-pooling.

Then fully connected layers are applied with a dropout function in between to connect our final classes to the learned weights and biases. Once weights and biases are trained, we can run the network numerous times hoping to get better weights. Weights in CNNs are also known as filters or kernels and are matrices only. Once trained, these kernels can detect many things of interest in an image.

2.4.2. Training CNNs

As we discussed earlier, the images are first resized to the shape (64,64,3) and then this batch of images has to be fed to the network for training. Every image in our training dataset has a height and width of 64 pixels and 3 color channels are used in each image which is namely Red, Green, and Blue (RGB).

Then we pass each image through the Convolutional Neural Network designed by us. The images are to go through the neural network in batches and we give a batch of 32 images at once which is an optimal number to train the weights of the neural network in an appropriate time frame considering limited memory usage.

The basic equations that describe the working of our CNN can be given as under:

$$\text{(32) Batch Size X (64 X 64) Images} \xrightarrow{\text{Convolution}} \text{(64) Kernels X (64 X 64) Convolved Images} \qquad \ldots(2.1)$$

$$\text{(64) Kernels X (64 X 64) Convolved Images} \xrightarrow{\text{MaxPooling}} \text{(64) Kernels X (32 X 32) Pooled Images} \qquad \ldots(2.2)$$

$$\text{(64) Kernels X (32X 32) Pooled Images} \xrightarrow{\text{Fully Connected Activation}} \text{(5) Classes} \qquad \ldots(2.3)$$

When this process completes, the trained weights are further optimized using the AdaDelta optimization algorithm. To train the weights to a considerable accuracy, many epochs have to be given meaning that many similar cycles have to run again and again.

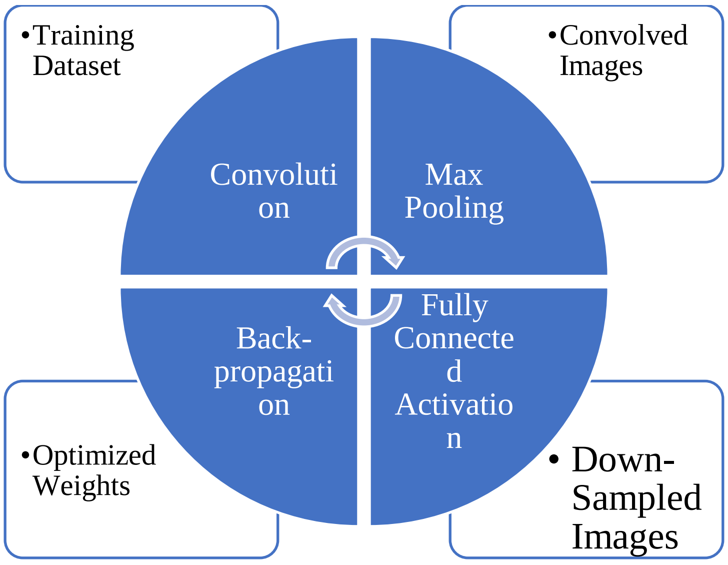


Figure 2.13: The complete process of training a Convolutional Neural Network

One epoch of the training process follows the given algorithm:

ALGORITHM 2.1: Procedure for one epoch of our Convolutional Neural Network

---

Step I.   Initialize 64 kernels and convolve each image with each kernel.

Step II.   Downsample images by max-pooling arrays of 2 x 2.

Step III. Dropping out nodes by 0.25 to avoid overfitting.

Step IV. Flattening the high dimensional input to a single dimension.

Step V. Reducing the count of outputs by a Dense function.

Step VI. Dropping out nodes again by 0.5 to avoid overfitting.

Step VII. Connecting final output to the exact category of image.

First, we initialize our CNN with a convolutional layer that creates 64 kernels of size 3 x 3 for training purposes. These kernels must convolve with the batch of 32 images at a single time for one epoch. Then, we applied a max-pooling layer which reduces the dimensions of convolved images by a factor of 4. Then we apply the dropout function which drops nodes randomly to reduce the overfitting of the model. Then we flatten high-dimensional data into a single dimension with a flattened layer. Finally, we connect all the nodes from the fully connected layer to our 5 classes using a Softmax activation. We then select the best kernels from the total kernels created and these are then chosen for further epochs. The optimization happens using the process of backpropagation where an optimization algorithm feeds backward important information in the network that includes weights and biases.

200 epochs of Algorithm 2.1 were given while training our model with the dataset discussed in Section 2. The model is compiled with the optimizer ‘AdaDelta’ which is a decent optimizing algorithm and a learning rate of 0.01 which is considered good enough to train a CNN so that neither the model misses the optimal weight nor slows down very much. The loss reduced in this network is the categorical cross-entropy loss which is the standard loss parameter to be reduced while training a network on a multi-class classification problem. The evaluation metric for validating our model is accuracy which includes both training and test accuracy to determine if the model is under-fit or over-fit.

After the compilation and execution processes of our code, the model delivered a training accuracy of 98% and a test accuracy of 73% with 200 epochs. As we can see in Fig. 2.14, CNN fares much better than other classical ML Algorithms. We implemented SVM, KNN, and Decision Tree Classifier, in addition to CNN to compare the performance of each technique, and from the data, it was clear that Convolutional Neural Networks performed much better than other classical Machine Learning algorithms. Complete details of the implementation of each ML technique are discussed.

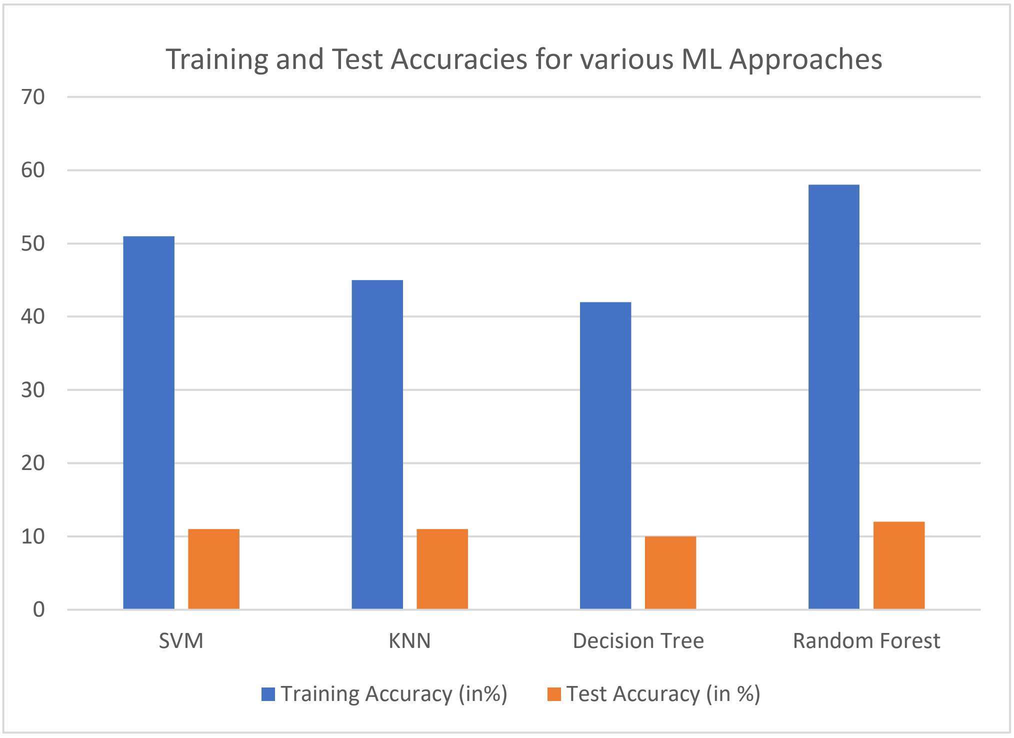


Figure 2.14: Percentage accuracies for Landmark Recognition.

After proper cleaning of our dataset, we ran codes to implement four ML techniques on our dataset which included the CNNs. The codes were used to classify images amongst five famous Landmark places in the National Capital Region, Delhi, India. Three ML techniques that were implemented are the famous classical Machine Learning techniques namely Support Vector Machines (SVMs), Decision Tree Classifier, and K-Nearest Neighbour (KNN) Algorithm. The last technique we applied was the recently developed, Convolutional Neural Networks (CNNs). CNNs are the subset of the latest ML techniques which are broadly included in the Deep Learning paradigm. Deep Learning is concerned with relatively deeper models that are developed for an Artificial

Intelligence task. In classical Machine Learning Techniques, we have only one activation function that controls the entire model, whereas, in Deep Learning, we develop models that consist of a network of activations. This is the same way that the neurons work in our body. So, Convolutional Neural Networks, as the name suggests, belong to the category of Machine Learning known as Neural Networks.

Today, there exists a variety of Neural Networks that perform various Artificial Intelligence tasks better than their classical counterparts. Similarly, Convolutional Neural Networks have emerged as one of the most efficient techniques in the field of Image Processing, Recognition, Retrieval, etc.

First, we shall discuss the results of classical ML algorithms on our dataset. Out of the classical machine learning algorithms listed earlier, the best performance was given by Support Vector Machines which delivered a test accuracy of 64% and a considerable training accuracy of 89% for the classification of the geographical landmarks in our dataset. This suggests that there was overfitting of the model but this overfitting was less than what we found for the other two algorithms.

TABLE 2.9: Training and test accuracies for various ML techniques on a landmark dataset.

ML Technique	**Training Accuracy (in %)**	**Test Accuracy (in %)**
Support Vector Machine (SVM)	89	64
K-Nearest Neighbor	74	54
Decision Tree Classifier	99	49
Convolutional Neural Network	98	73

K-nearest neighbor algorithm fared second best with a test accuracy of around 54% in which the value of K used was 3. Its training accuracy was 74% which again suggests that the model was overfitting. The worst recorded performance overall was given by the Decision Tree Classifier with a test accuracy of 49%. It also recorded a training accuracy of almost 99% which gives us a hint that the technique produced a highly overfit model to be of very little use for generalized tasks.

In this regard, a relatively simple Convolutional Neural Network designed by us fared much better than any of these algorithms. A CNN with just one Convolutional Layer, designed by us, recorded a test accuracy of 73%. Also, the training accuracy was 98% which suggests overfitting, but the test accuracy is considerably higher than other machine learning techniques which makes it a better competitor for Image classification tasks.

For classical machine learning techniques, we gave a train-to-test set ratio of 75:25 while for CNN we gave a more rigorous train-to-test ratio of 70:30. Still, we can see that CNN fared better than all the classical Machine Learning Algorithms.

2.5. Research Gaps

Today, there exist many NIC algorithms like Genetic Algorithm, Ant Colony Algorithm, Particle Swarm Optimization, etc. which are used to optimize existing AI techniques to get the best results for problems. However, it is believed that by using more such algorithms, we would be able to evolve CNNs to get even better performances. The research gaps in the existent literature can be described as follows:

- Convolutional Neural Networks are based on the working of Neural Networks in a living being. Neural Networks are inspired by the mechanisms of neurons, the spinal cord, and the brain in our body. However, every living being has a different set of neural networks in its body. We believe that neural networks can be evolved for even better performances if we evolve them with the Nature-Inspired-Computing (NIC) Algorithms.
- CNNs have a huge set of combinations of hyperparameters that can vary and produce striking results. Some CNN hyperparametric combinations are thought to perform well in a well-established manner but there have been instances of irregulated CNN architectures performing exceedingly well.
- Current Deep Learning techniques fare exceedingly poor on the large, complex, and noisy landmarks datasets. As seen from the survey, as the dataset sizes increase, CNNs tend to underperform, a challenge that needs to be undertaken.

2.6. Conclusion

Various Machine Learning approaches including the Convolutional Neural Networks have been surveyed for the classification of a landmark in an image. It is established how the increasing sizes of datasets degrade the performance of AI models. A performance analysis was also conducted, and the CNNs performed way better than classical ML approaches, implying that classical ML approaches cannot and should not be used for image classification. We also saw that CNNs are much more capable of variations in them making them highly flexible for better usage. Classical ML approaches like SVMs, Decision Tree, and KNN fared poorly on the task of Image Classification plus they are not very flexible for variations. With varying parameters in a CNN, we can and have achieved milestones that could not have been expected earlier.

Chapter 3: Geographical Landmark Recognition: Three Metaheuristic - Based Approaches

In this chapter, we present 3 metaheuristic-based approaches for geographical landmark recognition. The three metaheuristics are namely Paddy Field Algorithm, Genetic Algorithm, and 3 Parent Genetic Algorithm. The metaheuristics are used to search for the most optimal hyperparameters for the development of Convolutional Neural Networks for Geographical Landmark Recognition. The most optimal hyperparameters are searched for through a search space of more than a million hyperparametric combinations and thus specialized CNN architectures are evolved. The experiments were run successfully, and highly optimized CNN architectures are evolving. The CNN hyperparameters highly different from those usually used were found to be highly competent. Kernels with shapes 5 x 5 and 7 x7 were found to perform better than 3 x 3 of some combinations. Relative accuracies of CNNs increased up to more than 40%, thereby indicating that the evolution of CNNs with evolutionary metaheuristics is highly desirable.

3.1. Introduction

NP-Hard problems have existed from times unknown. An NP-Hard problem refers to a problem in the computing arena that is not known to be solved in polynomial time. No definite amount of looping can guarantee an answer to these problems. To find a solution to such problems, soft computing is required [118]. Soft computing tends to find the most optimal approximate solution to the problem instead of an exact solution [119]. The best instances of soft computing are the evolutionary algorithms, and they can be used to find highly optimal solutions to NP-Hard problems.

Millions of hyperparameters in Artificial Neural Architectures can be changed to create the most suitable networks [120]. However, we risk major resource exhaustion problems when evaluating so many parameters in an ANN. A distinct strategy for best-fit ANN search is required due to constraints on time, memory, processing power, etc.

Here is where the concept of NAS comes into play. Neural Architecture Search (NAS) amalgamates evolutionary computing and ANNs to produce better-fit artificial neural networks [121]. These ANNs are evolved on limited data and when tested on complete data, they prove to be highly proficient. Today NAS has been tried with many techniques like Genetic Algorithms (GA) and variants of Particle Swarm Optimization (PSO) [122], [123].

3.2. CNN Evolution with Paddy Field Algorithm

The practice of using metaheuristic methods to address optimization issues is known as metaheuristic optimization. Almost everywhere, from engineering to economics, optimization takes place. Making the most of the available resources is essential because time, money, and resources are all limited.

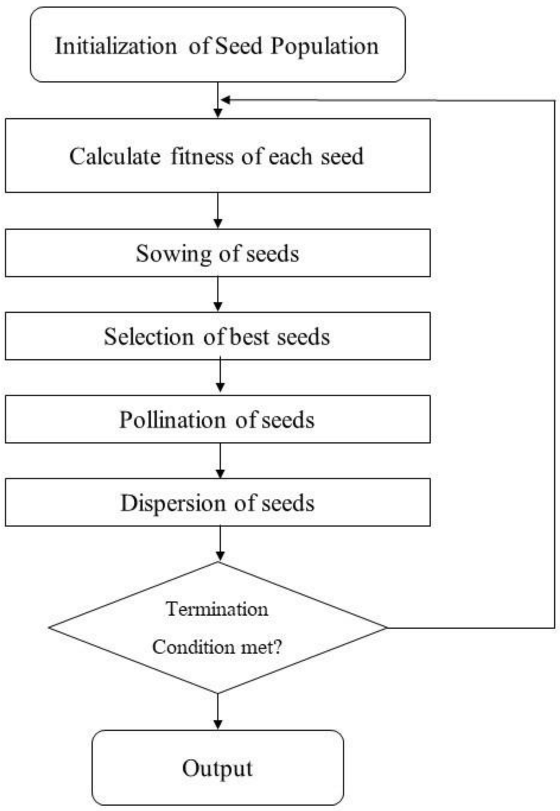

Initialization of Seed Population
Calculate fitness of each seed
Sowing of seeds
Selection of best seeds
Pollination of seeds
Dispersion of seeds
Termination Condition met?
Output

Figure 3.1: The complete process followed in the Paddy Field Algorithm.

The majority of optimizations are extremely nonlinear and multimodal under a variety of challenging constraints. Different hyperparameters may commonly conflict with a single neural network. There are circumstances where ideal solutions don't exist, even for a single purpose. Finding a perfect, or even close to perfect, answer is a difficult task in general.

Paddy Field Algorithm (PFA) is a metaheuristic that is used to search for the best individuals out of a generated population. It was introduced by Premaratne U, et al. in 2009 inspired by the biological process of pollination of seeds of paddy crop [124]. As shown in Figure 3.1, The Paddy Field Algorithm decides the best solution to a problem as paddy seeds spread themselves in a growing area to find the most suitable places to grow.

When one paddy crop grows in an area, the future generations of that crop are decided based on fitness due to pollination. Some seeds are distributed in the neighboring areas and the viability of those seeds is maximum with maximum fitness. A seed with lesser fitness will produce fewer seeds when it grows. This process repeats till the most optimal population of paddy seeds is found. To avoid the possibility of getting caught in a local optimum, seeds are dispersed to search for better spaces. This ensures that most of the search space is searched over by the algorithm.

Algorithm 3.1: Paddy Field Algorithm

Step 1: Initialize a random population of seeds.

Step 2: Calculate the fitness of each seed as:

$$s = q_{max}\left[\frac{y - y_t}{y_{max} - y_t}\right] \qquad \ldots(3.11)$$

The maximum fitness value is expressed as max y, y indicates the fitness value of seeds and y_t indicates the threshold.

Step 3: Sort the seeds in order of fitness.

Step 4: Produce seeds from the best individuals where best fits produce more seeds

Step 5: Pollinate seeds in the neighboring space:

$$S_v = U * s \qquad \ldots(3.1)$$

Wherein, $0 \le U \le 1$. Given a circle radius a, for two plants X j and X k, they will be neighbors to each other if they meet the following (3) and thus determining the neighbor number j v of each plant and the maximum neighbor number of the plant in the same generation is expressed as max v, the (4) is as follows:

$$u\left(X_j, X_k\right) = \left\| X_j - X_k \right\| - a < 0 \qquad \ldots(3.2)$$

$$U_j = e^{\left[\frac{v_j}{v_{max}} - 1\right]} \qquad \ldots(3.3)$$

Step 6: Disperse the plants with some function:

According to Gaussian distribution, the next generation of seeds produced by each plant is scattered in the parameter space, the positions of seeds are expressed as:

$$X_{seed}^{i+1} = N(x^i, \sigma) \qquad \ldots(3.4)$$

σ is the coefficient of dispersion, which can determine the dispersion degree of produced seeds.

Step 7: Reach termination if one of the termination conditions is met.

Otherwise, repeat from Step 2 Termination condition used by us was a time boundary of 8 hours.

$$S_{max} = S_1 * S_2 \ldots S_n \ : E(n) < 480 \, mins \qquad \ldots(3.5)$$

E(n) is the event after n iterations.

Step 8: Produce output as S_{max} when the termination condition is met.

We can also do a pattern search in the given algorithm if we have enough computing resources.

3.2.1. Dataset Pre-Processing

We used the Google Landmarks dataset available at https://www.kaggle.com/c/landmark-recognition-2021 for doing Network Architecture Search (NAS). It is a dataset comprising around 1.59M images including the ones shown in Figure 2. However, training on such a huge dataset would have taken extremely long. So, for the evolution of the network, we used only 600 images out of the dataset. Also, it was seen that one landmark class had a very small number of images, hence the data was augmented 15 times to about 9000 images. The data was augmented in terms of translation, rotation, and scaling. cropping. flipping, etc. Augmentation paved the path for improved generalization of neural networks.

Each picture was tested to be in an RGB color encoding scheme to ensure proper dimensions in colors. After this, each picture was tested to be of size (64,64). An image of inappropriate size was resized to the given dimensions.

Figure 3.2: Landmark Images from Google Landmarks Dataset V2 [8].

Then the entire dataset was processed so that the input variable had the tensor of shape:

$$x = (9000,64,64,3) \qquad \ldots(3.6)$$

The output tensor had the following shape:

$$y = (9000,24) \qquad \ldots(3.7)$$

This meant that the input tensor had 110,592,000 parameters in total. The output had to be one out of 24 classes as depicted by the output tensor. The dataset was converted into the input tensor variable and the output tensor was assigned one out of 0-23 numbers for each of the 9000 images. After this, the x tensor and y tensor were used for training. The TensorFlow library by Google was imported for creating tensors. However, the overall algorithm was self-designed.

3.2.2. Experimentation and Results

We ran the experiment on a laptop Asus VivoBook S15 with 8 GB RAM, 4 GB NVIDIA Graphics Card, and an intel CORE i7 processor. A self-designed code was run in a loop indefinitely for 8 hours. In the code, a 3-layer CNN is designed which is then evolved with the Paddy Field Algorithm and the results are observed.

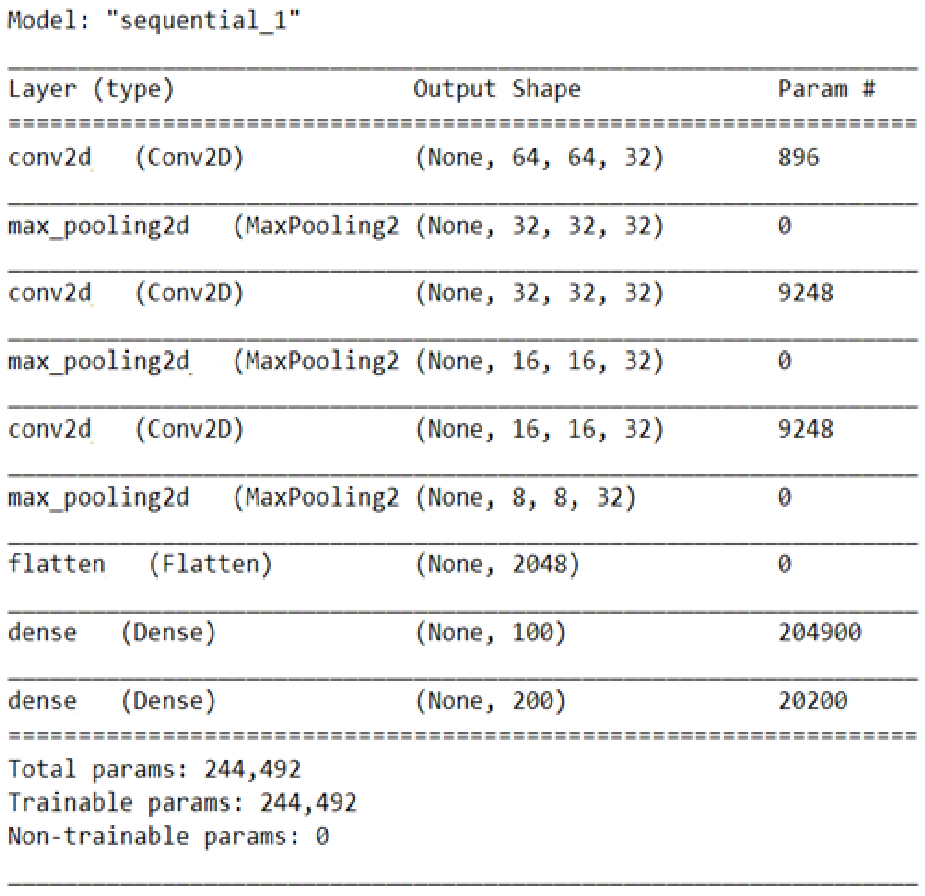

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d   (Conv2D)            (None, 64, 64, 32)        896
_________________________________________________________________
max_pooling2d   (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d   (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d   (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d   (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
max_pooling2d   (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
flatten   (Flatten)          (None, 2048)              0
_________________________________________________________________
dense   (Dense)              (None, 100)               204900
_________________________________________________________________
dense   (Dense)              (None, 200)               20200
=================================================================
Total params: 244,492
Trainable params: 244,492
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.3: The basic architecture of the designed CNN.

As shown in Figure 3.3, the designed CNN consisted of three convolutional layers which were followed by three max-pooling layers all of which are afterward followed by a fully connected layer with 100 neurons which are varied afterward, and 200 neurons which are kept constant.

Here T and F stand for True and False respectively, C_n stands for the nth class. Accuracy is one of the performance parameters which decide the goodness of a CNN. The complete hyperparameters that were evolved can be described as:

1. **Kernel Frame Size**: 3 x 3 kernel frames are considered highly optimal for CNNs. However, varying the kernel frame size, we saw that 7 x 7 was the kernel size for the best-fit seed. It performed the best with other arrangements of hyperparameters. The kernel frame sizes chosen were between 1 * 1 and 11 * 11. These were in the form of square matrices.

2. **Number of Kernels**: The number of kernels was varied to check the best number that could be checked for within 22 and 42. It is said that 32 or 64 kernels seem to work well but for us 42 was the variant in the best seed. Other numbers could have worked even better if we had increased the search space.

3. **Learning Rate:** The learning rate is the speed with which the network trains itself. With a slower learning rate, a network can achieve better accuracy, but it increases its chances of running into a local minimum. It also takes more time to run. A fast learning rate will quicken the rate of learning and run into a problem of deviation from the global-minima. A learning rate of 0.01 is considered optimal for usual cases, and indeed 0.099 was the best parameter found. The learning rate was varied between 0.001 and 0.99.

4. **Batch Size:** Batch size is the number of images given to the network for training in one go. A batch size of 32 is considered good and it was found that 32 is optimal and seemed to perform well. A little variation was found good in the batch size which included 33 and 34 even though it varied between 22 and 42.

5. **Neurons:** Neurons were also varied to check for the best type of connections. The initial 100 neurons were altered between 90 and 110 and results showed that many variations lying in this range seemed to do well though the end best fit was 102.

When the CNN is run the backpropagation runs in the following form:

$$W_x^{new} = W_x^{old} - a\frac{d(Error)}{dW_x} \qquad \ldots(3.8)$$

where, W_x^{new} are the newly trained weights, W_x^{old} are the old weights, a is the learning rate, and $a\frac{d(Error)}{dW_x}$ is the change an error with respect to the weights. It is run with AdaDelta optimizer using Keras in-built libraries for optimal change of weights [125]–[129].

In the code, the CNN hyperparameters were replaced by variables, and these variables were initially assigned default CNN hyperparameter values. The accuracy was measured in the first place. Then these variables were varied by the means of the Paddy Field Algorithm in the search space formed and the accuracies were consistently recorded. The best seed found based on accuracy was [7,42,0.0099,32,102]. This was in the Format [Kernel Frame Length, Number of Kernels, Learning Rate, Batch Size, Neurons].

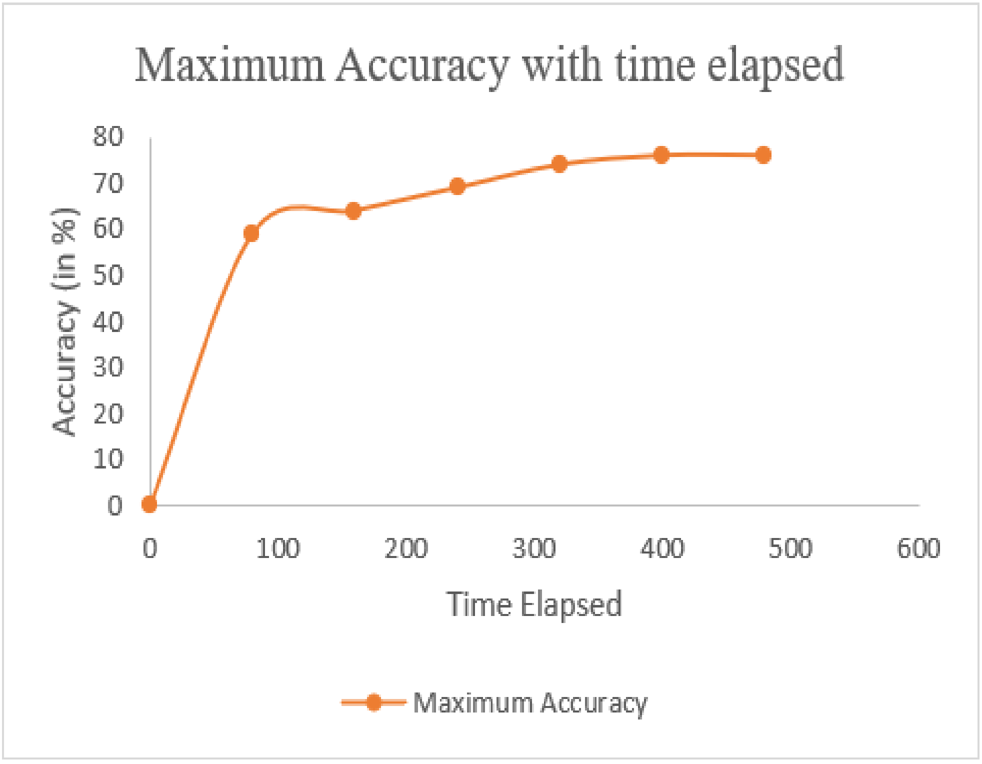


Figure 3.4: Accuracies of evolved PFANET variants as time elapses

Figure 3.4 describes how the maximum accuracy of the evolved networks elapsed over time. Figure 3.5 shows the accuracy of checked seeds as time progresses. It gives an idea of how the accuracy changes with the number of seeds that were checked. Figure 3.6 shows the accuracy change from start to finish as time progresses when a default CNN is compared to evolved CNN.

.

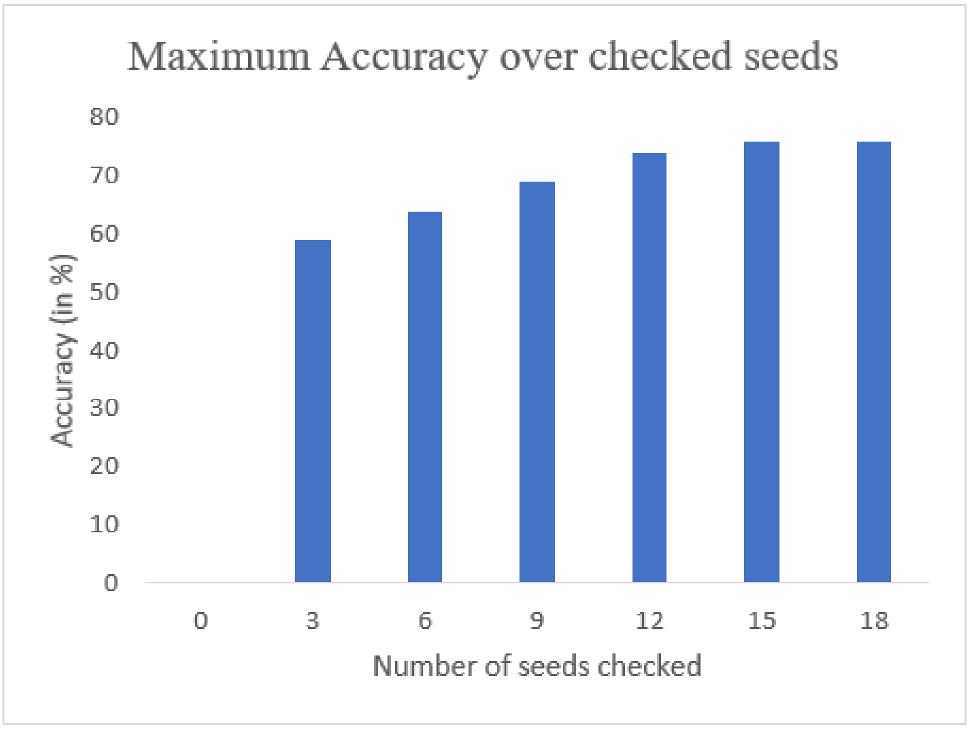


Figure 3.5: Accuracies for the evolved seeds in the network.

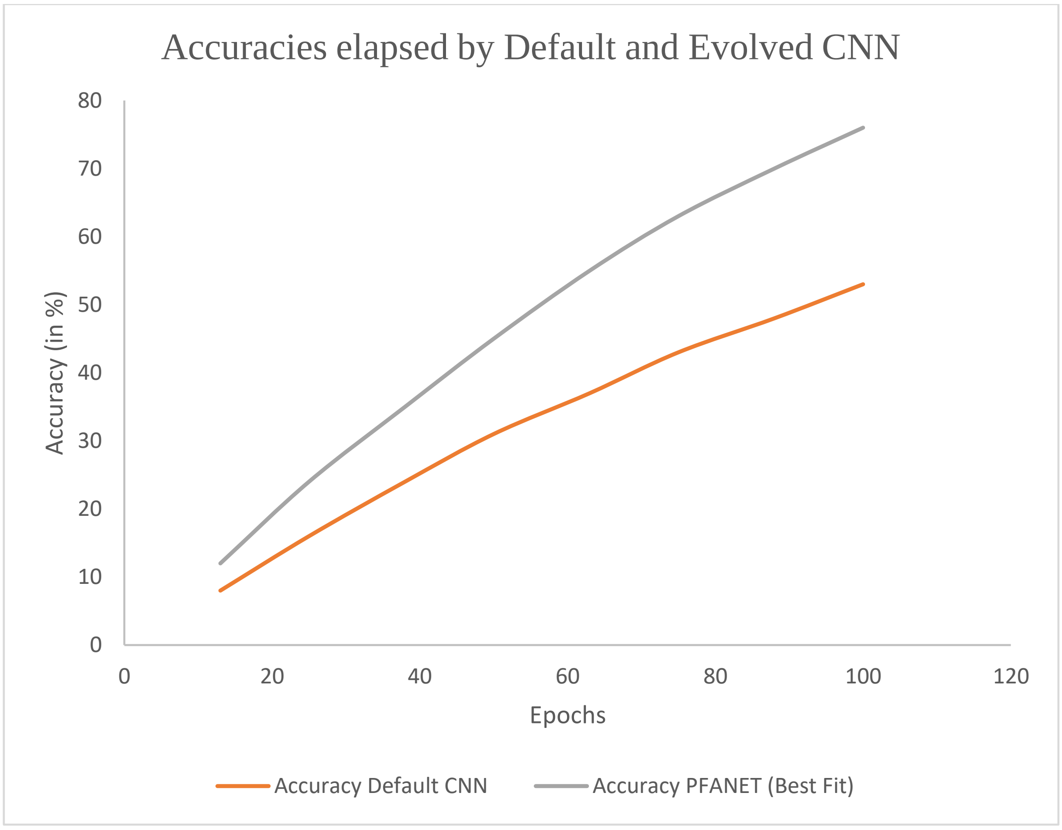


Figure 3.6: Accuracy change in evolved CNN vs. Default CNN.

Table 3.1: Accuracies in a default CNN and the network evolved with PFA compared over best fits and shown as steps of time and epochs elapsed.

			Accuracy	
S.No.	**Epochs**	**Time (in mins.)**	**Default CNN (approx. in %)**	**PFANET (Best Fit) (approx. in %)**
1	13	5	8	12
2	25	10	16	24
3	38	15	24	35
4	50	20	31	45
5	63	25	37	55
6	75	30	43	63
7	88	35	48	70
8	100	40	53	76

Table 3.1 gives an insight into how well the self-designed PFANET fared when compared to a regular CNN with default hyperparameters. Each network has been compared based on the approximate accuracy gained by each network over time. As it is apparent, the best fit PFANET variant gains more accuracy in shorter time frames as compared to a regular CNN. The key observations that should be made are:

- The best kernel frame length is 7 while the normally used length was 3. A kernel frame length of 5 also performed better than 3 in some combinations. One striking observation was that an unusual kernel frame length of 4, which is not expected to be good because of an even kernel frame, also performed better in many regards.
- The number of kernels after optimization was 42 in the range of 22-42 meaning the maximum number of kernels improved accuracy. However, more research would be required to validate how many kernels are optimum.

- The usually used learning rate is 0.01 and there was a good observation that the best learning rate came out to be 0.0099 which is almost 0.01.
- The number of optimum epochs is 100 but that was chosen manually since the network did not require much processing in one go
- The best batch size that was seen was the regularly used 32.
- The neurons did not seem to vary much, and the best number was 102 when initialized with 100.
- The code was run for 8 hours and as many as 18 seeds were checked over a wide variety of combinations.
- In 18 seeds only, the accuracy improved considerably from 53% of default CNN to 76%.
- The experiment showed that the Paddy field Search Algorithm is a very viable evolutionary metaheuristic for searching best-fit hyperparameters.

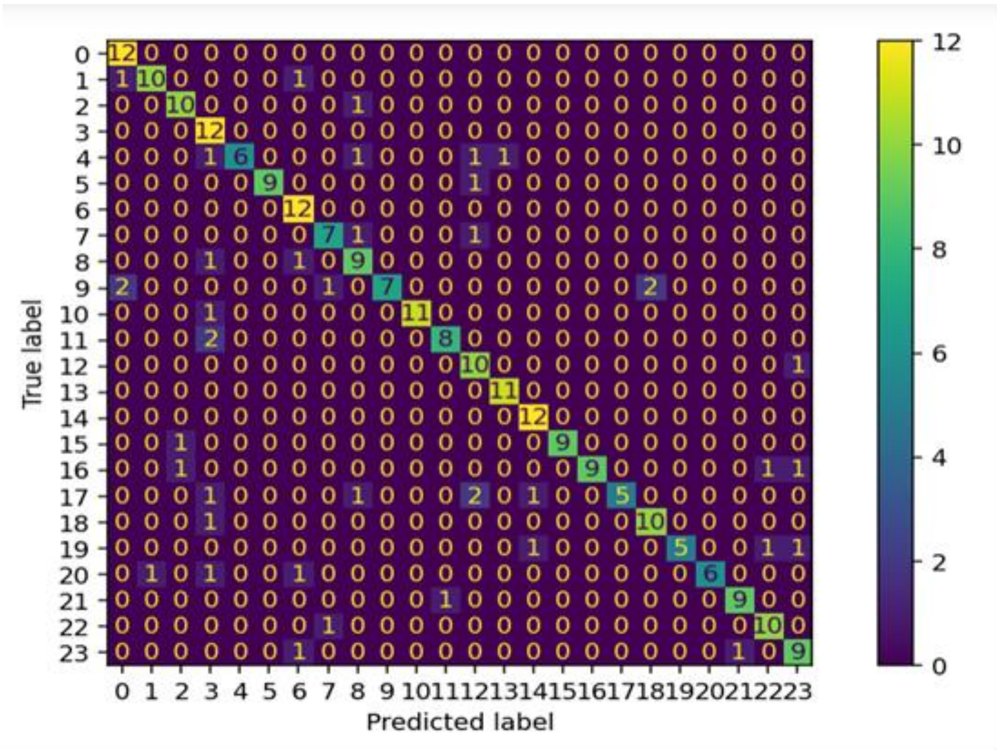


Figure 3.7: Confusion matrix for the trained model.

Figure 3.7 shows the confusion matrix developed for the trained CNN with evolved hyperparameters. Figure 3.8 shows the best-evolved hyperparameters. The study concluded that there can be a high change in the performance of a CNN when evolved with an evolutionary metaheuristic such as Paddy Field Algorithm.

7	42	0.0099	32	102
Kernel Frame Length	Number of Kernels	Learning Rate	Batch Size	Neurons

Figure 3.8: The best-fit seed after the evolution of CNN architecture with PFA.

We see how to evolve a Convolutional Neural Network with the Paddy Field Algorithm (PFA). We see that there is a big hyperparametric space to search for the best combination in a CNN and the evolutionary metaheuristics help us find the best possible combinations out of the big search space. It was found that high performance was recorded by the hyperparameters found by the Paddy Field Algorithm and the accuracy improved a lot when the CNN was trained. We also see that the hyperparameters, very different from the regularly used hyperparameters, performed much better. We drew a comparison between default CNN and evolved CNN and the evolved CNN or PFANET seems to perform much better. However, more research is required to establish if a particular hyperparameter can work with all combinations. Moreover, the evolution of networks can take too much time and we need to improve algorithms to evolve the ANNs quickly and with stable hyperparameters. Data augmentation helps in the overall improvement of the network.

3.3. CNN Evolution with GA and 3PGA

Massive amounts of data are continually being thrown at Computer Vision (CV). Imagery data is one of the most difficult types of data for an artificial intelligence (AI) system to process. Convolutional neural networks (CNNs) are used to handle this kind of Big Data, although advancement is slow. This study uses the 3 Parent Genetic Algorithm (3PGA), an evolutionary computation technique, to create a default CNN. An extension of GA called 3PGA has been improved for better optimization.

According to the literature, 3PGA performs far better on common benchmark functions than other more modern soft-computing-based methods. With a net improvement of more than 40%, the evolved CNN's accuracy rose from 53% to 75%. It was also found that a CNN's hyperparametric combinations or features, which are extremely different from those that are frequently used, appear to perform better. For testing, a Google collection of geographic markers was utilized. The optimization of a network on a landmarks dataset demonstrates that evolutionary computation can be significantly used in the future for the evolution of Artificial Neural Networks (ANNs), which is one of the most time-consuming tasks for an AI system.

Convolutional Neural Networks or CNNs are one of the most widely used machine learning architectures today. Almost all major projects in computer vision are built using CNNs and their subtypes today. CNNs were introduced in the 1980s by a then post-doctoral research fellow Yann LeCun [130] Since then, a lot of research has been done on CNNs, and they have emerged as the best possible solution for computer vision tasks today. The extreme flexibility of CNN architectures makes them well-suited for the broad-ranging tasks in today's world.

Numerous hyperparameters in a CNN can be optimized to get the best performance out of it. CNNs are highly flexible and highly efficient in delivering accurate image recognition, retrieval, and segmentation tasks. CNNs or Convolutional Neural Networks work on the concept of convolutions of images with trained kernels. In a specialized dot product, the convolution operation takes two apparent images and produces a new image out of them.

The original image is convolved with a trained microscopic image or set of images (mostly 3 x 3 pixels), and a network is trained. The network or model is trained using activation functions known as activation functions and linking all the original images to their respective classes via a network of activation functions. In the experiment performed, the task of the evolution of the hyperparameters with metaheuristics was undertaken in a Convolutional Neural Network. For this purpose, a dataset related to landmark recognition was deployed.

Evolutionary meta-heuristics are nature-inspired search algorithms that work based on inspiration from some natural phenomenon. These algorithms derive their working from evolutionary processes in nature like reproduction, the creation of the universe, ant behavior, etc. Based on the inspiration, the algorithms that followed are the Genetic Algorithm (GA), Big Bang Big Crunch (BBBC) algorithm, and the Ant Colony Optimization (ACO) algorithm. The 3-parent genetic algorithm is another metaheuristic that derives its inspiration from the reproduction phenomenon but is just a little different from GA [84], [89], [131].

A genetic algorithm in nature-inspired computing is a meta-heuristic used to search for optimal solutions to complex problems in the least possible time. The criteria they use for working is the reproductive process in living organisms. 'Survival of the fittest' is a famous statement and the genetic algorithm finds the optimal solutions just like nature finds the best living beings through evolution [86]. Genetic algorithms fall in the category of evolutionary algorithms. Three parents' genetic algorithm is a well-established nature-inspired-computing algorithm derived from the working of another well-established algorithm, namely the genetic algorithm. The genetic algorithm is a NIC algorithm that has been inspired by the natural process of reproduction in living beings. The genetic algorithm takes in an input of two-parent members and gives offspring, whereas the 3-parent genetic algorithm takes in an input of three parents instead of two [132].

A 3-parent genetic algorithm is used to optimize the hyperparameters in a CNN. It is discussed how the optimization of parameters leads to better accuracies in CNNs. The results of implementation have been discussed and it was inferred that 3PGA is one of the best meta-heuristics for bettering the hyperparameters. Also, 3PGA seems to fare better than the regular genetic algorithm. Like other neural networks, a CNN consists of layers of activation functions that are trained properly so that they get activated only upon reasonable information in the network. CNNs are a part of the Deep Learning paradigm that is used for heavy image processing. The designs and types of CNNs used for heavy imagery data processing are discussed in further sections.

3.3.1. CNN architecture

CNNs have an architecture like neurons in the human body that can help in the development of large-scale computer vision. What happens is that there is a network of activation functions that get activated on a particular input. These activation functions are primarily the dot product between image pixels and kernels in a CNN [133].

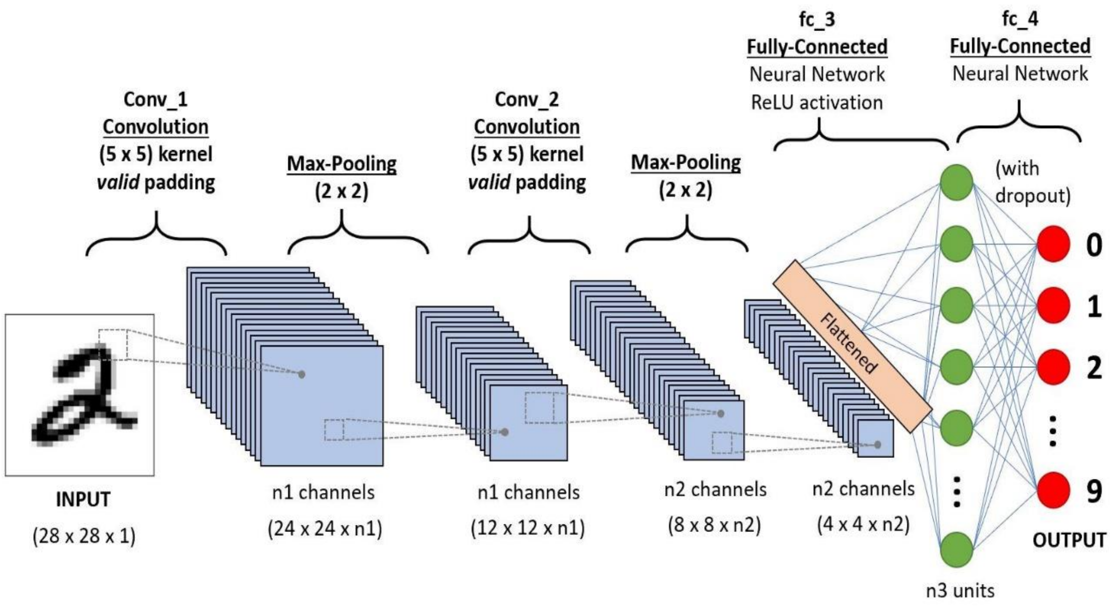


Figure 3.9: A simple CNN architecture [134]

The way this dot product acts as an activation function in Figure 2 is described as follows:

$$\mathrm{C}[\mathrm{m},\mathrm{n}] = \sum \mathrm{u} \sum \upsilon \mathrm{A}[\mathrm{m}+\mathrm{u},\mathrm{n}+\upsilon] \cdot \mathrm{B}[\mathrm{u},\upsilon] \qquad \ldots(3.9)$$

such that $\mathrm{C}[\mathrm{m},\mathrm{n}]$ stands for convolution operation at mth row and nth column.

u and υ stand for pixels of the kernel to be convolved.

A stands for Image with $\mathrm{m}+\mathrm{u}$ pixels length and $\mathrm{n}+\upsilon$ width.

After a Convolutional layer, a pooling layer is added to add non-linearity to the data. The pooling can be an average pooling layer, minimum pooling, or maximum pooling. Mostly, max-pooling (maximum pooling) is used for this purpose. The equation describing a max-pooling operation is as follows:

$$I_{l+1}^{1} = \max\,(I_l^2)\ \forall\, I_l^2 \in\ I_l^n \qquad \ldots(3.10)$$

where I_l^2 is the part of the image at layer l as a square of length 2 pixels.

After all the layers, fully connected layers are added to train the network into appropriate linear categories. The equation for the linear network layer can be given as:

$$I_l^n = A_{l+1}^{n*n} \qquad \ldots(3.11)$$

such that A is a 1-dimensional array of length n*n and l is the layer number.

In a typical CNN, the kernels act as an activation function and are trained with appropriate numerical values to activate when needed. Usually, the kernels in the first few layers in a CNN detect the most essential elements in a picture like borders, boundaries, and basic shapes. The following layers can evaluate complex objects like the nose, ears, and other small things, and the last layers can evaluate the complete subject like a person in an image [135].

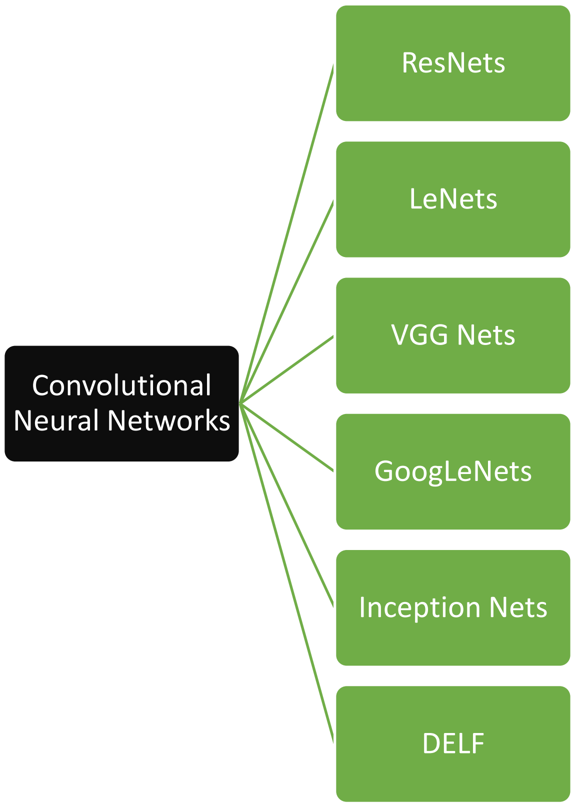


Figure 3.10: Sub-types of Convolutional Neural Networks

All the information travels back into the network in the backpropagation stage so that the associated data can be linked. However, there exist problems like linearity. To reduce the amount of linearity in the network, layers like Max-Pooling are introduced in the network [136]. This layer takes the maximum value from a group of pixels and creates a new image in the network. Then, the fully connected layers can connect all the related data points.

Convolutional Neural Networks or CNNs are a type of Artificial Neural Network (ANN) which are used for image recognition, retrieval, and classification [85]. Introduced by Yann LeCun in the 1980s, CNNs are the most used machine learning architecture in computer vision today.

Almost all the major computer vision models are designed using one or the other sub-architecture of CNNs today. There exist many CNN architecture sub-types today. These are summarised in Figure 3. It shows the various types of CNNs that have been developed to date. These architectures of CNN are categorized based on variations in their hyperparameters, primarily the number of layers. These CNN architectures vary significantly in the hyperparameters given to them. Usually, a greater number of layers in a CNN means greater accuracy. However, the ResNets developed in 2015 show that a different architecture for lesser layers can also help improve the accuracies [137]. It means that varying the hyperparameters in a CNN can improve its performance. All the above-given networks have been applied for usage for large-scale landmark recognition.

3.3.2. 3 Parent Genetic Algorithm: The Concept

There exist such problems in computing in which finding the exact, deterministic solutions is non-feasible. Here, the concept of soft computing comes in. Soft computing is that branch of computing that focuses on approximate, non-deterministic solutions that may or may not be the true solution but an optimal solution to the problem posed in front of us [91].

There are many types of soft computing today, including whole neural networking and evolutionary metaheuristics [138]. Metaheuristics are search algorithms that are used to find optimal solutions to seemingly complex computing problems. In other words, these algorithms are used to optimize solutions for complex computing problems, including the NP-complete and NP-hard problems [118].

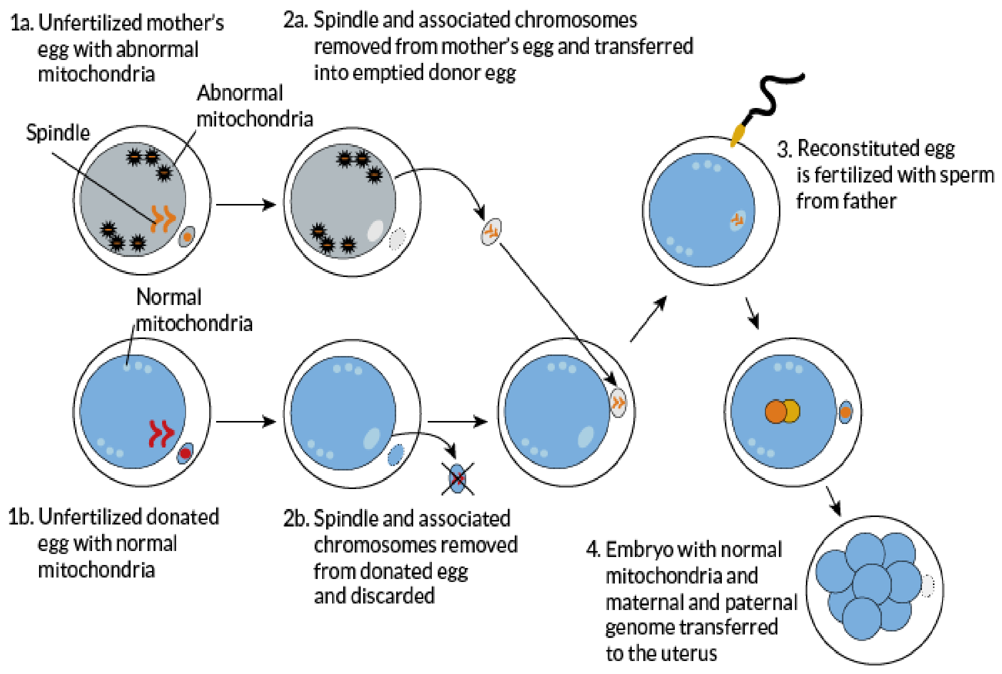

1a. Unfertilized mother's egg with abnormal mitochondria
2a. Spindle and associated chromosomes removed from mother's egg and transferred into emptied donor egg
Abnormal mitochondria
Spindle
3. Reconstituted egg is fertilized with sperm from father
Normal mitochondria
1b. Unfertilized donated egg with normal mitochondria
2b. Spindle and associated chromosomes removed from donated egg and discarded
4. Embryo with normal mitochondria and maternal and paternal genome transferred to the uterus

Figure 3.11: The process behind the 3-parent genetic algorithm.

Many types of optimization algorithms have been developed. These include the Big Bang Big Crunch algorithm, the genetic algorithm, ant colony optimization, the 3-parent genetic algorithm, and many more [87], [88], [139]. The 3-parent genetic algorithm is used for the optimization of a simple CNN architecture. A standard genetic algorithm finds out the best solutions to a problem by working in a way similar to the reproduction of organisms and is considered one of the best and most stable metaheuristics [140]. The best genes in a generation are carried forward, and the relatively poor ones are left behind. The usual crossover between two parents happens, and then mutation occurs [141]. However, this algorithm produces mutations at a relatively prolonged rate. This is where a 3-parent genetic algorithm comes in better.

Figure 3.11 shows the 3-parent baby concept. In 3-parent baby concept, crossover takes place between three parents instead of two. The purpose of this process is to generate the new offsprings from healthy mitochondria [142]–[144]. The concept of 3rd parent is to remove weak mitochondrial defects from a parent. Today, doctors produce a 3-parent baby by removing the maternal genome from a mother with abnormal mitochondria and replacing the genome of a healthy mother's egg. Then the resultant egg is fertilized by the father's sperm.

3.3.3. Proposed 3PGA-CNN Approach

In 3PGA, instead of the usual two parents, features from 3 parents are selected to go into the next generations. The concept of a three-parent baby was introduced in 2016 and it includes a process of fertilizing a donor female's egg with a father's sperm and then replacing the nuclei of the donor female with the nuclei of the mother in the embryo [145]. This process infuses the features of three parents and quickens the process of evolution. The complete process behind 3PGA is shown in Figure 3.12.

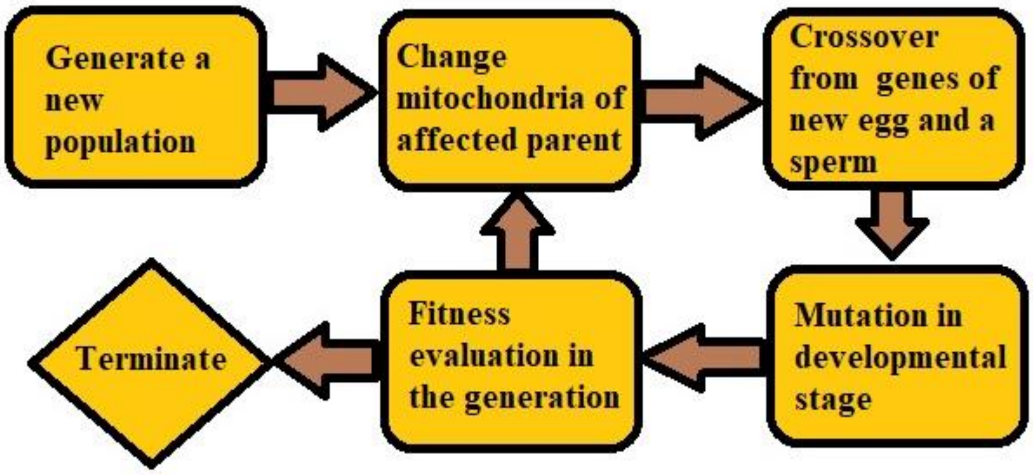

Generate a new population
Change mitochondria of affected parent
Crossover from genes of new egg and a sperm
Mutation in developmental stage
Fitness evaluation in the generation
Terminate

Figure 3.12: Implementation process of 3PGA explained.

The steps followed while implementing 3PGA can be given in Figure 3.13. As explained in Figure 3.13, to run a three-parent genetic algorithm, the first step is to generate a new population of at least three individuals. Then an affected egg is taken, and its mitochondria are replaced with mitochondria from a donor egg. Then the egg and sperm crossover, thus crossing genes from the chromosomes of these individuals. Then, mutations are brought into the developmental stage of progeny, and fitness is evaluated. The three from best fit then become the parents, and then the process repeats [146]. It is the same as reproduction.

3.3.4. Dataset Pre-Processing

Landmark Recognition is a central field in computer vision due to its imperative nature. Landmarks refer to any physically distinguishable object in any digital image that is expected to be identified by an AI system. Landmarks occur in almost all computer vision works except that the nature of landmarks varies. There are 48 classes of geographical landmarks through a dataset derived from Google.

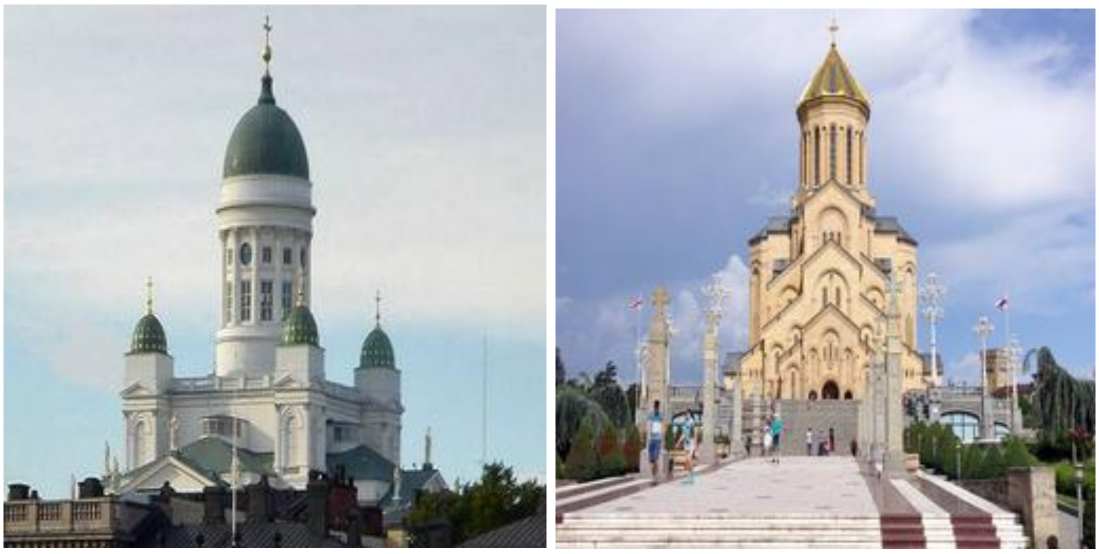


Figure 3.13: Two images from the Landmark dataset.

For the implementation of the proposed approach CNN optimization approach, a robust dataset was required. For this purpose, a very popular landmarks dataset from the literature was used. This is the Google Landmarks Dataset V2 [1]. Since the original dataset is extremely large and complex, just 48 classes out of the 200,000 classes of the dataset were used for experimentation. 12 images were kept in each class totaling up to 576 image files.

All the images have 3 color channels namely Red, Green, and Blue (RGB), and dimensions of (256,256) which were converted to (64,64). The images were placed in 48 folders with their respective class of landmarks. The shape of the X_train placeholder, that is the total set consisting of training and test images, is:

X.shape = (576,64,64,3) **…**(3.12)

The input set X was divided into two subsets namely X_train and X_test in the ratio of 70:30. This ratio is considered optimal in machine learning since it includes an optimal ratio of the test set data to test appropriately for over-fitting and underfitting. The shapes of X_train and X_test were now:

x_train shape: (403, 64, 64, 3) **…**(3.13)

x_test shape: (173, 64, 64, 3) **…**(3.14)

All the image datasets were converted from a pixelated format to a NumPy array with RGB values for each of the pixels. Every pixel contained one out of the 256 shades of Red, Green, and Blue. This amounts to more than 16M colors. The values for pixels were first normalized by scaling them to a range of (0,1) instead of (0,256).

3.3.5. Materials and Methodology

The designed CNN has an architecture with three convolutional and max-pooling layers and two fully connected layers. Also, the last fully connected layer has an output shape of (None, 48) since there are exactly 48 classes in the dataset.

In this experiment, many significant hyperparameters have been evolved for the betterment after which the performance of the models was evaluated. The hyperparameters were varied within certain specific ranges for each hyperparameter. The hyperparameters that were varied and evolved are:

1. **Kernel Frame Size:** It is a popular belief that 3 x 3 is the most optimal size for kernel frames. However, after varying the kernel frame size, it was seen that other frame sizes like 5 x 5 also do well and even better with certain specific arrangements of hyperparameters [147]. The kernel frame size was varied as a square matrix of lengths varying from 1 to 11.

2. **Number of Kernels:** It is also an arguable parameter to vary while designing a CNN architecture. Usually, it is said that 32 or 64 kernels seem to work well but it was discovered that unusual figures like 41 and 24 also seemed to do well [148]. The number was checked between 22 and 42.

3. **Learning Rate:** The learning rate is the speed with which the network trains itself. A slower learning rate means that a network can achieve better accuracy, but it can also run into local minima. Also, it takes more time to run. A fast learning rate will quicken the task of learning and run into a problem of deviating from the global minima. A learning rate of 0.01 is considered optimal for usual cases, but other learning rates also did well [126]. The learning rate varied between 0.001 and 0.99.

4. **Batch Size:** Batch size is the number of images given to the network for training in one go. A batch size of 32 is considered good and it was found that 32 is optimal and seemed to perform well [149]. A little variation was found good in the batch size which included 33 and 34. It varied between 22 and 42

5. **Neurons:** Another hyperparameter that was found worth evolving was the number of neurons in a layer. 100 neurons were altered between 90 and 110 and results showed that many variations lying in the range seemed to do well [150].

There exist other hyperparameters that can be optimized. These include strides, padding, the number of layers, and the backpropagation optimizer. However, the above-listed 5 parameters only were chosen for optimization. So, the stride in the network was the usual *'1'*, the padding used was *'Same',* and the number of layers was '3*'* [140], [151], [152]. Alongside that, the backpropagation optimizer used was *'AdaDelta'.* There exist many backpropagation optimizers in the literature today. These include Simple Gradient Descent, Stochastic Gradient Descent, RMSProp, AdaGrad, AdaM, and AdaDelta [125], [127]–[129], [153], [154].

The usual backpropagation equation is:

$$W_x^{new} = W_x^{old} - a\frac{d(Error)}{dW_x} \qquad \text{…(3.15)}$$

where, W_x^{new} are the newly trained weights,

W_x^{old} are the old weights,

a is the learning rate, and

$\frac{d(Error)}{dW_x}$ is the change is an error with respect to the weights.

The way these weights are updated defines the type of optimizer used by us. And the AdaDelta optimizer is used for the CNN architecture which is fixed and does not vary. Backpropagation is the actual step where all the training takes place.

3.3.6. Experimentation

Figure 3.15 describes how the hyperparameters of the designed CNN evolved with two well-known nature-inspired-computing algorithms namely the genetic algorithm and the 3-parent genetic algorithm. The choice was based on the feasibility and usability of algorithms. The proposed approach was implemented in Python language using the Jupyter Notebook program and the program was tested using an Asus Vivobook with a Core i7 processor, 8GB RAM, and an Nvidia GPU with a 2 GB graphics card.

Reproduction from two parents can be a slow process to run the process of evolution. Here is where the concept of a third parent comes in and helps. The third parent increases the randomness in the population even when it would be highly fit. All this procedure was executed with the help of Keras-built CNN and a 3 Parent Genetic Algorithm.

A population of eligible chromosomes was developed each containing five genes namely kernel frame length, number of kernels, learning rate, batch size, and neurons. After this, the fitness function evaluating the fitness of each individual was run. The evaluation metric was accuracy for this purpose [119]. The individuals with the best accuracy in training the network were kept for future generations and the recessive ones did not make it further.

A 3-parent genetic algorithm was used to extract the best hyperparametric features out of the population of eligible chromosomes of neural network architectural features. The base 3 parent population of the regularly used hyperparameters was used to generate offspring and the best offspring were evaluated based on the accuracy of the network architecture, thus produced.

The figure 3.14 shows the working of proposed 3PGA based approach to evolve the near-optimal architectures of CNN for landmark recognition. The proposed approach starts with random generation of CNN hyperparameters population generation. This population is called as two-parent population. For, 3 parent population generation purpose, we add or subtract a small random number in current population and generate a new population. This population is integrated with existing two parent population and three parent population is generated. Further, we apply all the genetic operations to evolve the neural network architectures. This process continues until termination criteria is met. The proposed approach would be terminated if any one of following conditions is true:

A. Maximum number of iterations reached.

or

B. Desired performance of the proposed approach is achieved.

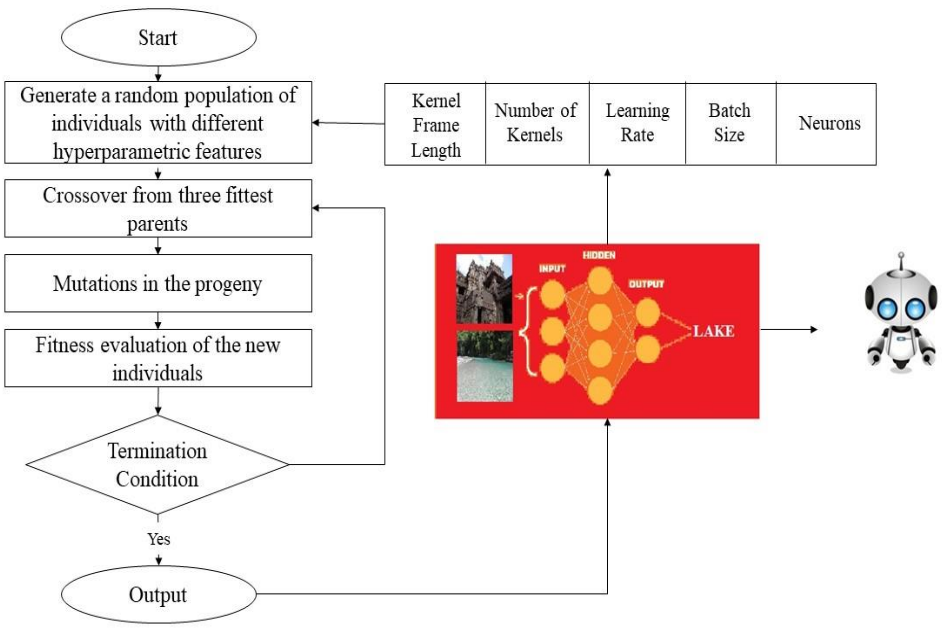

Start
Generate a random population of individuals with different hyperparametric features
Crossover from three fittest parents
Mutations in the progeny
Fitness evaluation of the new individuals
Termination Condition
Yes
Output
Kernel Frame Length
Number of Kernels
Learning Rate
Batch Size
Neurons
INPUT
HIDDEN
OUTPUT
LAKE

Figure 3.14: Working of the 3PGA optimization process.

3	**32**	**0.01**	**32**	**100**
Kernel Frame Length	Number of Kernels	Learning Rate	Batch Size	Neurons

Figure 3.15: A typical chromosome in the population

The base variant of chromosomes in the population looked like the one in Figure 3.15. During mutation, the kernel frame length varied from a size of merely 1 x 1 to 11 x 11. Similarly, the number of kernels and batch size varied between 22 and 42. The learning rate varied between 0.001 and 0.999 and neurons varied from 90 to 110 randomly in one iteration.

For, a regular genetic algorithm, the two best-fit parents were selected for crossover to produce the next generation. However, for a 3-parent genetic algorithm, three best-fit parents were selected for crossover. After the crossover and mutation, progeny is again evaluated for fitness and the best-fit progeny makes it to the next generations. The code is run in an infinite loop till convergence is not apparent.

3.3.7. Results and Discussions

The code optimizing CNN hyperparameters with the 3 Parent Genetic Algorithm was executed and some striking observations were made. For 100 epochs in the model, the best accuracy was observed not in the base variant chromosome but in a different chromosome. Even the expected kernel frame size was not the best kernel frame size. Instead of 3 x 3, the best kernel frame size came out to be 5 x 5. Figure 9 shows the hyperparametric space that was found after the code seemed to run into convergence. Table 2 shows all the best-fit chromosomes and accuracies.

3	**32**	**0.01**	**32**	**100**
Kernel Frame Length	Number of Kernels	Learning Rate	Batch Size	Neurons

5	**24**	**0.02**	**34**	**107**
Kernel Frame Length	Number of Kernels	Learning Rate	Batch Size	Neurons

5	**41**	**0.94**	**33**	**105**
Kernel Frame Length	Number of Kernels	Learning Rate	Batch Size	Neurons

4	**32**	**0.94**	**34**	**104**
Kernel Frame Length	Number of Kernels	Learning Rate	Batch Size	Neurons

Figure 3.16: The base variant of chromosome and some chromosomes that performed better than the base variant.

The base variant of the chromosome [3, 32, 0.01, 32, 100] shown in Figure 3.16 gave an accuracy of 53%. However, some chromosomes were seen to perform better. These are the chromosomes as shown in Figure 3.17. The first chromosome is the base variant and the rest are the chromosomes that were found upon the apparent convergence of code. The results were noted when the algorithms that were deployed seemed to reach convergence and they are meticulously depicted in Figures 3.18 and 3.19.

Table 3.2: Best Fit chromosomes in 3PGA (left) and in GA (right) as [Kernel Frame Length, Number of Kernels, Learning Rate, Batch Size, and Neurons] and their approximate accuracies.

3PGA		**GA**	
Chromosome (in 3PGA)	**Accuracy**	**Chromosome (in GA)**	**Accuracy**
[5, 24, 0.02, 34, 107]	75.24%	[3, 25, 0.06, 33, 100]	71.05%
[5, 41, 0.94, 33, 105]	72.36%	[3, 38, 0.06, 33, 100]	68.24%
[4, 32, 0.94, 34, 104]	69.13%		

Table 3.3: Accuracies for the Classical Machine Learning models namely SVM, KNN, Decision Tree, and Random Forest on Landmark Recognition dataset.

ML Algorithm	**Accuracy**
SVM	55%
KNN	48%
Decision Tree Classifier	42%
Random Forest Classifier	54%

For a comparison of these accuracies with the classical Machine Learning counterparts, the codes for four algorithms namely Support Vector Machines (SVM), K Nearest Neighbours (KNN), Decision Tree Classifier, and Random Forest Classifier were run and the results are noted as shown in Table 3.3.

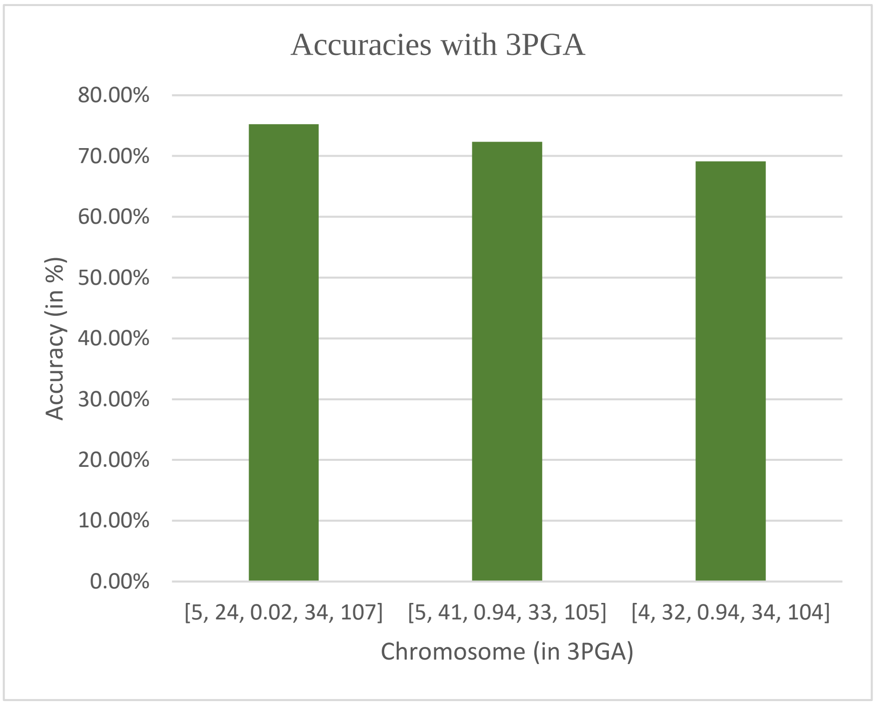


Figure 3.17: Best fit chromosomes as [Kernel Frame Length, Number of Kernels, Learning Rate, Batch Size, and Neurons] in 3 Parent Genetic Algorithm with their respective accuracies.

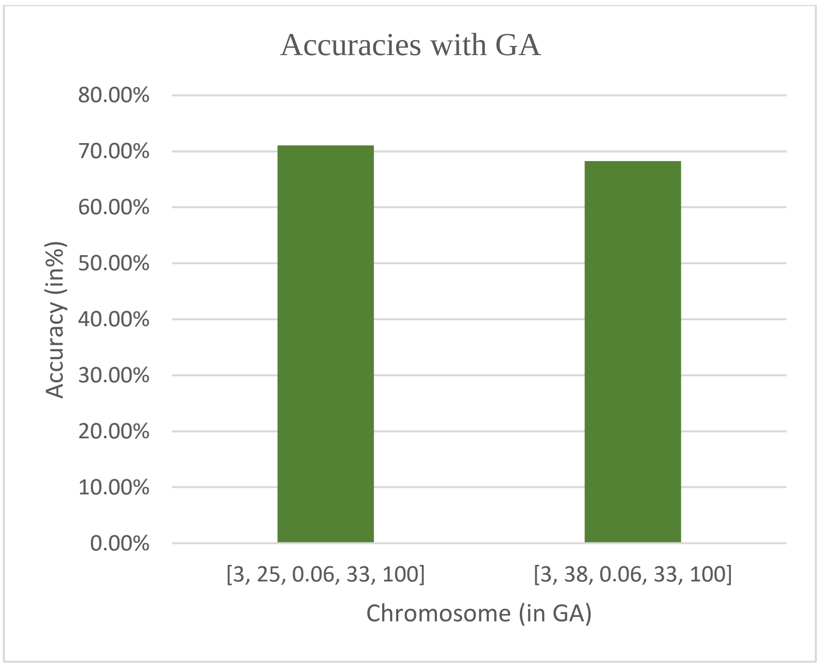


Figure 3.18: Best fit chromosomes as [Kernel Frame Length, Number of Kernels, Learning Rate, Batch Size, and Neurons] in Genetic Algorithm with their respective accuracies.

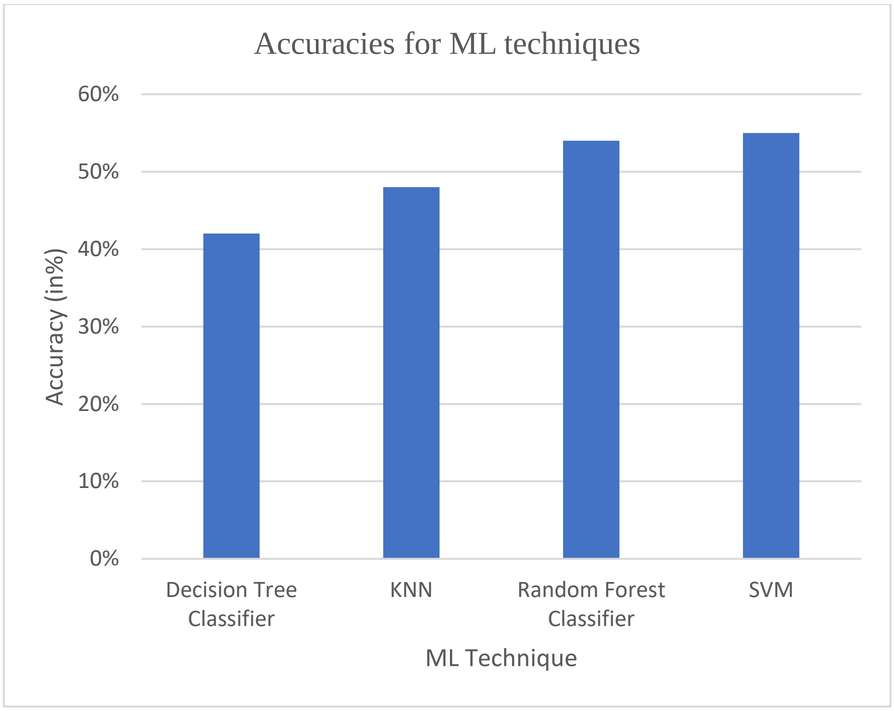


Figure 3.19: Various ML techniques with their respective accuracies for the Landmark Recognition task.

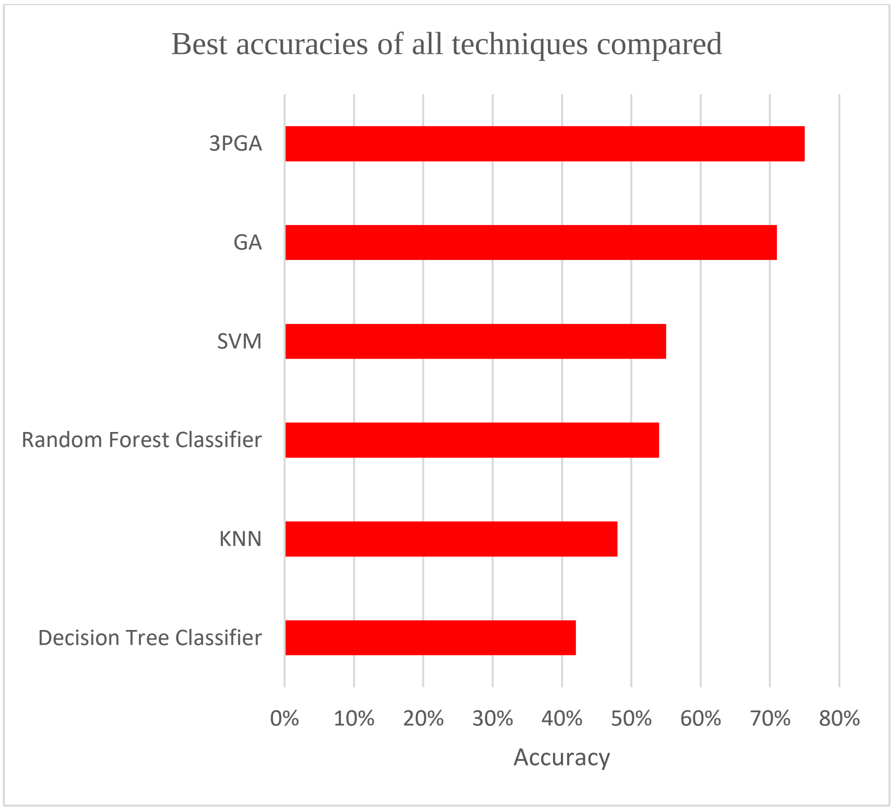


Figure 3.20: Best Accuracies from all the techniques compared.

As it is apparent from Table 3.3, Figure 3.20, and Figure 3.21, SVM performs the best in the Landmark Recognition task with an accuracy of 55%. It is followed by the Random Forest Classifier with an accuracy of 54% which is followed by the KNN and Decision Tree Classifier with accuracies of 48% and 42% respectively [155].

When a CNN is designed with the default or most used hyperparameters, referred to here as the base variant, an accuracy of 53% with 100 epochs was observed. This accuracy was bettered using metaheuristics namely the Genetic algorithm and its variant named 3 Parent Genetic algorithm.

Table 3.4: Training process of Default CNN and the Best fit chromosomes in GA and 3PGA.

		Default CNN		**Genetic Algorithm [Best Chromosome]**		**3 Parent Genetic Algorithm [Best Chromosome]**	
S.No.	**Epochs**	**Accuracy**	**Time (in mins)**	**Accuracy**	**Time (in mins)**	**Accuracy**	**Time (in mins)**
1	0	0	0	0	0	0	0
2	10	7	15	9	15	10	15
3	20	15	30	17	30	19	30
4	30	22	60	26	60	28	60
5	40	28	90	35	90	37	90
6	50	34	120	43	120	45	120
7	60	40	150	50	150	52	150
8	70	44	180	56	180	59	180
9	80	48	210	62	210	65	210
10	90	51	240	67	240	70	240
11	100	53	300	71	300	75	300

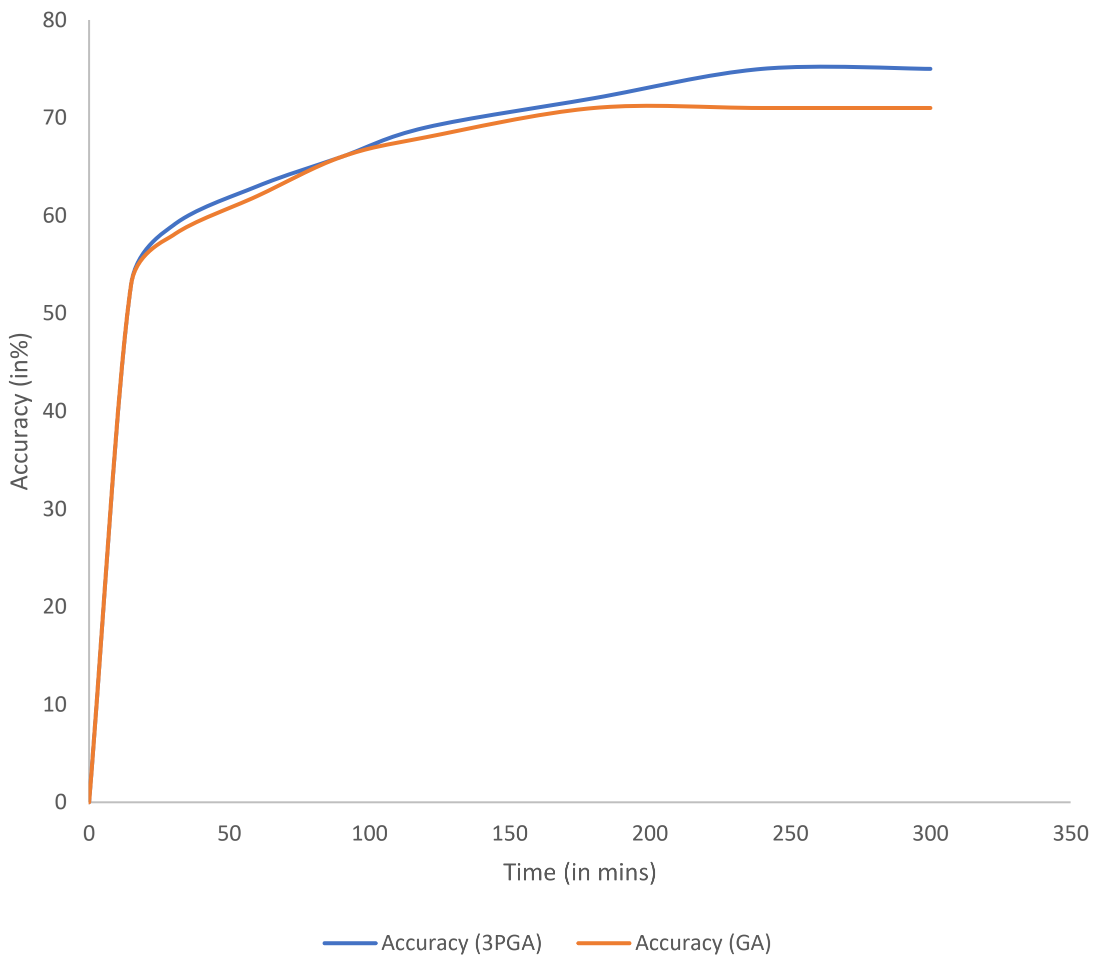


Figure 3.21: Variation in accuracy with time for optimization with 3PGA and GA.

The hyperparametric space was found after the evolution of CNN architectures as explained in Table 3.4 and the following observations were made:

- Figure 3.17 shows the best-fit hyperparameters and the basal variant chromosome. Table 3.2 summarizes these hyperparameters with their accuracies.
- The number of epochs was taken to be 100 and accuracy mostly showed an increasing trend by the time these epochs were completed.

- The dataset was reduced in size to allow room for more epochs and accuracy. With a larger dataset, the time for each iteration of 3PGA increased manifolds.
- Each iteration of 3PGA took three parents and tried to better these with three progenies. One iteration of 3PGA took approximately 15-20 mins while that of GA took around 10 mins. The program was run for 5 hours for both algorithms.
- The basal variant produced an accuracy of around 53% while the best-fit chromosome produced an accuracy of around 75% for 3PGA and 71% for GA.
- One striking observation was that the basal variant uses a 3 x 3 kernel frame size. But after optimization, the best frame size came out to be 5 x 5 and even some variants of 4 x 4 did better in 3PGA while GA was clogged with 3 x 3.
- 3PGA took longer for one iteration but it tended to introduce more variation and produced better results in the same time for which both the algorithms were run.
- Learning rates varied in a high range and many good variants used a learning rate of as high as 0.94.
- Batch size did not vary much and was seen to be around 32 only.
- The number of kernels and the neurons used varied highly with each better variant as expected.
- GA was a good choice for optimization but 3PGA did better in almost all respects.
- 3PGA proved to be a good choice for the evolution of a Convolutional Neural Network, giving at least 40% more average accuracy than the regular hyperparameters.
- An ANOVA test or analysis of variance provided insight that the number of kernels is the most significantly varying variable. It is also imperative since the number of kernels varied till 64 also gives good results whereas we varied it between 22 and 42 only.

The results showed us how a CNN can be evolved for better hyperparameters using a 3 3-parent Genetic Algorithm. The 3-parent genetic algorithm not only improved the accuracy of a CNN but also fared better than the Genetic algorithm with 2 parents for the evolution of the architecture of the CNN. One can safely assume that the hyperparametric space searched by these metaheuristics can help in improving the CNNs for even better performance. The code run for just 576 images found good hyperparameters which can be employed for further improvement of CNNs with a huge number of images.

With data augmentation on the available dataset, it was found that the accuracies reached as high as 92%, and with the application of 3PGANET, this accuracy improved to more than 98%. The hyperparametric space developed by the 3PGANET was used for training the Convolutional Neural Network (CNN), and the results as shown above were promising.

Further, it was found that by building up on a small part of the dataset, one can improve upon the hyperparametric space which can be then deployed for the complete large dataset. The default hyperparameters can be replaced by the hyperparameters found by the metaheuristics available in the literature and thus the accuracy is improved.

It has been seen in the literature that accuracy decreases miserably for larger datasets that include more than 100,000 images. For such datasets, it is highly advisable to find suitable hyperparameters first using the metaheuristics available in AI literature. The metaheuristics like GA and 3PGA can be applied to a small part of the dataset and best-suited hyperparameters are found.

The experimentation proves that the default or regularly used hyperparameters may not be the best-suited hyperparameters for some Machine Learning tasks. Thus, 3PGA has been used to find the best hyperparametric space for less than the 1000th part of Google Landmarks Dataset V2 and the accuracies have been improved. The found hyperparameters are then applied to the augmented dataset and test accuracies are found. The training and testing, both accuracies were better for evolved CNN as compared to default CNN. Hence it is advisable to evolve ML techniques with nature-inspired metaheuristics for better results.

3.4. Conclusion

Convolutional Neural Networks (CNNs) are one of the most widely used techniques in computer vision today. It is discussed that CNNs also have a chance of improvement through an automated evolution of their architecture. Nature-inspired computing helps in the evolution of CNN architectures by varying hyperparameters automatically. It was seen that a specific NIC algorithm named the genetic algorithm can substantially automate the evolution of a CNN. It was also seen that the introduction of a third parent helps hasten the process of evolution. It was found that CNN's usual hyperparameters can be evolved for better performance. Fine-tuning was performed and it produced strikingly different results and hyperparameters better than the regular hyperparameters were seen. Hence, based on the experiment, it is advisable that one should try evolving the hyperparameters on a part of the dataset before training an AI model on an entire dataset. However, the initial evolution of hyperparameters can become cumbersome and time-consuming. Also, we might not receive the expected results after the evolution of hyperparameters. Hence, more research is required to quicken the process of evolution, and that too in a stabilized manner.

Chapter 4: Parallel Bat Colony Optimization Algorithm: An Improved Metaheuristic for Global Optimization

A Parallel Bat Colony Optimization Algorithm based on the introduction of colonies to the usual Bat Algorithm is proposed in this chapter. Bat Algorithm is a popular evolutionary metaheuristic that is used to find solutions to NP-Hard problems like the Travelling Salesman Problem (TSP). The concept of colonies was introduced in the regular Bat Algorithm along with phenomena like communicative search and migration. The resultant Parallel Bat Colony Optimization Algorithm (PBCOA) was tested on the standard CEC Benchmark Functions, implemented in MATLAB, and the performance was compared to that of 17 other algorithms. On 6 test functions, the proposed algorithm showed the best performance in comparison with other algorithms. Additionally, it came out to be the sole best performer for CEC functions F9, F14, F15, F16, F17, F19, F22, F24, F25, and F27. The proposed approach performed the best overall across 17 compared algorithms in terms of finding the minimal cost to maximum benchmark CEC-14 functions.

4.1. Introduction

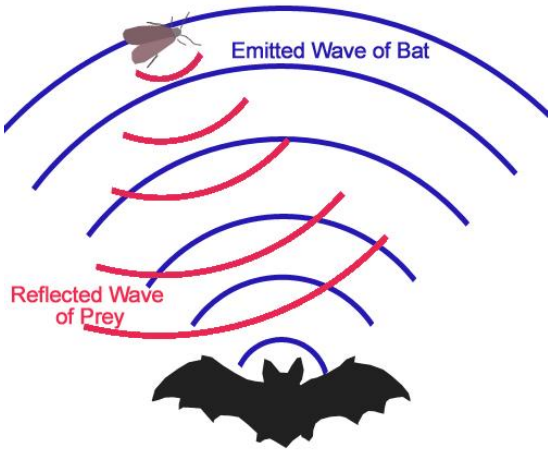


Figure 4.1: The bat sends ultrasound in the surrounding environment, and it reflects from the object. In this way, the bat finds prey and makes a virtual view of the world.

The Parallel Bat Colony Optimization Algorithm or PBCOA is based on the concept of the evolution of environmental conditions of bats while living in their Colonies. The bat Algorithm is a powerful algorithm designed by Xin She Yang in 2010 based on the echolocation behavior of bats. Bats are the only existent mammalian creatures that can fly. The extraordinary thing about bats is that they have very poor eyesight and rely on echolocation to find their way out in the wild. As shown in Figure 4.1, Echolocation is the phenomenon of the transmission of ultrasounds through the surroundings to communicate with the surroundings [156].

Figure 4.2: Actual visual of Bat Colonies.

The Parallel Bat Colony Optimization Algorithm (PBCOA) takes into consideration the all-around behavior of bats when they live in their colonies. PBCOA takes into account the generalized environmental conditions of a bat including food, warmth, moisture, proximity to their mates, etc. PBCOA works on the operations of echolocation using which the bat models its immediate surroundings. Then the communicative search is used which is the social communication calls within the bats. Bats use a variety of encoded roosts for communication of different environmental conditions [157]–[160]. Migration helps in the search for the best possible and safest surroundings for a bat to live in.

The ultrasounds are high-frequency sound waves that can go beyond 20KHz, which is the upper listening limit for a normal human. Bats transmit these ultrasounds in the environment and an echo of these sounds is heard by the bats. Bats receive these echoes and develop a worldly view of themselves. The existent bat algorithms, as shown in

Figure 4.2, only consider the echolocation behavior of a singular bat to develop the metaheuristic algorithm that tends to find a global optimum. However, bats always exist in colonies themselves. Here, the concept of the Parallel Bat Colony Optimization algorithm is a major improvement over existing Bat Algorithms [161].

The Parallel Bat Colony Optimization algorithm extends the Bat Algorithm to include colonies of bats instead of a single bat. The concept is to find the best environmental conditions for a bat which can be done inside one colony as well as many colonies at a time. The Parallel Bat Colony Optimization algorithm does not only include the echolocation behavior of bats but also the guided communication of bats within a colony and the migration of bats outside their current environmental condition to find better environmental conditions for themselves [100].

In the proposed algorithm, the best sustaining environmental conditions for a bat are found via echolocation first and then through the communications between the bats and also the migration of bats from one location to another. The communicative search follows almost the same procedure as the genetic algorithm. The migration follows the same procedure as Migratory Birds Optimization (MBO). Combined with echolocation, finding the best suitable environmental conditions for a bat becomes easy.

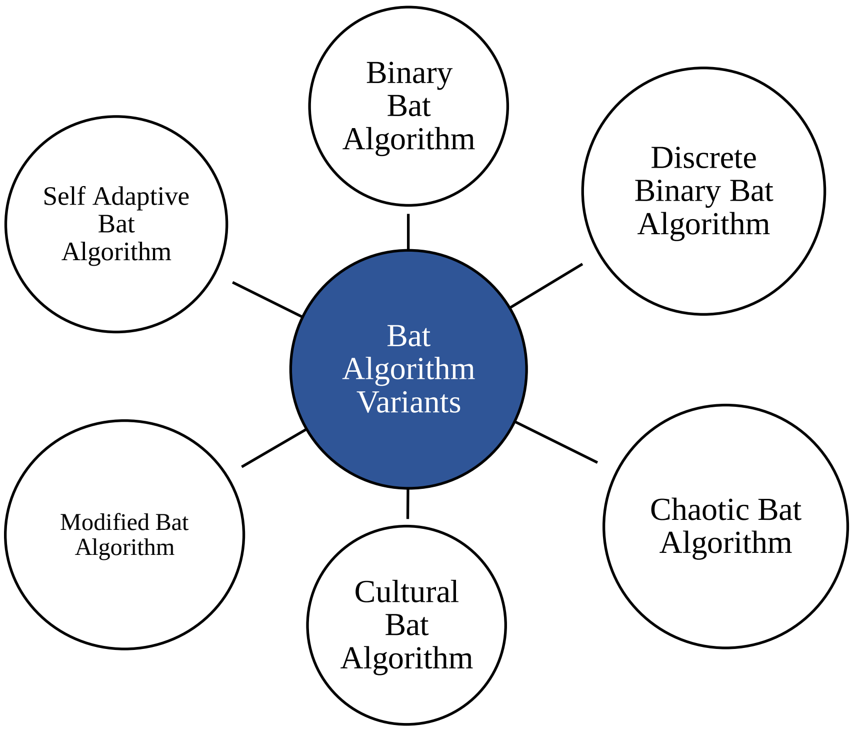


Figure 4.3: Various improvements in the Bat Algorithm.

In PBCOA, echolocation is not the only criterion for deciding where the bat will move. Reproductive crossover and migration additionally ease the exploitation and exploration processes. PBCOA works based on the introduction of 2-dimensional colonies to the regular Bat Algorithm. A singular bat is designed as a 1-dimensional object of N decision variables. The decision variables act as the environmental conditions of a bat that it might want to optimize. The algorithm works on the process of survival of the fittest whereby only the bats with the best or appropriate environmental conditions can survive.

A bat has very poor eyesight, and it relies on bio sonars for modeling the real world for survival. A bat sends ultrasounds in the environment to model the real world. Based on the model that it develops, bats forage into surroundings in search of the best possible locations to survive. Usually, bats live in colonies where they bring their food, reproduce, and live in a place that feels the safest to them. Based on this phenomenon, the parallel bat colony optimization algorithm is designed.

The environmental conditions of a bat act as the decision variables of the algorithm. The exploration process is guided by the migration of bats from one colony to another in search of better environmental conditions. The process of echolocation additionally helps in this process. The process of communicative search helps the bats in the exploitation process and it maintains the best features in bats, generation after generation.

Table 4.1: Possible decision variables of the Parallel Bat Colony Optimization Algorithm.

Var1	**Var2**	**Var3**	**Var4**	**Var5**
Preys	Mates	Predators	Warmth	Moisture

The constraints that the bats undergo are the movement constraints that decide how much energy, a bat can expend to reach the best possible and safest environmental situation. A bat cannot send infinite bio sonars. This provides the constraints to the algorithm to prevent it from webbing into an infinite loop. Under these constraints, the bat might want to optimize food choices, warmth, moisture, proximity to mates, safety from predators, etc.

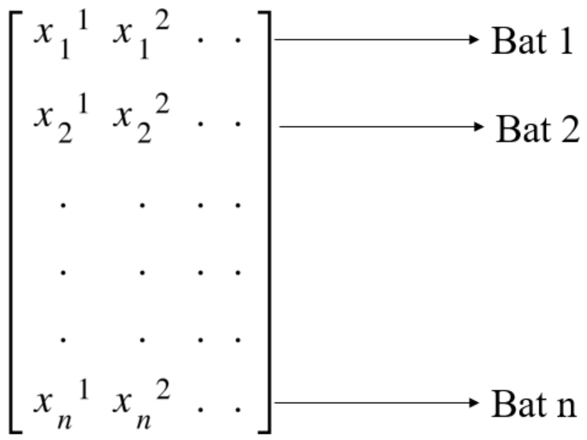


Figure 4.4: Representation of a bat colony. A tuple represents a single bat and its environmental features or decision variables are described by each entity of a single row.

PBCOA has been compared with 17 other algorithms including the United Multi-Operator Evolutionary Algorithms (UMOEAS) [162], LSHADE [163], Differential Evolution with Replacement Strategy (RSDE) [164], Memetic Differential Evolution Based on Fitness Euclidean-Distance Ratio (FERDE) [165], Partial Opposition-Based Adaptive Differential Evolution Algorithms (POBL_ADE) [166], Differential Evolution strategy based on the Constraint of Fitness values classification (FCDE) [167], Mean-Variance Mapping Optimization (MVMO) [168], RMA-LSCh-CMA [169], Bee-Inspired Algorithm for Optimization (OptBees) [170], Simultaneous Optimistic Optimization (SOO) [171], SOO+ Bound Optimization BY Quadratic Approximation (SOO + BOBYQA) [171], Fireworks Algorithm with Differential Mutation (FWA-DM) [172], algorithm Based on Covariance Matrix Leaning and Searching Preference (CMLSP) [173], Gaussian Adaptation Based Parameter Adaptation for Differential Evolution (GaAPADE) [174], Non-Uniform Mapping in Real-Coded Genetic Algorithms (NRGA) [175], and DE_b6e6rlwithrestart [176]. It is clear from Table 4.4 that PBCOA has outshone most other candidates in state-of-the-art technology. PBCOA emerged a clear sole winner on 10 out of the 30 test benchmark functions while it shared a draw with a 0 Fitness Cost on the other 6 benchmark functions. Thus, it performed best on 16 out of the 30 provided test benchmark functions.

4.2. Working Example

Let us assume that we have the following problem before us:

Statement: Find the minima to the function: $f(x)\colon x_1^2 + x_2^2 + x_3^2 \mid x_1, x_2, x_3 \in \mathbb{R}$.

Applied Approach: Let x_1, x_2, x_3 be the environmental conditions of the bats which will act as decision variables. We aim to assign four optimizable environmental conditions to bats and create bat colonies of multiple bats. Let us make 2 bat colonies using 3 bats each. Then we have to initialize the random population. Let the initial population be:

$$\begin{bmatrix} 4 & 9 & 1 \\ 2 & 4 & 6 \\ 5 & 3 & 8 \end{bmatrix} \quad \begin{bmatrix} 3 & 5 & 1 \\ 2 & 3 & 2 \\ 5 & 9 & 1 \end{bmatrix}$$

Now, communicative search works towards the exploitation process, and migration supports exploration. Each row in the colony matrices represents a bat with the decision variables as environmental conditions. Echolocation works towards getting the best possible environmental condition using a constrained search.

Similarly, communicative search helps in the exchange of better features in the bats over the generations. And best local bats or local elites, migrate to other colonies in search of more optimal environmental conditions. In this manner, along with the continuity of the regular bat algorithm, the algorithm can decide on better environmental conditions more discretely.

Echolocation of bats, in search of the best possible conditions, changes the initial population to:

$$\begin{bmatrix} 3 & 7 & 1 \\ 0 & 2 & 4 \\ 3 & 1 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 1 \\ 3 & 7 & 1 \end{bmatrix}$$

Then, the communicative search between bats results in a change of worthwhile features, and [3 7 1] changes feature with [0 2 4] to become [0 2 1] and [3 2 1] via a change of dominant features. Thus, changes look like this:

$$\begin{bmatrix}0 & 2 & 1\\3 & 2 & 1\\0 & 1 & 4\end{bmatrix} \begin{bmatrix}1 & 1 & 1\\0 & 1 & 1\\0 & 7 & 1\end{bmatrix}$$

Now the best bat from colony 2 i.e. [0 1 1] migrates from colony 2 and displaces the worst performing bat of colony 1 i.e. [0 1 4] and hence the structure changes to:

$$\begin{bmatrix}0 & 1 & 1\\0 & 2 & 1\\3 & 2 & 1\end{bmatrix} \begin{bmatrix}1 & 1 & 1\\0 & 1 & 1\\0 & 7 & 1\end{bmatrix}$$

The code is run till the termination conditions are met. In this working example, environmental conditions are optimized using integer decision variables for ease of understanding. The bat colonies, at termination, will look something like this:

$$\begin{bmatrix}0 & 0 & 1\\2 & 1 & 1\\0 & 3 & 1\end{bmatrix} \begin{bmatrix}0 & 1 & 1\\2 & 1 & 2\\3 & 3 & 1\end{bmatrix}$$

Thus, the bats echolocate, search due to communication, and migrate in search of the best environmental conditions and as shown in the matrices, the best bat or the local elite of the most optimal colony is considered the global elite. Here [0,0,1] has been declared the global elite.

The algorithm for PBCOA can be shown as follows:

Algorithm 4.1: Parallel Bat Colony Optimization Algorithm

```
Begin
  Generate a random population of bats as a 2nd-degree tensor of colonies of bats
  including their environmental condition vectors.
            $C = \mathbb{R}^{D*N}$                                    (4.1)
  where C is a bat Colony, and N is the number of bats from D environmental
  conditions.
  For Bats=1:N
    For Env. Cond.s=1:D
      In each colony, bats search for better environmental conditions using a guided
      communicative search:
          Bat1 = $\alpha.x + (1-\alpha).y$                            (4.2)
```

```
        Bat2 = α.x + (1-α).y                                              (4.3)
        where α is a random number, and x and y are guided communications from the
        best bats of a colony.
        Migration operation is defined as:
                $Migration = T|B_{ij} \leftrightarrow B_{i'j'}$                   (4.4)
        Where $B_{ij}$ are the environmental conditions of ith Bats in the jth Colony that
        travels to better colony j' in search of better environmental conditions by
        removing the worse performing bat i' in that colony.
        Bats travel via echolocation with frequency f, velocity v, and position x:
                $f_t = f_{min} + (f_{max} - f_{min})\beta,$                        (4.5)
                $v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*)f_i,$                        (4.6)
                $x_i^t = x_i^{t-1} + v_i^t$                                        (4.7)
        where β ∈ [0, 1] is a random vector drawn from a uniform distribution.
        The best bats travel to other habitats in search of better conditions. The
        condition for the selection of local elite bats is:
                $q_{ij} = fitness(B_{ij})$                                         (4.8)
                $q_{max} = \max(q_{ij})$                                           (4.9)
        The fittest bats survive in the process and the worse ones die off.
    End for
  End for
  Output is produced after termination:
                $S_{best} = B_{ij}|fitness(B_{ij}) = q_{max}$                      (4.10)
End
```

4.3. CEC-14 Benchmark Functions

The best standards developed in the field of computation are updated regularly by the Institute of Electrical and Electronic Engineers (IEEE). The IEEE Congress on Evolutionary Computation is one of the procedures handled by IEEE [112]. The formation of standard benchmarks used for the evaluation of evolutionary computation algorithms is determined by CECs held by IEEE regularly [86], [177]. Table 4.2

displays the 30 benchmark test functions that are still used today by the IEEE CECs to determine whether an evolutionary metaheuristic is effective. Every metaheuristic development should have the goal of bringing these benchmark test functions' costs down.

$$CF(\boldsymbol{X}) = Min\big(F_T[x_1, x_{2,}x_3, \dots, x_D]\big) \mid CF \geq 0, 0 < T \leq 30$$

where CF is the Cost Function that is always greater than or equal to 0. The aim of a good algorithm should be to reduce the value of CF as close to 0 as possible. Cost is the outcome of minimizing the value of the test function that can take in D variables or dimensions, in other words. According to IEEE CEC-14, the test functions can be tested with either 10, 30, 50, or 100 dimensions. All these values range from [-100,100].

The code for the Parallel Bat Colony Optimization Algorithm was developed in MATLAB and implemented. The CEC-14 benchmark test functions for MATLAB were downloaded from the internet. The code for the Parallel Bat Colony Optimization Algorithm was modified according to the requirements of CEC-14 benchmark functions. 10 dimensions were provided as D. The lower bound and upper bound of these 10 variables were respectively [-100,100]. Fitness functions were provided from the MATLAB library. Costs were evaluated and noted.

The hyperparameters in the Parallel Bat Colony Optimization Algorithm were fine-tuned for the best performance. The code was run continuously for more than 2 days with 50,000 generations provided before each benchmark function recorded its results. The machine used for the implementation of the code was an Asus VivoBook S15 PC with 8GB RAM, 512 GB SSD, NVIDIA 2GB Graphics card, and an Intel Core i7 processor. The observations were noted down and compared with the state-of-the-art technology. The results are promising.

4.4. Results and Discussions

The Parallel Bat Colony Optimization Algorithm was compared with the state-of-the-art metaheuristics on the CEC-14 benchmark test functions and the results that were seen, seem to be promising. Out of 30 benchmark functions, it was noteworthy that, in some cases, the Fitness Cost (FC) reached exact zero meaning thereby global optima.

In other cases, while an FC of 0 was not seen, instances of FC nearly equal to zero were seen. However, it is seen that in other state-of-the-art algorithms, an FC that nearly equals 0 is seen. Here, the concept of precision decides the goodness of one algorithm over the other.

As it is discussed, fitness costs nearly equal to zero were noted. However, the algorithm with the smallest value even after the decimal is declared the winner. There were instances where precision as low as E-03 was required to decide the winner over a benchmark test function. The Parallel Bat Colony Optimization Algorithm or PBCOA was tested alongside 17 other algorithms as shown in Table 4.3 referred from [146]. The results clearly depict the goodness of PBCOA over every other algorithm given in the literature.

Table 4.2: Fitness Costs of PBCOA over each of the 30 Benchmark Functions in the CEC-14 Benchmark Functions List [178]

Type	S. No.	Functions	Cost
Unimodal Functions	**F1**	Rotated High Conditioned Elliptic Function	4.73E-01
	F2	Rotated Bent Cigar Function	7.32E+02
	F3	Rotated Discus Function	0.00E+00
Simple Multimodal Functions	**F4**	Shifted and Rotated Rosenbrock's Function	5.20E-05
	F5	Shifted and Rotated Ackley's Function	2.00E+01
	F6	Shifted and Rotated Weierstrass Function	0.00E+00
	F7	Shifted and Rotated Griewank's Function	0.00E+00
	F8	Shifted Rastrigin's Function	0.00E+00
	F9	Shifted and Rotated Rastrigin's Function	9.95E-01
	F10	Shifted Schwefel's Function	0.00E+00

	F11	Shifted and Rotated Schwefel's Function	1.34E+01
	F12	Shifted and Rotated Katsuura Function	8.64E-12
	F13	Shifted and Rotated HappyCat Function	4.22E-02
	F14	Shifted and Rotated HGBat Function	8.55E-03
	F15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	2.85E-01
	F16	Shifted and Rotated Expanded Scaffer's F6 Function	9.72E-02
Hybrid Function 1	**F17**	Hybrid Function 1 (N=3)	2.08E-01
	F18	Hybrid Function 2 (N=3)	4.42E+01
	F19	Hybrid Function 3 (N=4)	1.97E-02
	F20	Hybrid Function 4 (N=4)	1.19E-01
	F21	Hybrid Function 5 (N=5)	2.48E-01
	F22	Hybrid Function 6 (N=5)	2.61E-02
Composition Functions	**F23**	Composition Function 1 (N=5)	3.29E+02
	F24	Composition Function 2 (N=3)	1.05E+02
	F25	Composition Function 3 (N=3)	1.12E+02
	F26	Composition Function 4 (N=5)	1.00E+02
	F27	Composition Function 5 (N=5)	2.94E-01
	F28	Composition Function 6 (N=5)	3.54E+02
	F29	Composition Function 7 (N=3)	2.54E+02
	F30	Composition Function 8 (N=3)	4.98E+02

Table 4.3: Comparative performance of 18 algorithms from the literature shown over the 30 benchmark test functions. [179], [180]

ALGORITHM	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
NRGA	27900	915	1520	15.4	19.6	2.45	0.203	5.59	8.69	119
FWA-DM	5010	0.000134	0	1.41	20	0.706	0.0948	0.254	6.01	1.59
UMOEAS	0	0	0	0	16.8	0	0	0	2.73	0.374
SOO+BOBYQA	4570	0.036	5840	0	20	0.002	0.049	18.9	8.96	130
SOO	8810000	6.64	6640	0.678	20	0.002	0.049	18.9	8.96	130
RSDE	0	0	0	2.81	19.2	0.0529	0.0355	0.661	8.52	68.4
POBL_ADE	16200	2270	0.000574	25.5	19.1	1.04	0.163	7.81	7.63	153
FERDE	2.37	6.29E-05	0.00135	0	19.1	0.889	0.0188	0	5.64	0.0367
FCDE	0	0	0	18.4	20.3	3.57	0.196	16.1	21	292
DE_b6e6rlwithrestart	0	0	0	1.13	18.5	0	0.0169	0	4.9	0.00123
CMLSP	1.77E-07	0	0.000106	0	16.9	0.062	0	2.07	1.66	196
GaAPADE	0	0	0	30.7	19.7	0.148	0.00316	0	3.38	0.152
OptBees	784	0.00988	0.921	2.69	20	3.02	0.156	0	20.8	219
LSHADE	0	0	0	29.4	14.2	0.0175	0.00304	0	2.35	0.00857
RMA-LSCh-CMA	0	0	1.03E-07	0.085	13.7	0.000148	0	0	3.32	7.68
MVMO	0.000495	0	0	9.55	16.6	0.00345	0.0186	0	3.49	2.14
P3PGA	17.6	14.8	0	0.0572	0	0.724	0.0986	0	1.16	0
PBCOA	**0.473**	**732**	**0**	**0.000052**	**20**	**0**	**0**	**0**	**0.995**	**0**

ALGORITHM	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
NRGA	576	0.124	0.158	0.254	1.02	2.75	16100	7420	2.09	1720
FWA-DM	372	0.0425	0.121	0.214	0.775	1.76	255	25.2	1.3	13.4
UMOEAS	144	0	0.00944	0.11	0.667	1.53	8.48	0.784	0.2	0.371
SOO+BOBYQA	349	0	0.03	0.13	0.42	2.52	423	3950	0.55	6930
SOO	349	0	0.03	0.13	0.44	2.52	3120000	12900	0.55	9360
RSDE	291	0.221	0.128	0.136	0.983	2.23	47.7	2	1.03	0.722
POBL_ADE	208	0.269	0.131	0.26	0.712	1.41	257	33.2	2.09	12.6
FERDE	75.5	0.123	0.116	0.0936	0.673	1.53	8.23	2.73	0.509	1.7
FCDE	75.5	0.123	0.116	0.0936	0.673	1.53	8.23	2.73	0.509	1.7
DE_b6e6rlwithrestart	197	0.293	0.128	0.111	0.832	1.87	1.4	0.621	0.142	0.0559
CMLSP	153	0.0303	0.0273	0.189	0.897	1.56	313	30.9	1.25	19.9
GaAPADE	183	0.14	0.0601	0.0942	0.606	1.98	9.91	0.223	0.257	0.432
OptBees	393	0.13	0.416	0.369	2.44	2.64	684	33.5	9.330-01	8.96
LSHADE	32.1	0.0682	0.0516	0.0814	0.366	1.24	0.977	0.244	0.0773	0.185
RMA-LSCh-CMA	20.1	0.0165	0.0329	0.127	0.472	1.05	78.3	5.22	0.0766	8.06
MVMO	96.3	0.0422	0.0355	0.0891	0.435	1.45	9.36	0.783	0.158	0.313
P3PGA	3.56	1.37E-06	0.0388	0.0254	0.329	0.166	48.2	6.33	0.0489	0.0768
PBCOA	**13.4**	**8.64E-12**	**0.0422**	**0.00855**	**0.285**	**0.0972**	**0.208**	**44.2**	**0.0197**	**0.119**

ALGORITHM	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30
NRGA	4820	37.6	329	131	184	100	281	477	413	1730
FWA-DM	94.6	34.1	330	127	179	100	321	347	212	394
UMOEAS	0.54	0.245	330	108	126	100	25.5	313	196	234
SOO+BOBYQA	1940	127	200	116	139	100	200	200	200	200
SOO	24700	127	200	116	145	100	200	200	200	200
RSDE	1.21	11.7	330	119	130	100	91.3	387	213	505
POBL_ADE	103	30	329	124	186	100	256	423	355000	638
FERDE	8.54	3.24	330	115	136	100	366	366	318	535
FCDE	148	27.5	330	137	184	100	47.5	457	34100	867
DE_b6e6rlwithrestart	0.787	0.154	330	112	129	100	61.6	363	218	467
CMLSP	36.4	89.5	202	110	128	100	41.1	280	200	216
GaAPADE	0.509	3.25	330	109	164	100	89.7	383	222	467
OptBees	57.1	17	272	137	146	100	7.42	307	220	389
LSHADE	0.408	0.0441	330	108	133	100	58.1	381	222	465
RMA-LSCh-CMA	49.3	8.48	330	108	175	100	185	389	227	585
MVMO	1.94	0.263	330	109	116	100	17.2	361	181	492
P3PGA	0.213	0.119	329	107	113	100	1.27	356	203	492
PBCOA	**0.248**	**0.0261**	**329**	**105**	**112**	**100**	**0.294**	**354**	**254**	**498**

Table 4.4: Number of benchmark functions over which each of the candidate algorithms showed their best performance.

Rank	Algorithm	Sole Winner	Joint Winner	Best Performance
1	**PBCOA**	**10**	**6**	**16**
2	**UMOEAS**	1	9	10
3	**b6e6rlwithrestart**	1	6	7
3	**P3PGA**	2	5	7
4	**GaAPADE**	1	5	6
5	**OO+BOBYQA**	0	5	5
5	**LSHADE**	0	5	5
5	**MA-LSCh-CMA**	0	5	5
5	**MVMO**	1	4	5
6	**SOO**	0	4	4
6	**RSDE**	0	4	4
6	**FCDE**	0	4	4
6	**CMLSP**	0	4	4
7	**FERDE**	0	3	3
8	**FWA-DM**	0	2	2
8	**OptBees**	0	2	2
9	**POBL_ADE**	0	1	1
10	**NRGA**	0	0	0

As shown in Table 4.3, we have compared the performances of PBCOA with 17 other algorithms over the 30 benchmark functions. It is clear from Table 4.5 that PBCOA has outshone most other candidates in state-of-the-art technology. PBCOA emerged as a clear sole winner on 10 out of the 30 test benchmark functions while it shared a draw with a 0 Fitness Cost on the other 6 benchmark functions. Thus, it performed best on 16 out of the 30 provided test benchmark functions.

None of the competing algorithms could stand up to the performance of PBCOA and while it outperformed most other candidates, the fitness costs are jotted down in Table 4.2 for future reference.

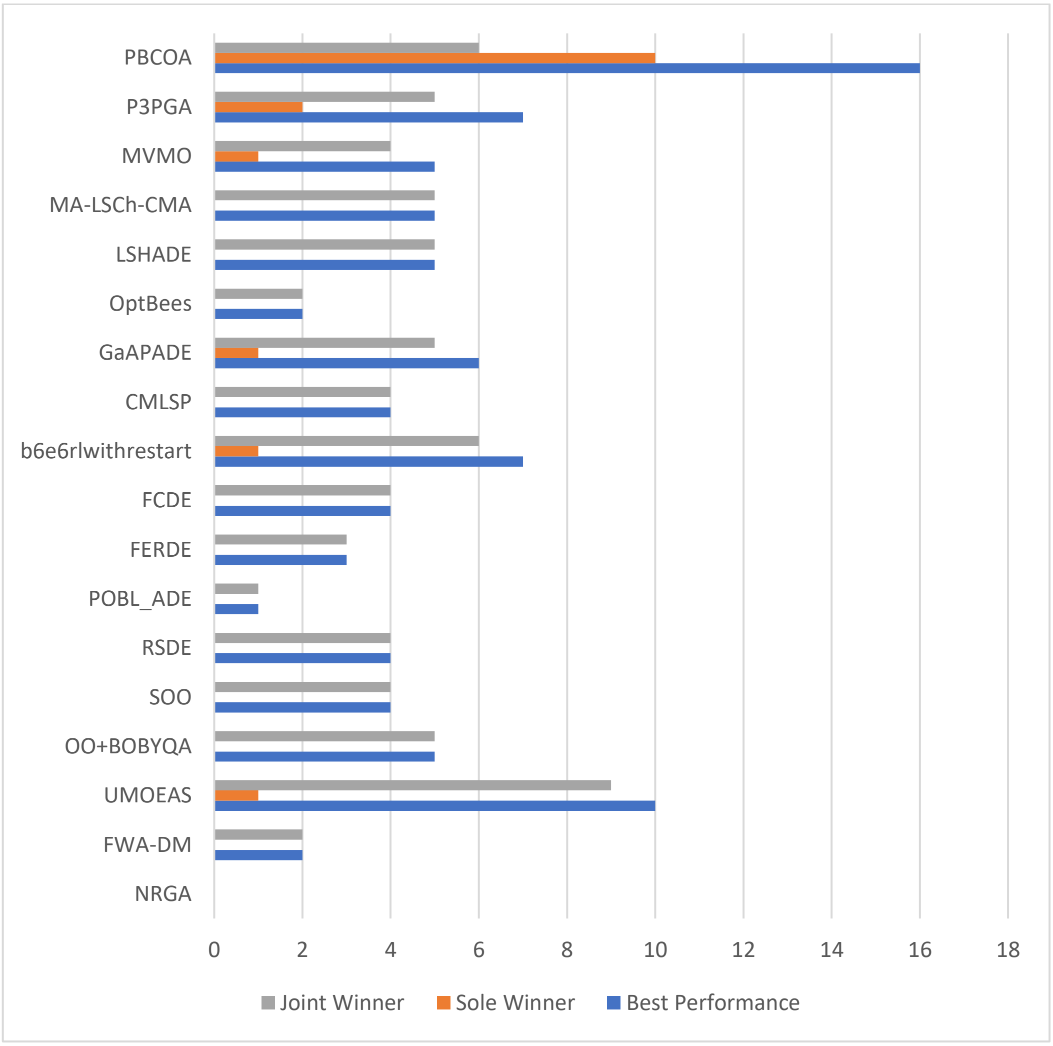


Figure 4.5: The description of best performances by all the candidate algorithms.

It is clear from Table 4.4 that it was only the Parallel Bat Colony Optimization Algorithm that performed best on the maximum number of benchmark test functions under consideration. The PBCOA performed best on 16 benchmark test functions with 10 where it was the sole winner. The second-best performance was shown by 'b6e6rlwithrestart' which showed the best performance on 7 test functions. However, it is noteworthy that, on only 1 benchmark test function it was declared the sole winner. Thereby, meaning that PBCOA, by far, performed best out of all the candidate algorithms.

4.5. Conclusion

It has been seen that the introduction of the concept of colonies in the regular Bat Algorithm greatly improves the performance of the metaheuristic. Additional phenomena like communicative search and migration, as would be seen in real bat populations, tend to improve the metaheuristic. The performance of PBCOA is tested on IEEE CEC-14 Benchmark test functions and compared with the state-of-the-art algorithms. Out of all the candidate algorithms, found in the literature, Parallel Bat Colony Optimization Algorithm fares best in almost all respects. The results of the Fitness Costs on each Benchmark function have been provided for future reference. The algorithm is compared on the grounds of results from 30 test functions available from the literature and the performance is unparalleled by any other candidate algorithm.

Chapter 5: Evolution of CNN Over Geographical Landmarks Dataset Using Parallel Bat Colony Optimization Algorithm

This chapter proposes an evolutionary framework of CNNs with a Parallel Bat Colony Optimization Algorithm (PBCOA) over a geographical landmarks' dataset. We discuss how the application of the PBCOA metaheuristic to CNNs to evolve its hyperparameters works towards the evolution of better hyperparameters out of a search space of more than a million hyperparameters. We evolved initial weights, kernel frame size, number of kernels, and other such parameters. It was found that hyperparametric combinations highly different from the usually used ones seemed to perform better. Also, the performance of CNNs was evaluated using Accuracy and F1 score, and it was found that the evolved architectures performed much better in comparison to the unevolved architectures. The evolved CNN architectures seemed to perform better on the training sets as well as the test sets.

5.1. Introduction

Massive amounts of data are always being generated for Computer Vision (CV). Images are one of the data categories that Artificial Intelligence (AI) systems find the most difficult to process. Although progress is slow, convolutional neural networks (CNNs) are used to handle such kinds of Big Data. In this study, a default CNN is evolved using the Parallel Bat Colony Optimization Algorithm (PBCOA), an evolutionary computation technique. PBCOA is an improvement over BA that has been further refined for greater optimization. In comparison to other current soft-computing-based techniques, we found from the literature that PBCOA produces good performance on common benchmark functions. With a considerable improvement in F1 scores and accuracy, PBCOA produced better performance over an unevolved CNN with regularly used hyperparameters and also over the CNN evolved with regular BA.

5.2. Proposed Methodology and Data Pre-processing

The experiment was conducted on an Asus VivoBook S15 laptop equipped with 8 GB RAM, 4 GB NVIDIA graphics card, and an Intel CORE i7 processor. The experimental programs were written in Python using Jupyter Notebooks. The Convolutional Neural Networks were designed and applied on Google Landmarks Dataset V2 (accessed on 29th January 2022). The network architecture was evolved using two evolutionary metaheuristics namely Bat Algorithm and a self-created and improvised Parallel Bat Colony Optimization Algorithm. The performances were evaluated based on accuracy and F1 score metrics.

The Google Landmarks Dataset V2 originally contained more than 4 million images spread throughout 200,000 classes. These images were of variable resolutions and their color schemas were also different. Some of them were in RGB color schema while few were in CMYK color schema. A subset of a little more than 1000 images was downloaded from the Kaggle website. These images were spread through 50 geographical landmark classes. These images were checked for their color schema first. The ones in an 8-bit RGB color schema were kept and others were discarded.

Also, the resolution of each image was changed to 64*64 from all the higher resolutions. Then, to maintain generalization in the data, the data was augmented in 5 ways namely scaling, cropping, flipping, rotation, and translation. The initial number of images was 1000 which changed to 6000 after data augmentation. After this, the combined dataset was split into training and test sets in the ratio of 70:30. Hence the training and test set tensors had the shape of:

$$X_{train}.shape = (4200,64,64,3) \quad \text{…(5.1)}$$

$$X_{test}.shape = (1800,64,64,3) \quad \text{…(5.2)}$$

$$X.shape = (6000,64,64,3) \quad \text{…(5.3)}$$

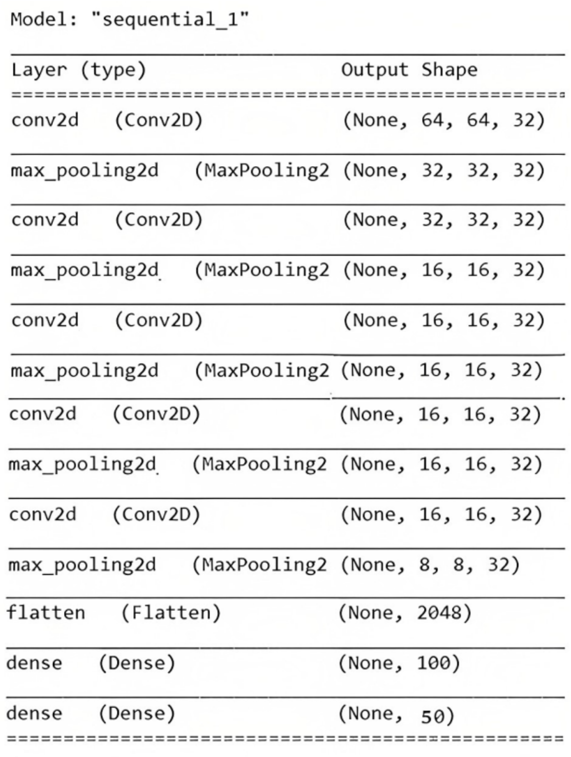
The shape of output tensor Y can be given by:

$$Y.shape = (6000,50) \quad \text{…(5.4)}$$

A convolutional neural network was designed with 5 convolutional layers each followed by a Max-Pooling layer. After these layers, two fully connected layers connect these hidden layers to the output node. The early convolutional layers perform the task of identification of broader features in an image and as we go deeper, the convolutional layers help in the recognition of very specified and minute characteristics in an image that can help in its characterization into one of the Landmark classes. The Max-Pooling layers help in the reduction of linearity in the network so that the activation functions get trained properly without facing issues of vanishing or exploding gradients.

The designed CNN architecture, as described in Figure 5.1, can be summarised as:

- Inputting images in the specified size of 64*64 with RGB encoding



```
Model: "sequential_1"
_________________________________________________
Layer (type)                 Output Shape
=================================================
conv2d   (Conv2D)            (None, 64, 64, 32)
_________________________________________________
max_pooling2d   (MaxPooling2 (None, 32, 32, 32)
_________________________________________________
conv2d   (Conv2D)            (None, 32, 32, 32)
_________________________________________________
max_pooling2d   (MaxPooling2 (None, 16, 16, 32)
_________________________________________________
conv2d   (Conv2D)            (None, 16, 16, 32)
_________________________________________________
max_pooling2d   (MaxPooling2 (None, 16, 16, 32)
_________________________________________________
conv2d   (Conv2D)            (None, 16, 16, 32)
_________________________________________________
max_pooling2d   (MaxPooling2 (None, 16, 16, 32)
_________________________________________________
conv2d   (Conv2D)            (None, 16, 16, 32)
_________________________________________________
max_pooling2d   (MaxPooling2 (None, 8, 8, 32)
_________________________________________________
flatten   (Flatten)          (None, 2048)
_________________________________________________
dense   (Dense)              (None, 100)
_________________________________________________
dense   (Dense)              (None, 50)
=================================================
```

Figure 5.1: The layered architecture of 5-layered CNN

- Convolutional Layers
    - 5 layers in a fixed value
    - Kernels with variable Frame Size
    - Strides of variable value

    - ‘Same’ padding to reduce anomalies
    - Variable Learning Rate
    - Variable Batch Size
- Max-Pooling Layers
    - 5 layers each followed after a convolutional layer
    - Stride fixed to 2
    - Window size fixed to 2
- Fully Connected Layers
    - 2 layers containing variable neurons
    - Last layer connecting to the output node with 50 classes
    - Use of Softmax Activation Function

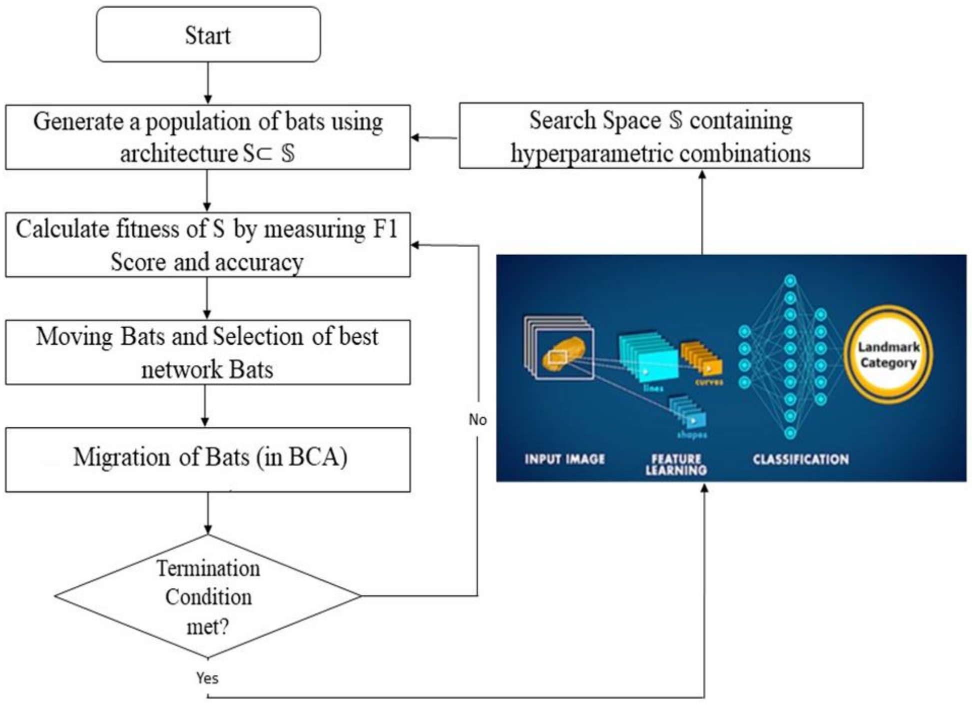

Start
Generate a population of bats using architecture S⊂ 𝕊
Search Space 𝕊 containing hyperparametric combinations
Calculate fitness of S by measuring F1 Score and accuracy
Moving Bats and Selection of best network Bats
Migration of Bats (in BCA)
Termination Condition met?
No
Yes
Landmark Category
lines
curves
shapes
INPUT IMAGE
FEATURE LEARNING
CLASSIFICATION

Figure 5.2: The process of evolution of the network through BA and PBCOA is described diagrammatically

Figure 5.2 shows the working of proposed PBCOA based approach to evolve the architecture of CNN for landmark recognition. Initially, the proposed approach generates the random multiple populations of CNN hyperparameters. Then, the proposed Parallel Bat Colony Optimization Algorithm produces new architectures of the Convolutional Neural Networks based on their performance on landmark recognition. The new architectures are produced by varying the hyperparameters of regularly used CNN architectures. The termination criteria of the proposed approach is same as 3-parent genetic algorithm based approach (already discussed in Chapter 3 of the thesis).

The Bat Algorithm evolved architectures based on the regular echolocation phenomenon where Bats (combination of hyperparameters) were evaluated based on two metrics namely Accuracy and F1 score of the CNN. Parallelly, the Bats were evolved based on the survival of bats in colonies in PBCOA. The fittest bats were observed after the algorithms tended to reach convergence.

The list of hyperparameters that are evolved in the network is listed in Table 5.1.

Table 5.1: A list of the evolved hyperparameters along with the range in which they were varied.

Hyperparameters	**Range**
Initial Weights	0-5
Kernel Frame Size	1-11
Stride	1-5
Number of Kernels	32-96
Learning Rate	0.001-0.999
Batch Size	16-48
Neurons	80-120

5.3.Results and Discussions

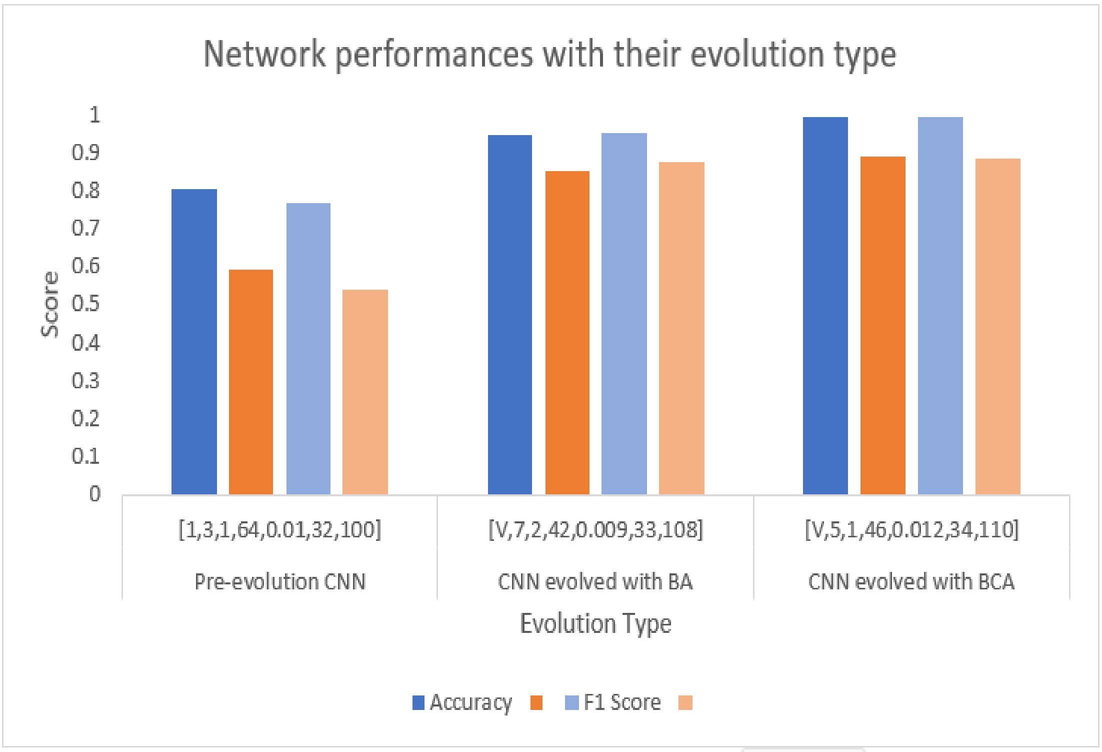


Figure 5.3: The graph depicting network performances along with their respective evolution mechanisms. As it is clear, CNN with PBCOA performs the best in all categories, and the pre-evolution CNN fares worst. Evolution with BA shows drastic improvements over unevolved CNN.

Before evolving the network, the 5-layered CNN was tested with the most commonly used hyperparameters. These hyperparameters, as shown in Table 5.1, showed an F1 score of 0.767. After this, initial combinations of hyperparameters as shown in Table 5.2 were provided to BA and PBCOA as one bat each. The evolution of networks was done till an apparent convergence was not reached. After evolution, some striking observations were made. The hyperparametric combinations, quite different from those used normally, were seen to perform better. Also, the best fits, which are described in Table 5.3 were extremely different from regular CNNs. It is described in Figure 5.3.

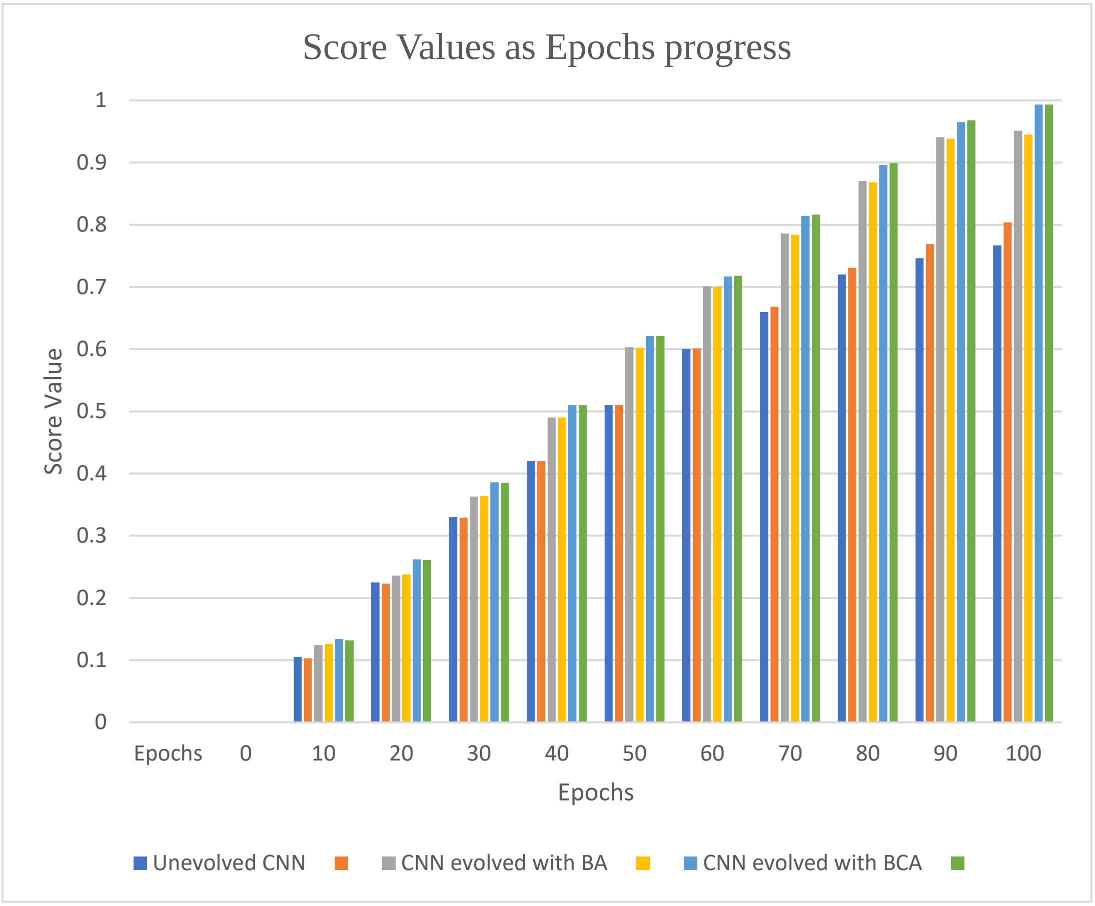


Figure 5.4: The graph depicts the changes in F1 Scores and accuracies of the Best Fit networks as epochs progress.

Table 5.2: The hyperparameters set for the initial CNN for the purpose of comparison with evolved networks. These hyperparameters are the ones that are most commonly used while designing CNNs.

Hyperparameter	**Value**
Initial Weights	1
Kernel Frame Size	3
Stride	1
Number of Kernels	64
Learning Rate	0.01
Batch Size	32
Neurons	100

Table 5.3: The network architectures before evolution, after evolution with BA, and after evolution with PBCOA. The hyperparametric combinations are described in the format [Initial Weights, Kernel Frame Size, Stride, Number of Kernels, Learning Rate, Batch Size, Neurons]. Each architecture's train and test set accuracy and F1 scores have been provided. (V in the Best Fit stands for Variable)

		Accuracy		**F1 Score**	
Network Architecture	**Best Fit**	**Train**	**Test**	**Train**	**Test**
Pre-evolution CNN	[1,3,1,64,0.01,32,100]	0.804	0.591	0.767	0.539
CNN evolved with BA	[V,7,2,42,0.009,33,108]	0.945	0.854	0.951	0.874
CNN evolved with PBCOA	[V,5,1,46,0.012,34,110]	0.993	0.892	0.993	0.885

Table 5.4: The F1 Scores and Accuracies of networks as time passes and epochs elapse. Clearly, PBCOA shows maximum accuracy within the minimum time frame.

	Unevolved CNN			**CNN evolved with BA**			**CNN evolved with PBCOA**		
Epochs	**Time (in mins)**	**F1 Score (approx.)**	**Accuracy (approx.)**	**Time (in mins)**	**F1 Score (approx.)**	**Accuracy (approx.)**	**Time (in mins)**	**F1 Score (approx.)**	**Accuracy (approx.)**
0	0	0	0	0	0	0	0	0	0
10	15	0.105	0.103	15	0.124	0.126	13	0.134	0.132
20	30	0.225	0.223	30	0.236	0.238	25	0.262	0.261
30	60	0.33	0.329	60	0.363	0.364	50	0.386	0.385
40	90	0.42	0.42	90	0.49	0.49	75	0.51	0.51
50	120	0.51	0.51	120	0.603	0.602	100	0.621	0.621
60	150	0.6	0.601	150	0.701	0.7	125	0.717	0.718
70	180	0.66	0.668	180	0.786	0.784	150	0.814	0.816
80	210	0.72	0.731	210	0.87	0.868	175	0.896	0.899
90	240	0.746	0.769	240	0.941	0.938	200	0.965	0.968
100	300	0.767	0.804	300	0.951	0.945	250	0.993	0.993

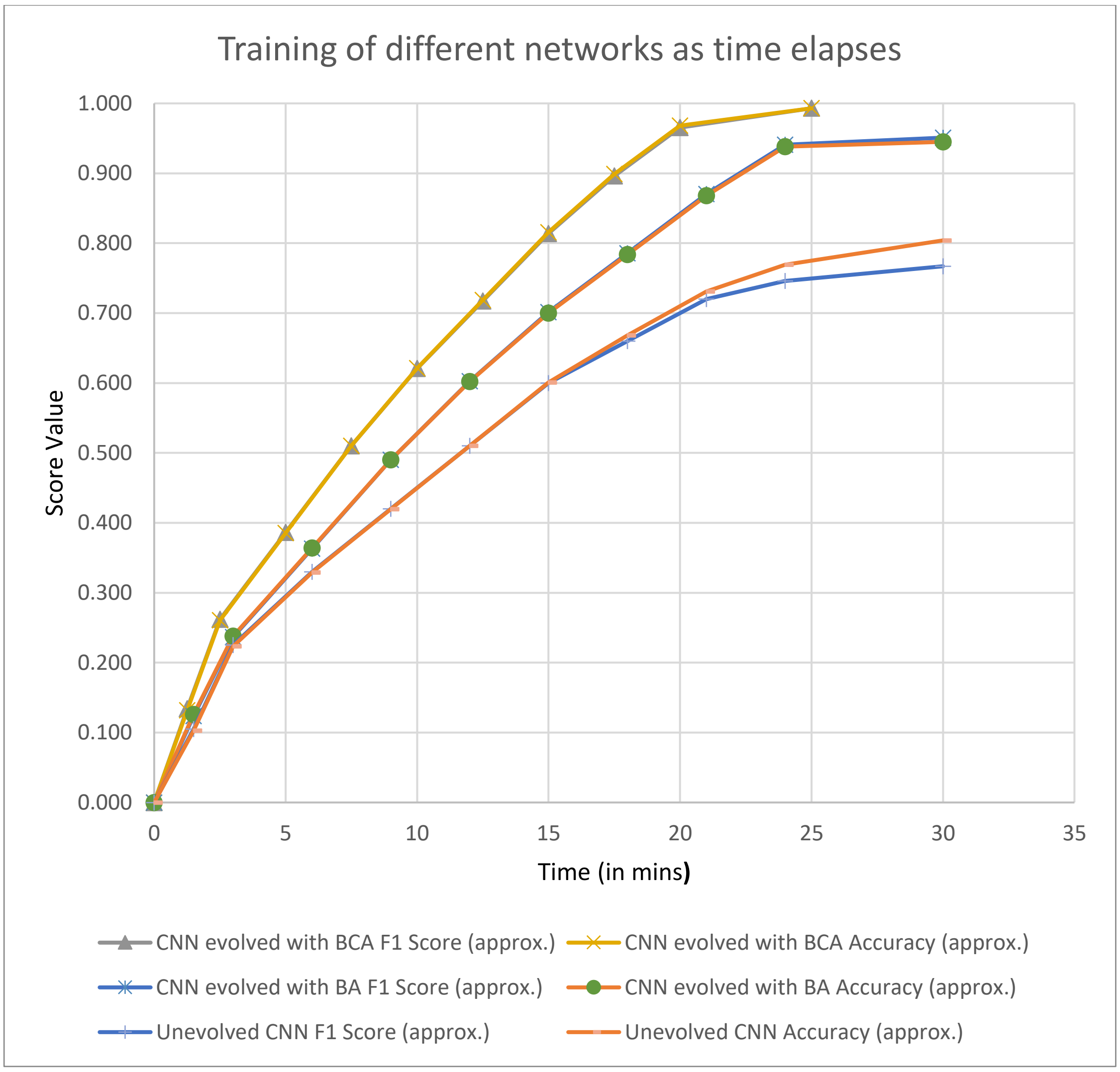


Figure 5.5: The graph depicts that the network evolved by PBCOA shows higher accuracies and F1 Scores in lesser time frames as compared to BA and unevolved CNN.

As shown in Figures 5.4 and 5.5, the networks evolved by PBCOA and BA trained faster than unevolved networks. The network evolved by PBCOA evolved faster than the one evolved by BA. Table 5.4 gives a complete review of this. From Figure 5.6, it was noteworthy that PBCOA not only gave a better overall performance but as time elapsed, it seemed to reach convergence earlier than BA. Thus, PBCOA evolved the networks more quickly and efficiently than BA. This data has been provided in Table 5.5.

Table 5.5: The table shows how accuracies and F1 scores vary as the networks evolve over time.

	The network evolved by BA		**The network evolved by PBCOA**	
Time (in mins)	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**
0	0.000	0.000	0.000	0.000
30	0.767	0.804	0.767	0.804
60	0.806	0.842	0.811	0.821
90	0.824	0.867	0.855	0.838
120	0.839	0.892	0.871	0.855
150	0.859	0.903	0.898	0.872
180	0.882	0.914	0.940	0.934
210	0.910	0.924	0.961	0.960
240	0.935	0.935	0.987	0.984
270	0.945	0.940	0.992	0.990
300	0.951	0.945	0.993	0.993

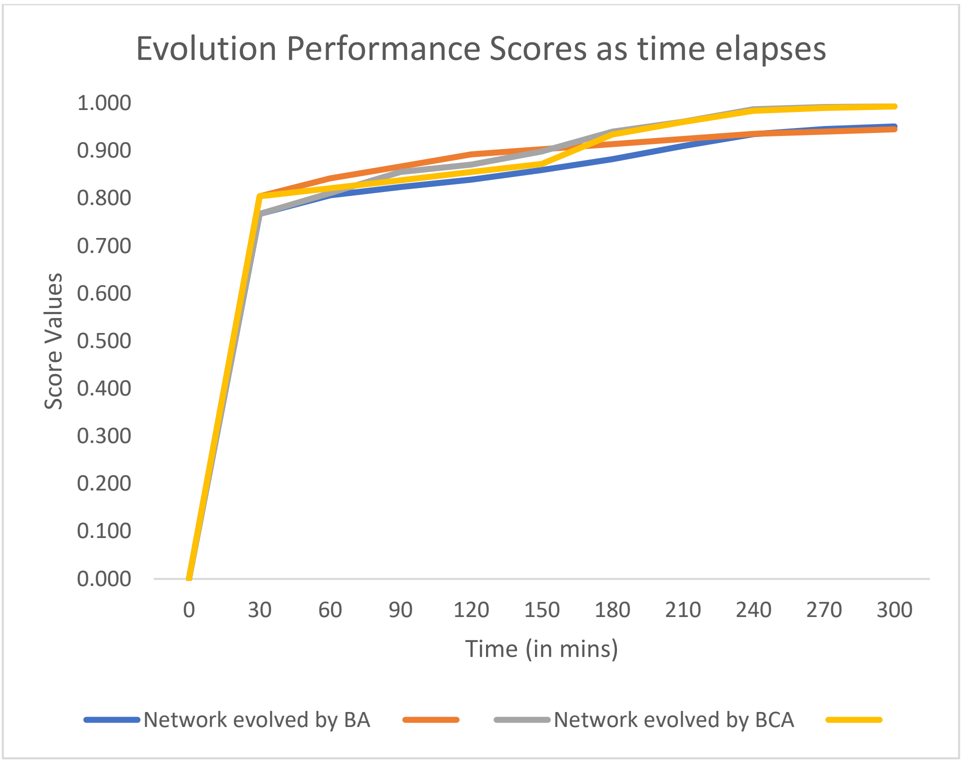


Figure 5.6: The graph depicting accuracies and F1 Scores of networks as they evolve toward better performance.

The network gets trained in the backpropagation phase when the learned weights are fed back to the network. This can be described by:

$$W_x^{new} = W_x^{old} - a\frac{d(Error)}{dW_x} \qquad \ldots(5.5)$$

where, W_x^{new} are the newly trained weights, W_x^{old} are the old weights, a is the learning rate and $a\frac{d(Error)}{dW_x}$ is the change an error with respect to the weights. For an optimal change of weights, Keras' built-in libraries and the AdaDelta optimizer are used.

As described in Figure 5.2, the hyperparameters to be evolved were fed to Bat Algorithm (BA) and Parallel Bat Colony Optimization Algorithm (PBCOA). These metaheuristics selected the best combination of hyperparameters as the best bats and continued their characteristics into newer generations. Slowly, the metaheuristics evolved these bats into bats with the fittest characteristics suited for the network.

As shown in Figure 5.5, After the evolution of networks with PBCOA and BA, we saw drastic improvements in the performances of networks along with a major change in the optimal hyperparameters for designing the architectures of CNNs for geographical landmark recognition. The ranges in which these hyperparameters were varied are described in Table 5.6. Some changes that were observed in the evolved hyperparameters were:

- The optimal initial weights evolved from a set of 3X3 kernels of all 1s changed to a kernel of 5X5 with extremely unpredictable values in each pixel. Although seemingly symmetric, the kernel with the best initial weights from PBCOA was:

0	1	2	1	0
1	2	4	2	1
2	4	15	4	2
1	3	4	2	1
0	1	2	1	0

- The optimum kernel frame size was a square kernel of length 7 in BA and a square kernel of size 5 in PBCOA. This was against the common notion that a kernel frame size of 3X3 is the best.

- The optimal stride evolved by BA was 2 but PBCOA suggested that a stride of 1 was only better. The unevolved CNN was also designed with a stride of 1.
- In the range of 32–96, the number of kernels after optimization with BA was 42 while it was 46 with PBCOA, indicating that varying the number of kernels increased the model's accuracy. To confirm the ideal number of kernels, an additional study would be needed.
- The standard learning rate is 0.01; however, it was discovered that the optimal learning rate was 0.009 and 0.012, which are nearly equal to 0.01.
- Since the network did not then demand a lot of processing power at once, the number of optimum epochs was manually set at 100. And the accuracies seemed to converge after 100 epochs as suggested by Figures 5.5 and 5.6.
- The standard batch size of 32 was the best one that was observed. The BA evolved it to 33 while PBCOA evolved it to 34 which can be safely ignored.
- The best result was 110 when initiated with 100, suggesting that the neurons did not appear to affect the performance significantly. This was with PBCOA. BA evolved it to 110.
- In total, the codes for evolution were run for more than 10 hours checking over 40 bats with different hyperparametric combinations. After 5 hours each for BA and PBCOA, convergence seemed to be reached.
- PBCOA seemed to reach convergence earlier and with higher performance than BA. As provided in Table 5.4, BA reached an F1 Score of 0.952 while PBCOA reached 0.993.
- The final training and test accuracies and F1 scores are shown in Table 5.5. It was clear that PBCOA not only fared well on training sets but also on test sets. Overfitting was lesser in the network that evolved with PBCOA.

5.4. Conclusion

From the experiments conducted, it was clear that the evolution of Deep Neural Networks with evolutionary metaheuristics is highly desirable before building cumbersome architectures which might not perform as well on specified datasets. Evolutionary metaheuristics help in searching the most desirable hyperparametric combinations for building DNN architectures that would improve the overall performance of networks for certain specified tasks. Finding a suitable set of hyperparameters for building DNNs can become a tedious task either manually or by brute force in the search space consisting of more than a million combinations. The bat Algorithm and the improvised Parallel Bat Colony Optimization Algorithm help in this search and optimize the structures to a great extent which is not expected manually.

Chapter 6: Comparative Analysis of All the Proposed Approaches

This chapter describes a comparative analysis of all the developed Soft-Computing approaches. We have discussed all the metaheuristics which were used to evolve CNN architectures and in addition to this, we have discussed the differences in datasets that were used to evolve the CNN architectures. We have discussed how the size of datasets was varied to maintain a healthy specialization and generalization ratio. We discuss the concept of data augmentation and how it was used to develop better datasets. Every ML technique ranging from classical ML techniques to evolved CNN architectures has been discussed. The evolved hyperparametric combinations were used on a landmarks dataset with 200 classes and the performance of each architecture was measured using accuracy. It was concluded that the CNN hyperparameters evolved with PBCOA seemed to perform the best on the complete dataset.

6.1. Introduction

In our study, we have developed many soft-computing-based techniques for landmark recognition in images. Starting from classical Machine Learning ensemble-based AI models, we developed regularly built Convolutional Neural Networks (CNNs) and evolved them further with evolutionary metaheuristics. We used various geographical landmark recognition datasets and experimented with them with many AI techniques. These include Support Vector Machines (SVMs), Decision Tree Classifiers, Random Forest Classifiers, K-Nearest Neighbors, regular CNNs, CNNs using transfer learning, and last but not the least, CNNs evolved with evolutionary metaheuristics like Paddy Field Algorithm, Genetic Algorithm, 3 Parent Genetic Algorithm, and our own developed Parallel Bat Colony Optimization Algorithm.

The Landmark Recognition datasets that were used included geographical landmarks datasets of Delhi’s famous monuments and a subset of Google Landmarks Dataset V2. The subset of Google Landmarks Dataset V2 that was extracted was further augmented in many ways including shearing, cropping, translation, zooming, etc. This paved the way for more generalization in our models and thus more test accuracy. A comparison between different techniques studied and developed by us has been drawn.

6.2. Classical Machine Learning Techniques vs CNN

In this section, we discuss the implementation of some machine learning algorithms on the task of landmark recognition. We use a publicly available dataset taken from the renowned machine-learning platform Kaggle. The dataset named Qutub Complex Monuments' Images Dataset consisted of 1286 image files belonging to 5 classes. Out of these 1286 image files, 1270 images had a 3-color coding. We used only the 3 color-coded images (RGB) for the sake of uniformity.

So, our dataset was appropriately cleaned for being fed to various machine learning models. We trained 4 machine-learning models with this dataset. These models were namely:

1. Convolutional Neural Network
2. Support Vector Machine
3. K Nearest Neighbours
4. Decision Tree Classifier

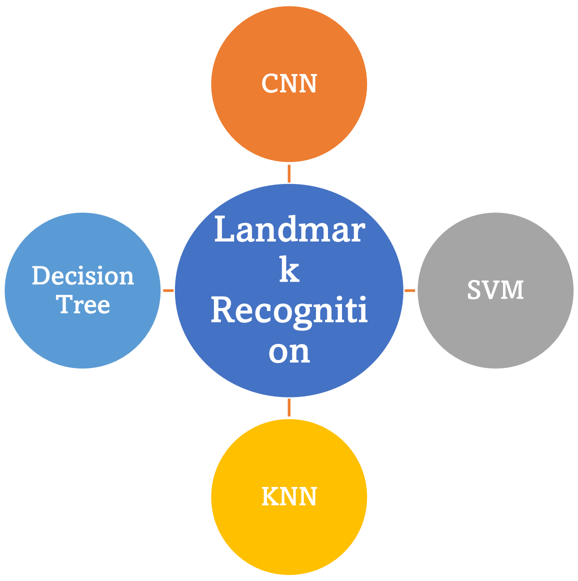


Figure 6.1: Implementation of various ML techniques on Landmark Recognition

A generalized algorithmic approach for the implementation of a Machine Learning Technique for image classification can be shown in Figure 6.2.

On implementing the above-mentioned ML techniques, we found the training and test accuracies as mentioned in Table 6.1. Where all the training accuracies were quite high, the test accuracies were considerably lower than the training accuracy, thereby, indicating overfitting of the models.

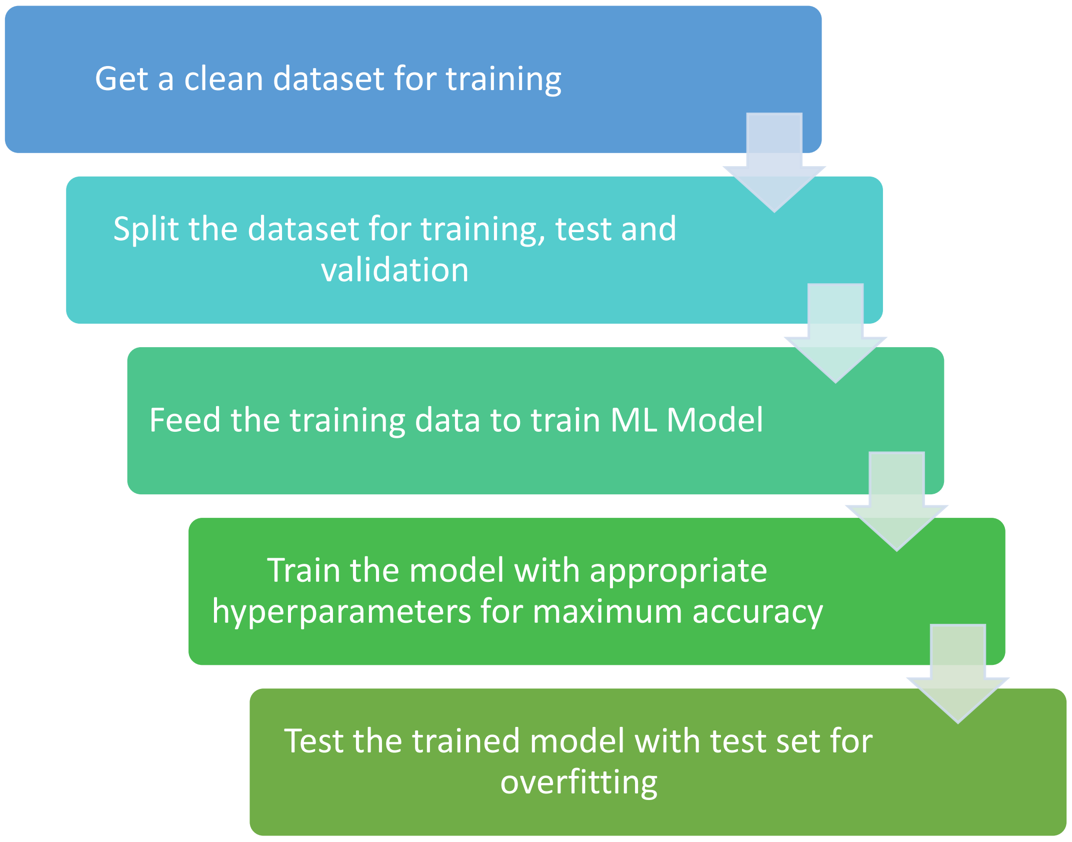

Get a clean dataset for training
Split the dataset for training, test and validation
Feed the training data to train ML Model
Train the model with appropriate hyperparameters for maximum accuracy
Test the trained model with test set for overfitting

Figure 6.2: Process of training an ML model.

All the ML models were run in their most basic form without any changes in hyperparameters. The convolutional neural network was run with a single layer. In SVM too, no kernel or hyperparameters were changed. Similarly, KNN and Decision Tree Classifier were also run in their default form. As it is imperative from Table 6.1, Convolutional Neural Networks fared better than all other Machine Learning techniques.

Table 6.1: Summarising training and test accuracies for various ML models

ML Technique	Training Accuracy (in %)	Test Accuracy (in %)
Support Vector Machine (SVM)	89	64
K-Nearest Neighbour	74	54
Decision Tree Classifier	99	49
Convolutional Neural Network	98	73

It should not be difficult to understand that CNNs belong to a whole new class of Machine Learning algorithms named the "Deep Learning" techniques. In Deep Learning, instead of a single model being trained to classify data, we have a whole network of activation functions that work like a neuron in our body. The deep learning paradigm may also be thought of as a model trained from numerous models.

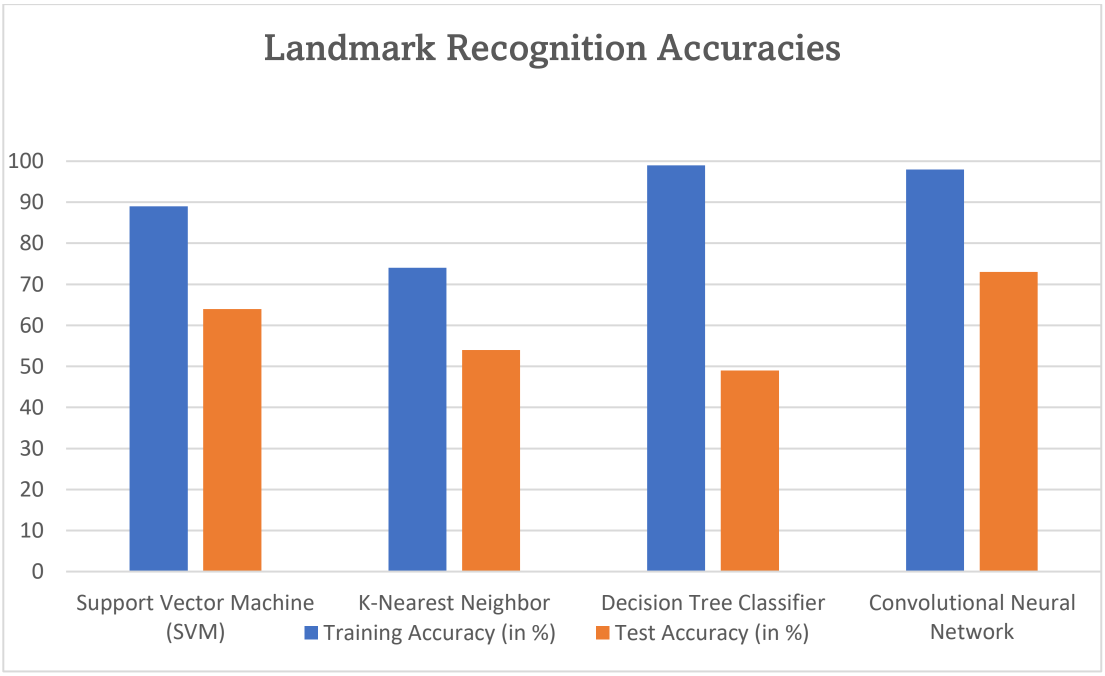


Figure 6.3: Summarising training and test accuracies for various ML models

As we can see from Table 6.1, CNN performs the best on our dataset both in terms of training and test accuracy. Though the test accuracy is a little lower than the training accuracy, thereby, indicating overfitting; the accuracies are much better than the rest of the techniques. Out of the rest of the techniques known as the classical machine learning techniques, Support Vector Machines fare the best.

As shown in Figure 6.3, out of the classical machine learning approaches, SVMs are known to fare the best. It is also thought that SVMs can achieve 100% training accuracy if appropriate parameters are changed. However, a 100% training accuracy does not imply that our model is good enough to be generalized. Also, the other two algorithms do not seem to fare that well. Though decision trees seem to give a training accuracy of 99%, a test accuracy of 49% makes it inappropriate to be used for practical uses.

6.3. Unevolved CNN vs CNNs evolved with Metaheuristics

We have seen that CNNs perform the best out of all the Machine Learning techniques in the previous section. However, the test accuracy of CNN was barely sufficient to be of any much practical usage. Hence, the accuracies still need to be improved for the CNNs. One way that the researchers have developed, is to use heavily generalized models with complex architectures that improve performance in a generalized manner for many datasets.

The highly complex and generalized model is expected to not only perform well on the dataset on which it was trained but also on the ones on that, it was not trained. If a positive transfer of knowledge is seen in these architectures, they are made public so that a variety of AI models can be trained without much hassle.

The other way around, a method called NAS or Neural Architecture Search is used to search for the most optimal architectures of Artificial Neural Networks for a specific task. This architecture is mostly evolved with evolutionary metaheuristics and such architectures tend to overperform other architectures at the task at hand.

In our study, we applied NAS with some evolutionary metaheuristics like Paddy Field Algorithm, Genetic Algorithm, 3 Parent Genetic Algorithm, and an algorithm developed by us namely the Parallel Bat Colony Optimization Algorithm which is an inspiration by the previously existent Bat Algorithm.

The datasets that we used were different subsets of the same dataset named Google Landmarks Dataset V2 which is a highly recognized dataset for the development of robust AI models for landmark recognition. These subsets were used in various forms which are depicted as follows:

1. **Subset 1**: This dataset had an initial number of 600 images spread over 24 classes. Each class had an equal number of images i.e. 25. This dataset was augmented by a massive 15 times to create a dataset of 9000 images. This was done to improve generalization in the dataset and the model generalized well over the larger dataset.
2. **Subset 2**: This dataset had an initial number of only 576 images spread over 48 classes uniformly. This dataset was divided in the ratio of 70:30 for training and test images meaning that there were 403 images available for training and 173 images for the test. The dataset was not augmented, and it was seen that the model showed relatively lesser generalization of data.
3. **Subset 3**: This dataset had an initial 1000 images spread uniformly over 50 classes. The data was augmented 6 times to 6000 images and divided into training and test images with a ratio of 70:30. The model not only performed exceedingly well on the trained dataset but also on the test data.

As shown in Figures 6.4, 6.5, and 6.6, the dataset sizes were varied not only based on the amount of generalization required but also to test whether our model evolved successfully or not. We used the Paddy Field Algorithm for the first dataset, GA and 3PGA for the second, and our own proposed Parallel Bat Colony Optimization algorithm for the third dataset.

For the first experiment, we relied on augmentation entirely for generalization and we saw good results. For the second experiment, we did not use augmentation. The results were slightly worse but the evolution with better metaheuristics made up for the worsening in the AI model.

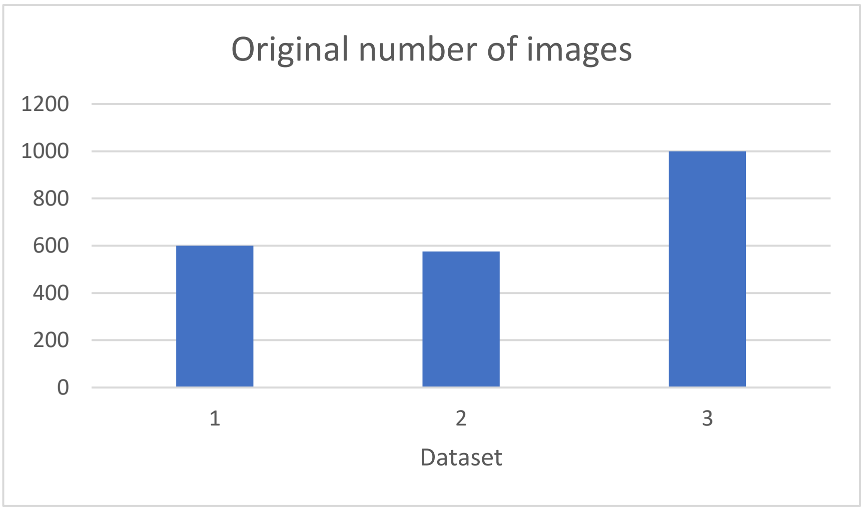


Figure 6.4: Number of Classes in each subset of dataset

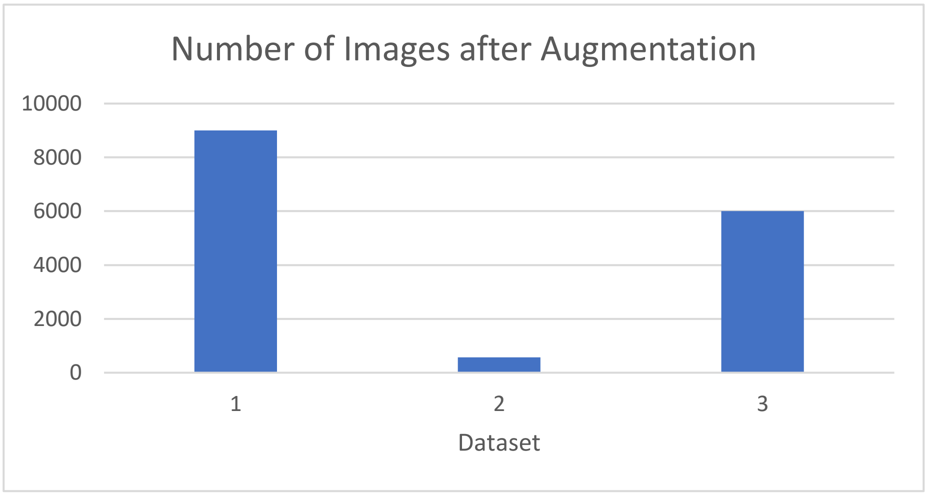


Figure 6.5: Number of images in the original subset

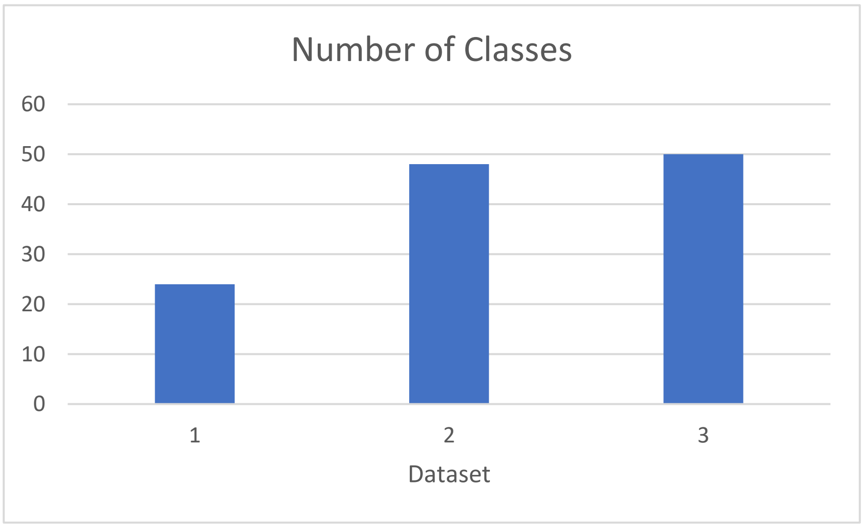


Figure 6.6: Total number of images in each dataset after augmentation of datasets

For the third experiment, we had to test on a holistic approach since the metaheuristic was proposed by us. First, we benchmarked the algorithm on available CEC 14 benchmark functions and the results made it outstanding when compared to current counterparts. After that, the CNN was evolved with an extended experiment to observe training and test accuracies and F1 scores. The results clearly showed that PBCOA outperformed regular BA while evolving the CNN hyperparameters.

6.4. Evaluation of CNNs over Evolved Hyperparameters

The hyperparameters evolved by different metaheuristics over different Landmark Recognition datasets were evaluated over a larger dataset that comprised 200 classes. Each class had 50 images and the dataset was not further augmented to produce any generalization. The larger dataset was also a subset of the Google Landmarks Dataset V2 just like any other smaller subset.

Smaller subsets with more generalization were used to evolve better CNN hyperparameters and now, the evolved hyperparameters were tested on the larger dataset with lesser generalization. The larger dataset was also divided into training and test sets and accuracies were checked. The tensors involved in this experiment were:

$$X.shape = (10000,64,64,3) \qquad \ldots(5.1)$$

$$X_{train}.shape = (7000,64,64,3) \qquad \ldots(5.2)$$

$$X_{test}.shape = (3000,64,64,3) \qquad \ldots(5.3)$$

$$Y.shape = (10000,200) \qquad \ldots(5.4)$$

Where X is the input tensor, X_{train} is the input tensor used for training purposes, X_{test} is the input tensor used for test purposes. Y is the output tensor. The shape function is used to determine the dimensional information of each tensor.

Every input tensor has the first dimension of 3 consisting of 3 color channels corresponding to RGB, 64 x 64 image height and width, and each 4th dimension containing the number of images in each tensor.

The performances of CNN architectures evolved with each metaheuristic have been provided in Table 6.2.

Table 6.2: Training and test accuracies achieved by CNN architectures evolved with each metaheuristic. Each evolved architecture has been named after the evolutionary metaheuristic used to evolve it.

	Accuracy	
Evolved CNN	**Training**	**Test**
PFANET	98.56	90.15
GANET	99.48	92.37
3PGANET	99.14	94.47
PBCOANET	99.20	95.11

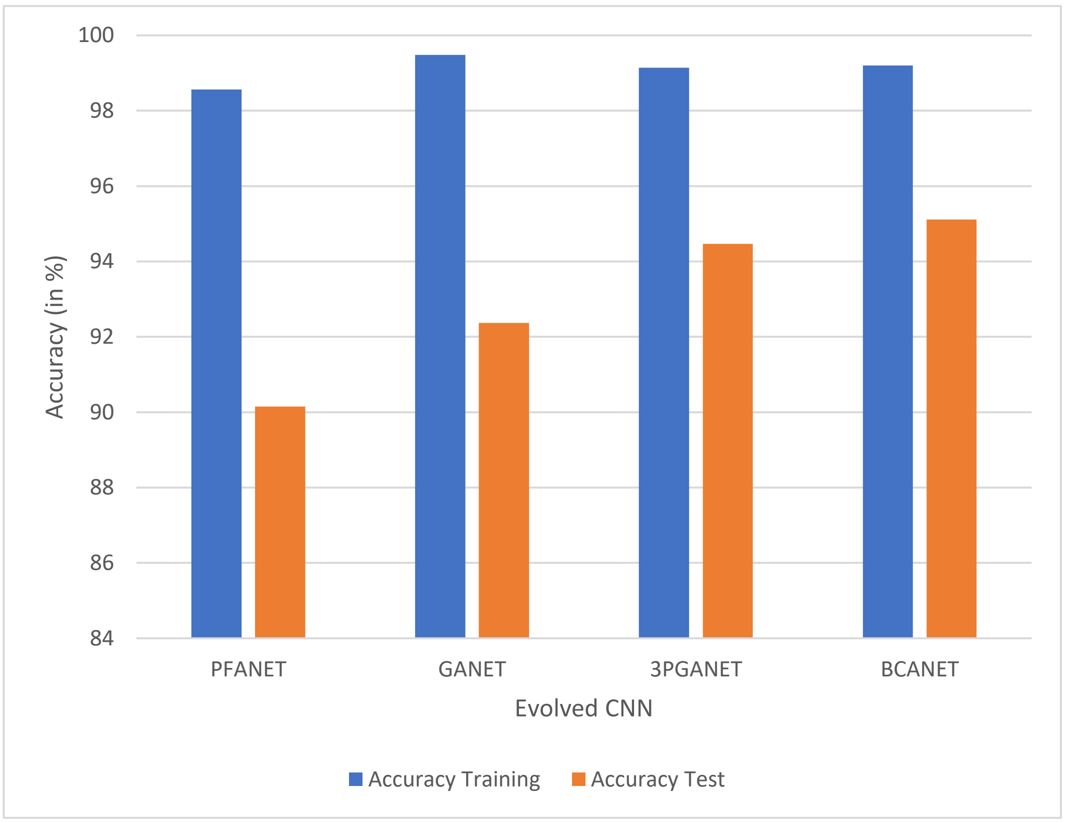


Figure 6.7: Training and test accuracies achieved by CNN architectures evolved with each metaheuristic.

Although the evolved hyperparameters showed significant improvements over an unevolved one, it was decided that a faster, quicker, and more space-efficient CNN be evolved. For this purpose, while evolving the hyperparameters, the learning rate was abruptly kept as high as 0.9. This was done to ensure a very quick training of the network.

Similarly, the layers in the network were artificially selected to be 3 to ensure that the network doesn't become too complex. Alongside the artificially selected hyperparameters, other hyperparameters were varied to produce the most efficient architecture.

It was observed that with the two hyperparameters being kept static, the other hyperparameters automatically align themselves to produce the most accurate results. The resulting network led to quicker training and lesser space complexity while maintaining the highest possible performance. This network was further applied to larger datasets and the results were noted. It was noted that, when applied to larger datasets, the network produced very high-performance results as compared to those that were expected.

6.5. PBCOA-CNN variation with dataset sizes

The evolution of CNN architectures with an artificial selection of features for better time and space complexity was automated and applied to datasets with higher amounts of data. The evolved architectures were tested on datasets with sizes of 200, 500, 800, and 1000 classes. The evolved architectures not only trained quickly due to the faster learning rate but also occupied lesser memory storage since layers of the architecture were limited to 3. Despite these constraints, the network evolved other hyperparameters in such a manner that CNN architectures with very high performance were trained.

Table 6.3: Table depicting performance of an evolved network of larger datasets in terms of accuracy and F1 Score.

	Performance	
Dataset Size (Number of Classes)	**Accuracy**	**F1 Score**
200	94.59	93.21
500	90.25	89.52
800	85.21	84.56
1000	79.51	79.48

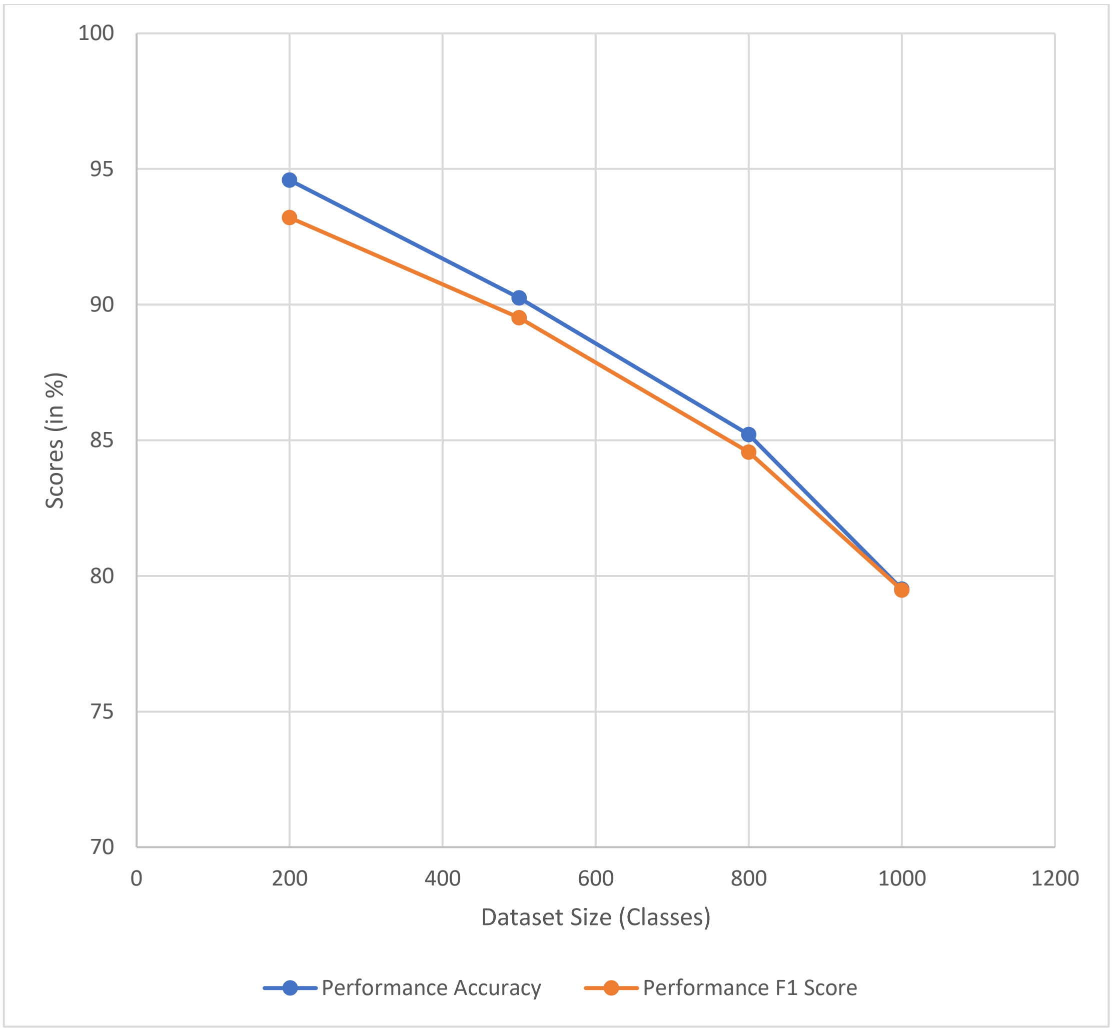


Figure 6.8: Performance analysis of evolved architecture on a higher number of classes while giving them 200 epochs to train.

From table 11, it is observed that as the dataset sizes are increased, the accuracies and F1 scores of the dataset seem to show a downward trend. As expected, the data shows an exponential decrease in performance. Hence, it is advisable to evolve the networks before training on larger datasets.

As expected, the training times were considerably lesser than what was expected, the memory usage was minimal as well. The performance with these parameters static did tend to somewhat show an unexpected decline with an increase in dataset sizes but it is expected that with the use of more generalized approaches like transfer learning techniques, the network can be evolved both specifically and generally.

6.6. Conclusion

Thus, we conclude that the evolution of Artificial Neural Networks should be highly desirable before experimenting on the entire dataset. The evolved CNN architectures tend to outperform their unevolved CNN counterparts in almost all respects. While dealing with large datasets, it should be desirable that we first evolve proper CNN architectures for the task at hand and then pursue the training. This helps the AI model to perform well on the task at hand.

Chapter 7: Conclusions and Future Scope

In our research work, we proposed the “Development of new Soft-Computing Based Approach for Landmark Recognition”. We approached the research work by development of a new evolutionary metaheuristic named ‘Parallel Bat Colony Optimization Algorithm’ (PBCOA) and applying it to automatically evolve the architectures of Convolutional Neural Networks for Landmark Recognition. We have also benchmarked the newly developed PBCOA against the state-of-the-art nature-inspired evolutionary metaheuristics and its supremacy was established.

7.1. Conclusions

Chapter 1 introduces the fields of Artificial Intelligence, Machine Learning, Deep Learning and Artificial Neural Networks. The topic of landmark recognition with the help of soft computing is introduced and the problem formulation is presented. Also, the first chapter describes the organization of the thesis.

Chapter 2 presents the literature survey conducted in the field of landmark recognition using AI. An extensive survey is presented that describes how performances in the field of landmark recognition fared over the times. Also, a short survey of the classical machine learning approaches is conducted along with its comparison with the Convolutional Neural Networks.

Chapter 3 proposes integrated CNN approaches along with three different evolutionary metaheuristics namely Paddy Field Algorithm, Genetic Algorithm and 3 Parent Genetic Algorithm. It is established, how the evolution of Convolutional Neural Networks using evolutionary metaheuristics tends to evolve CNN architectures that are seemingly very different from the ones used regularly and still perform much better than the regularly used CNN architectures.

Chapter 4 proposes a new soft computing nature-inspired evolutionary metaheuristic namely the Parallel Bat Colony Optimization Algorithm and its comparison is drawn with state-of-the-art optimization algorithms. Its supremacy over the other algorithms is properly established via the standard CEC Benchmarking Functions and all the algorithmic comparisons are drawn inspired from the natural phenomenon of Bat livelihoods.

Chapter 5 proposes the integration of PBCOA with CNN to evolve architectures for geographical landmark recognition. All the data related to performance of CNNs is provided and it has been described how the evolution of networks using PBCOA produces highly efficient CNN architectures that tend to overperform the architectures evolved using regular Bat Algorithm even.

Chapter 6 is the comparative analysis of all the proposed approaches, datasets used, augmentation preferences and other things. The overall comparative analysis clearly shows supremacy of PBCOA integrated CNN over others and suggests the evolution of networks before training.

Chapter 7, that is, this chapter tends to conclude all the research work and intends to provide future scope for the current thesis work. Also, some brief details have been mentioned about the publications that came out as a result of the current research work. All the research that has been published has been summarised in the current section and a future scope provided.

For the research work, first an extensive survey of the existent techniques for landmark recognition was conducted. This survey was communicated to 'International Conference on Innovations, Research and Challenges in Emerging Technologies' (IRCET 2022) held in November 2022 and the article was presented as a conference article and is currently under process for publication in Springer CCIS (Communications in Computing and Information Sciences) Conference Proceedings.

We also developed a self-designed Convolutional Neural Network that was aimed at improving accuracy in the field of geographical landmark recognition. An article in this regard was communicated to the "3rd International Conference (online) on Innovations in Communication Computing and Sciences (ICCS 2021)". The conference article was presented and is currently available online in the "American Institute of Physics Conference Proceedings".

Based on the amalgamation of two realms of NIC i.e. Evolutionary Metaheuristics and Artificial Neural Networks, we have developed a 3PGANET based on the search for the best neural architecture of Convolutional Neural Networks by the evolutionary metaheuristic named 3 Parent Genetic Algorithm. We have compared the performance of the evolution of CNN architectures from the regular 2-parent genetic algorithm and our 3 Parent Genetic Algorithm. Similarly, we evolved a CNN with Paddy Field Algorithm thus creating a PFANET.

We also worked on the VGG-16 transfer learning technique for Geographical Landmark Recognition. In this research work, we used the VGG-16 deep learning model trained on the ImageNet dataset. We used it to train a geographical landmarks dataset and a positive transfer of knowledge was seen. A conference article was communicated in this regard and accepted at the RACS 2022 (Recent Advances in Computing Sciences), Lovely Professional University. The article was accepted, presented, and will soon be published in Taylor and Francis Conference Proceedings.

Also, we worked to develop a Parallel Bat Colony Optimization Algorithm (PBCOA) based on the concept of the introduction of colonies in the regular bat algorithm metaheuristic. It was benchmarked on 30 CEC 14 test benchmark functions and the results were compared with those of 17 algorithms available in the literature. In most regards, PBCOA performed the best out of all the other algorithms giving the best performance on 16 out of the provided 30 benchmark functions to be tested on. We then used the proposed Parallel Bat Colony Optimization Algorithm for evolving Convolutional Neural Networks on popular Landmarks Recognition datasets. A journal article has been accepted in the reputed Springer journal "Soft Computing" with an impact factor of 3.7.

It has been seen that Landmark datasets are usually very complex and building models to predict them correctly is not a very easy task. So, we have deployed evolutionary computation for the improvement of existing CNNs. We optimize the hyperparameters of CNN with evolutionary metaheuristics to get the maximum possible performance. We have used the accuracy metric in our research for evaluation purposes since it is the most basic and intuitive metric to evaluate the performance of an AI model.

We published a Book chapter named “Landmark Recognition Using Ensemble-Based Machine Learning Models” in the Book entitled “Machine Learning and Data Analytics for Predicting, Managing, and Monitoring Disease” by the IGI Global publishers. The book has been included in the Web of Science database.

Two research articles were presented at International Conferences which will be published in Web of Science-indexed Conference Proceedings. The first conference article entitled “Geographical Landmark Recognition: Using CNN-Based approach to improve accuracy” was presented at the “3rd International Conference (online) on Innovations in Communication Computing and Sciences (ICCS 2021)”. It will be published in the “American Institute of Physics Conference Proceedings” which is a Web of Science Indexed Conference Proceedings.

The second conference article that was entitled “Landmark Recognition with Artificial Intelligence: A State of the Art”, was presented at the “International Conference on Innovation, Research, and Challenges in Emerging Technologies (IRCET-2021)”. It will be published in the “Communications in Computer and Information Science Conference Proceedings”. It is a Web of Science Indexed Conference Proceedings.

A third conference article entitled “Analysis of CNNs built using Transfer Learning and Data Augmentation on Landmark Recognition datasets” was presented at the conference “Recent Advances in Computing Sciences (RACS 2022)”. It will be published in the “Taylor and Francis” Conference Proceedings which is a Scopus Indexed Conference Proceeding.

Finally, two journal articles were published in Web of Science Indexed journals. The first one entitled “Evolving CNN with Paddy Field Algorithm for Geographical Landmark Recognition” was published in the MDPI journal “Electronics”. This was published in March 2022. The second one entitled “Automated evolution of CNN with 3PGA for geographical landmark recognition” was published in “Journal of Intelligent and Fuzzy systems” in September 2022. Both journals are indexed by the Web of Science as well as Scopus Indexing.

One journal article entitled “Development of VGG-16 transfer learning framework for geographical landmark recognition” has been published in the journal of “Intelligent

Decision Technologies” in May 2023 for publication. This article provides insight into how to use existent CNN architectures for the purpose of geographical landmark recognition.

The separate sections for all the list of publications, conference certificates and the list of attended workshops along with their certificates have also been provided in the following sections.

7.2. Future Scope

Soft computing contains both the fields of Evolutionary Algorithms and Artificial Neural Networks. However, the integration of two, to search for optimal architectures of ANNs remains a topic less researched into. Through this thesis, we claim that the two can be amalgamated to produce highly efficient AI models. However, the standards, the efficacy and the results continue to be a topic of debate. We propose that new integrated soft computing paradigms can be developed to create intensely meaningful Deep Learning models. New researchers can establish the new benchmarks in this research area.

This research area that includes the amalgamation of two Soft computing techniques namely Evolutionary metaheuristics and Artificial Neural Networks has a significant scope into the future. The evolution of ANNs with evolutionary metaheuristics has been shown in the thesis research work to produce highly efficient non-regular neural architectures. It is expected that just like the biological neurons, ANNs too may be evolved, with variety of hyperparametric combinations that may work in parallel to produce highly efficient, non-obvious artificial neural architectures.

In our research work, we proposed 1 nature-inspired-computing algorithm and 4 approaches for landmark recognition. The performance of the nature inspired computing algorithms is evaluated on standard CEC-Benchmark functions. The performances of all proposed landmark recognition approaches are evaluated on Google Landmarks Dataset V2. The proposed PBCOA approach can be used to solve different kinds of non-linear problems. The PBCOA can be used for machine learning model development, energy efficiency in wireless sensor networks, routing in wireless mesh networks. In future, all the 4 proposed approaches for landmark recognition can be implemented for medical diagnoses, plant disease detection and engineering defects detection. In future, new multi-objective metaheuristic algorithms could be developed and implemented in different application areas.

REFERENCES

[1] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition,* 2020, pp. 2575–2584.

[2] K. Zeng, Y. Li, Y. Xu, D. Wu, and N. Wu, "Introducing AI to Undergraduate Students via Computer Vision Projects," in *Thirty-Second AAAI Conference on Artificial Intelligence,* 2018.

[3] K. Ozaki and S. Yokoo, "Large-scale landmark retrieval/recognition under a noisy and diverse dataset," *arXiv preprint arXiv:1906.04087,* 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 2016. doi: 10.1007/978-3-319-46493-0_38.

[5] L. Wan, Y. Chen, H. Li, and C. Li, "Rolling-element bearing fault diagnosis using improved lenet-5 network," *Sensors (Switzerland),* vol. 20, no. 6, 2020, doi: 10.3390/s20061693.

[6] F. Yuesheng *et al.*, "Circular Fruit and Vegetable Classification Based on Optimized GoogLeNet," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3105112.

[7] S. Wan, Y. Liang, and Y. Zhang, "Deep convolutional neural networks for diabetic retinopathy detection by image classification," *Computers and Electrical Engineering,* vol. 72, 2018, doi: 10.1016/j.compeleceng.2018.07.042.

[8] D. Das, K. C. Santosh, and U. Pal, "Truncated inception net: COVID-19 outbreak screening using chest X-rays," *Phys Eng Sci Med*, vol. 43, no. 3, 2020, doi: 10.1007/s13246-020-00888-x.

[9] H. Huang, Y. Xie, G. Wang, L. Zhang, and W. Zhou, “DLNLF-net: Denoised local and non-local deep features fusion network for malignancy characterization of hepatocellular carcinoma,” *Comput Methods Programs Biomed*, vol. 227, 2022, doi: 10.1016/j.cmpb.2022.107201.

[10] S. Russell and P. Norvig, “Artificial intelligence: a modern approach,” 2002.

[11] Y. Ge, R. Zhang, X. Wang, X. Tang, and P. Luo, “Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5337–5345.

[12] A. Mikołajczyk and M. Grochowski, “Data augmentation for improving deep learning in image classification problem,” in *2018 international interdisciplinary PhD workshop (IIPhDW)*, 2018, pp. 117–122.

[13] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” *IEEE Trans Pattern Anal Mach Intell*, vol. 40, no. 6, pp. 1452–1464, 2017.

[14] K. Chen, C. Cui, Y. Du, X. Meng, and H. Ren, “2nd place and 2nd place solution to kaggle landmark recognition andretrieval competition 2019,” *arXiv preprint arXiv:1906.03990*, 2019.

[15] J. Kim, “A Study on Designing Metadata Standard for Building AI Training Dataset of Landmark Images,” *Journal of the Korean Society for Library and Information Science*, vol. 54, no. 2, pp. 419–434, 2020.

[16] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla, “24/7 place recognition by view synthesis,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1808–1817.

[17] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, “Large-scale image retrieval with attentive deep local features,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3456–3465.

[18] T. Weyand and B. Leibe, “Visual landmark recognition from internet photo collections: A large-scale evaluation,” *Computer Vision and Image Understanding*, vol. 135, pp. 1–15, 2015.

[19] F. Radenović, G. Tolias, and O. Chum, “Fine-tuning CNN image retrieval with no human annotation,” *IEEE Trans Pattern Anal Mach Intell*, vol. 41, no. 7, pp. 1655–1668, 2018.

[20] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, “Deep image retrieval: Learning global representations for image search,” in *European conference on computer vision*, 2016, pp. 241–257.

[21] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, “Revisiting oxford and paris: Large-scale image retrieval benchmarking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5706–5715.

[22] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, “Worldwide pose estimation using 3d point clouds,” in *European conference on computer vision*, 2012, pp. 15–29.

[23] S. Agarwal *et al.*, “Building rome in a day,” *Commun ACM*, vol. 54, no. 10, pp. 105–112, 2011.

[24] D. M. Chen *et al.*, “City-scale landmark identification on mobile devices,” in *CVPR 2011*, 2011, pp. 737–744.

[25] Y. Avrithis, G. Tolias, and Y. Kalantidis, “Feature map hashing: Sub-linear indexing of appearance and global geometry,” in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 231–240.

[26] J. Knopp, J. Sivic, and T. Pajdla, “Avoiding confusing features in place recognition,” in *European Conference on Computer Vision,* Berlin, Heidelberg: Springer.

[27] J. Philbin, “Lost in quantization: Improving particular object retrieval in large scale image databases,” in *2008 IEEE conference on computer vision and pattern recognition*, IEEE.

[28] H. Jegou, M. Douze, and C. Schmid, “Hamming embedding and weak geometric consistency for large scale image search,” in *European conference on computer vision*, Berlin, Heidelberg: Springer.

[29] J. Cao, “Landmark recognition with sparse representation classification and extreme learning machine,” *J Franklin Inst*, vol. 352, no. 10, pp. 4528–4545.

[30] A. Boiarov and E. Tyantov, “Large scale landmark recognition via deep metric learning,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management,*

[31] F. Magliani and A. Prati, “An accurate retrieval through R-MAC+ descriptors for landmark recognition,” in *Proceedings of the 12th International Conference on Distributed Smart Cameras,*

[32] F. Magliani, N. M. Bidgoli, and A. Prati, “A location-aware embedding technique for accurate landmark recognition,” in *Proceedings of the 11th International Conference on Distributed Smart Cameras,*

[33] F. Magliani, T. Fontanini, and A. Prati, “Efficient nearest neighbors search for large-scale landmark recognition,” in *International Symposium on Visual Computing*, Cham: Springer.

[34] R. Vandaele, “Landmark detection in 2D bioimages for geometric morphometrics: a multi-resolution tree-based approach,” *Sci Rep*, vol. 8, no. 1, pp. 1–13.

[35] P. Stefani and C. Oprean, “Fast Landmark Recognition in Photo Albums,” in *International Conference on Computer Analysis of Images and Patterns,* Cham: Springer.

[36] K. C. A. Cumbo, “Bee-inspired landmark recognition in robotic navigation,” in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion,*

[37] X.-B. Wen, H. Zhang, and Z.-T. Jiang, “Multiscale unsupervised segmentation of SAR imagery using the genetic algorithm,” *Sensors*, vol. 8, no. 3, pp. 1704–1711.

[38] J. Maeda, “Perceptual image segmentation using fuzzy-based hierarchical algorithm and its application to dermoscopy images,” in *2008 IEEE Conference on Soft Computing in Industrial Applications*, IEEE.

[39] H. Talbi, M. Batouche, and A. Draa, “A quantum-inspired evolutionary algorithm for multiobjective image segmentation,” *International Journal of Mathematical, Physical and Engineering Sciences*, vol. 1, no. 2, pp. 109–114.

[40] J. Moreira and L. D. F. Costa, “Neural-based color image segmentation and classification using self-organizing maps,” *Anais do IX Sibgrapi*, vol. 12, no. 6, pp. 47–54.

[41] A. Babenko, “Neural codes for image retrieval,” in *European conference on computer vision,* Cham: Springer.

[42] P. Suau, “Robust artificial landmark recognition using polar histograms,” in *Portuguese Conference on Artificial Intelligence,* Berlin, Heidelberg: Springer.

[43] Y. Chen, “SimpleDet: A Simple and Versatile Distributed Framework for Object Detection and Instance Recognition,” *Journal of Machine Learning Research*, vol. 20, no. 156, pp. 1–8.

[44] J. P. Crall, “Hotspotter—patterned species instance recognition,” in *2013 IEEE workshop on applications of computer vision (WACV*, IEEE.

[45] W. J. Scheirer, “Meta-recognition: The theory and practice of recognition score analysis,” *IEEE Trans Pattern Anal Mach Intell*, vol. 33, no. 8, pp. 1689–1695.

[46] M. Teichmann, “Detect-to-retrieve: Efficient regional aggregation for image search,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,*

[47] B. Zitová and J. Flusser, “Landmark recognition using invariant features,” *Pattern Recognit Lett*, vol. 20, no. 5, pp. 541–547.

[48] S. Srinivasan and L. Kanal, “Qualitative landmark recognition using visual cues,” *Pattern Recognit Lett*, vol. 18, no. 11–13, pp. 1405–1414.

[49] J. Cao, T. Chen, and J. Fan, “Fast online learning algorithm for landmark recognition based on BoW framework,” in *9th IEEE Conference on Industrial Electronics and Applications*, IEEE.

[50] T. Chen and K.-H. Yap, “Discriminative BoW framework for mobile landmark recognition,” *IEEE Trans Cybern*, vol. 44, no. 5, pp. 695–706.

[51] T. Chen, K.-H. Yap, and D. Zhang, “Discriminative soft bag-of-visual phrase for mobile landmark recognition,” *IEEE Trans Multimedia*, vol. 16, no. 3, pp. 612–622.

[52] X. Li, “Modeling and recognition of landmark image collections using iconic scene graphs,” in *European conference on computer vision*, Berlin, Heidelberg: Springer.

[53] T. Chen, K.-H. Yap, and D. Zhang, “Discriminative bag-of-visual phrase learning for landmark recognition,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP*, IEEE.

[54] T. Chen, “A survey on mobile landmark recognition for information retrieval,” in *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware,* IEEE.

[55] W.-C. Chen, A. Battestini, N. Gelfand, and V. Setlur, “Visual summaries of popular landmarks from community photo collections,” in *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, 2009, pp. 1248–1255.

[56] H. Li and S. X. Yang, “An evolutionary visual landmark recognition system,” *IEEE International Conference on Systems, Man and Cybernetics,* vol. 7.

[57] T. Chen, K.-H. Yap, and L.-P. Chau, “Integrated content and context analysis for mobile landmark recognition,” *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 21, no. 10, pp. 1476–1486.

[58] J. Guo, X. Mei, and K. Tang, “Automatic landmark annotation and dense correspondence registration for 3D human facial images,” *BMC Bioinformatics,* vol. 14, no. 1, p. 232.

[59] J. Kragskov, “Comparison of the reliability of craniofacial anatomic landmarks based on cephalometric radiographs and three-dimensional CT scans,” *The Cleft palate-craniofacial journal*, vol. 34, no. 2, pp. 111–116.

[60] O. Déniz, “Face recognition using histograms of oriented gradients,” *Pattern Recognit Lett,* vol. 32, no. 12, pp. 1598–1603.

[61] Asselen, E. F. Marieke, and A. Postma, “The influence of intentional and incidental learning on acquiring spatial knowledge during navigation,” *Psychol Res,* vol. 70, no. 2, pp. 151–156.

[62] R. A. Epstein and L. K. Vass, “Neural systems for landmark-based wayfinding in humans,” *Philosophical Transactions of the Royal Society B: Biological Sciences,* vol. 369, no. 1635, p. 20120533.

[63] F. Magliani, T. Fontanini, and A. Prati, “Landmark recognition: from small-scale to large-scale retrieval,” in *Recent Advances in Computer Vision*, Cham: Springer, pp. 237–259.

[64] Y.-T. Zheng, “Tour the world: building a web-scale landmark recognition engine,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE.

[65] J. Wu and J. M. Rehg, “Where am I: Place instance and category recognition using spatial PACT,” in *2008 Ieee Conference on Computer Vision and Pattern Recognition*, IEEE.

[66] H. Li and S. X. Yang, “A behavior-based mobile robot with a visual landmark-recognition system,” *IEEE/ASME transactions on mechatronics*, vol. 8, no. 3, pp. 390–400.

[67] H. Nasr and B. Bhanu, “Landmark recognition for autonomous mobile robots,” in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 1218–1223.

[68] P. E. Trahanias, S. Velissaris, and T. Garavelos, “Visual landmark extraction and recognition for autonomous robot navigation,” in *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications*, IEEE.

[69] J.-S. Lee, “In-pipe robot navigation based on the landmark recognition system using shadow images,” in *2009 IEEE international conference on robotics and automation*, IEEE.

[70] M. Elmogy and J. Zhang, “Robust real-time landmark recognition for humanoid robot navigation,” in *2008 IEEE International Conference on Robotics and Biomimetics*, IEEE.

[71] M. Mata, “A visual landmark recognition system for topological navigation of mobile robots,” *Proceedings 2001 ICRA. IEEE*

International Conference on Robotics and Automation (Cat, no. 01CH37164). Vol. 2.

[72] T. Liu, “Flexible FTIR spectral imaging enhancement for industrial robot infrared vision sensing,” *IEEE Trans Industr Inform,* vol. 16, no. 1, pp. 544–554.

[73] S. X. Yang and H. Li, “An autonomous mobile robot with fuzzy obstacle avoidance behaviors and a visual landmark recognition system,” in *7th International Conference on Control, Automation, Robotics and Vision, 2002. ICARCV 2002*, IEEE.

[74] Q. V Do and L. C. Jain, “A visual landmark recognition system for autonomous robot navigation,” in *2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA’06,* IEEE.

[75] R. C. Luo, H. Potlapalli, and D. W. Hislop, “Neural network based landmark recognition for robot navigation,” in *Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation*, IEEE.

[76] M. R. Peterson, T. E. Doom, and M. L. Raymer, “Ga-facilitated knn classifier optimization with varying similarity measures,” in *2005 IEEE congress on evolutionary computation,* vol. 3, IEEE.

[77] S.-W. Lin, “Particle swarm optimization for parameter determination and feature selection of support vector machines,” *Expert Syst Appl,* vol. 35, no. 4, pp. 1817–1824.

[78] F. Samadzadegan, H. Hasani, and T. Schenk, “Simultaneous feature selection and SVM parameter determination in classification of hyperspectral imagery using ant colony optimization,” *Canadian Journal of Remote Sensing*, vol. 38, no. 2, pp. 139–156.

[79] E. Houby, E. MF, N. I. R. Yassin, and S. Omran, “A hybrid approach from ant Colony optimization and K-nearest neighbor for classifying datasets using selected features,” *Informatica*, vol. 41, no. 4.

[80] Y. Wei, “An improved grey wolf optimization strategy enhanced SVM and its application in predicting the second major,” *Math Probl Eng*.

[81] K. Sasirekha and K. Thangavel, “Optimization of K-nearest neighbor using particle swarm optimization for face recognition,” *Neural Comput Appl*, vol. 31, no. 11, pp. 7935–7944.

[82] Z. Tao, “GA-SVM based feature selection and parameter optimization in hospitalization expense modeling,” *Appl Soft Comput*, vol. 75, pp. 323–332.

[83] R. Al-wajih, “Binary Grey Wolf Optimizer with K-Nearest Neighbor classifier for Feature Selection,” in *2020 International Conference on Computational Intelligence (ICCI*, IEEE.

[84] S. Binitha, S. S. Sathya, and others, “A survey of bio inspired optimization algorithms,” *International journal of soft computing and engineering*, vol. 2, no. 2, pp. 137–151, 2012.

[85] A. Singh, S. Kumar, S. S. Walia, and S. Chakravorty, “Face Recognition: A Combined Parallel BB-BC & PCA Approach to Feature Selection,” *International Journal of Computer Science & Information Technology*, vol. 2, no. 2, pp. 1–5, 2015.

[86] A. Singh, S. Kumar, A. Singh, and S. S. Walia, “Three-parent GA: A Global Optimization Algorithm.,” *Journal of Multiple-Valued Logic & Soft Computing*, vol. 32, 2019.

[87] S. Kumar, A. Singh, and S. Walia, “Parallel Big Bang–Big Crunch Global Optimization Algorithm: Performance and its Applications to routing in WMNs,” *Wirel Pers Commun*, vol. 100, no. 4, pp. 1601–1618, 2018.

[88] A. Singh, S. Kumar, A. Singh, and S. S. Walia, “Parallel 3-Parent Genetic Algorithm with Application to Routing in Wireless Mesh Networks,” *Implementations and Applications of Machine Learning*, vol. 782, p. 1, 2020.

[89] X.-S. Yang, “A new metaheuristic bat-inspired algorithm,” in *Nature inspired cooperative strategies for optimization (NICSO 2010),* Springer, 2010, pp. 65–74.

[90] E. O. K. and E. I, “A new optimization method: Big Bang-Big Crunch”,” *Advances in Engineering Software*, vol. 37, pp. 106–111.

[91] S. He, Q. H. Wu, and J. R. Saunders, “Group search optimizer: an optimization algorithm inspired by animal searching behavior,” *IEEE transactions on evolutionary computation*, vol. 13, no. 5, pp. 973–990, 2009.

[92] A. Kaveh and N. Farhoudi, “A new optimization method: dolphin echolocation,” *Advances in Engineering Software*, vol. 59, pp. 53–70.

[93] C. Sur, S. Sharma, and A. Shukla, “Egyptian vulture optimization algorithm–a new nature inspired meta-heuristics for knapsack problem,” in *Proceedings of 9th International Conference on Computing and Information Technology, (IC2IT2013*, King Mongkut’s University of Technology North Bangkok, Springer, pp. 227–237.

[94] H. Hernández and C. Blum, “Distributed graph coloring: an approach based on the calling behavior of japanese tree frogs,” *Swarm Intelligence*, vol. 6, no. 2, pp. 117–150.

[95] R. S. Parpinelli and H. S. Lopes, “An eco-inspired evolutionary algorithm applied to numerical optimization,” in *Proceedings of Third World Congress on Nature and Biologically Inspired Computing (NaBIC,* Salamanca: IEEE, pp. 466–471.

[96] C. J. Bastos Filho, L. N. F. B., A. J. Lins, A. I. Nascimento, and M. P. Lima, “A novel search algorithm based on fish school behavior,” in

Proceedings of IEEE International Conference on Systems, Man and Cybernetics, SMC 2008, Singapore, pp. 2646–2651.

[97] J. A. B. F. Carmelo, B. de L. N. Fernando, J. C. C. L. Anthony, I. S. N. Antonio, and P. L. Marilia, "A novel search algorithm based on fish school behavior," in *IEEE International Conference on Systems, Man and Cybernetics, Singapore*, pp. 2646–2651.

[98] Y. X. S, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation*, in Series Title : Lecture Notes in Computer Science. , Springer, pp. 240–249.

[99] X. S. Yang, K. M., and X. He, "Multi-objective flower algorithm for optimization," *Procedia Comput Sci*, vol. 18, no. Supplement C, pp. 861–868.

[100] C. Ferreira, "Gene expression programming in problem solving," in *Proceedings of Soft Computing and Industry: Recent Applications*, London: Springer, pp. 635–653.

[101] A. Mozaffari, A. Fathi, and S. Behzadipour, "The great salmon run: a novel bioinspired algorithm for artificial system design and optimisation," *International Journal of Bio-Inspired Computation*, vol. 4, no. 5, pp. 286–301.

[102] L. M. Zhang, C. Dahlmann, and Y. Zhang, "Human-inspired algorithms for continuous function optimization," in *Proceedings of IEEE International Conference on Intelligent Computing and Intelligent Systems*, Shanghai, China, pp. 318–321.

[103] A. R. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecol Inform*, vol. 1, no. 4, pp. 355– 366.

[104] R. D. Maia, L. N. Castro, and W. M. Caminhas, "Bee colonies as model for multimodal continuous optimization: The optbees algorithm," in

Proceedings of IEEE Congress on Evolutionary Computation (CEC, Brisbane, QLD, Australia, pp. 1–8.

[105] T. C. Havens, C. J. Spain, N. G. Salmon, and J. M. Keller, “Roach infestation optimization,” in *Proceedings of Swarm Intelligence Symposium, SIS 2008, IEEE, St*, Louis, MO, USA, pp. 1–7.

[106] S. H. Jung, “Queen-bee evolution for genetic algorithms,” *Electron Lett*, vol. 39, no. 6, pp. 575–576.

[107] M. M. Eusuff and K. E. Lansey, “Optimization of water distribution network design using the shuffled frog leaping algorithm,” *J Water Resour Plan Manag*, vol. 129, no. 3, pp. 210–225.

[108] R. Hedayatzadeh, F. A. Salmassi, M. Keshtgari, R. Akbari, and K. Ziarati, “Termite colony optimization: A novel approach for optimizing continuous problems,” in *Proceedings of 18th Iranian Conference on Electrical Engineering (ICEE*, Iran: IEEE, Isfahan, pp. 553–558.

[109] A. Ahmadi-Javid, “June. Anarchic Society Optimization: A human-inspired method,” in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, IEEE, pp. 2586–2592.

[110] P. Civicioglu, “Artificial cooperative search algorithm for numerical optimization problems”,” *Inf Sci (N Y)*, vol. 229, pp. 58–76.

[111] P. Civicioglu, “Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm”,” *Comput Geosci*, vol. 46, pp. 229–247.

[112] M. O’Neill and C. Ryan, “Grammatical evolution”,” *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349–358.

[113] A. H. Kashan, “League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships”,” *Appl Soft Comput*, vol. 16, pp. 171–200.

[114] Y. Xu, Z. Cui, and J. Zeng, “Social emotional optimization algorithm for nonlinear constrained optimization problems,” in *International Conference on Swarm, Evolutionary and Memetic Computing*, Berlin, Heidelberg: Springer, pp. 583–590.

[115] P. Civicioglu, “Backtracking search optimization algorithm for numerical optimization problems,” *Appl Math Comput*, vol. 219, no. 15, pp. 8121–8144.

[116] E. Atashpaz-Gargari and C. Lucas, “Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition”,” in *Evolutionary computation, CEC 2007. IEEE Congress on IEEE*, pp. 4661–4667.

[117] K. Bansal and A. Singh, “Geographical landmark recognition: Using CNN-Based approach to improve accuracy,” in *AIP Conference Proceedings*, 2022, p. 50007.

[118] D. S. Hochba, “Approximation algorithms for NP-hard problems,” *ACM Sigact News*, vol. 28, no. 2, pp. 40–52, 1997.

[119] K. Bansal *et al.*, “Evolving CNN with Paddy Field Algorithm for Geographical Landmark Recognition,” *Electronics (Basel)*, vol. 11, no. 7, 2022, doi: 10.3390/electronics11071075.

[120] M. A. Z. Raja, F. H. Shah, M. Tariq, I. Ahmad, and others, “Design of artificial neural network models optimized with sequential quadratic programming to study the dynamics of nonlinear Troesch’s problem arising in plasma physics,” *Neural Comput Appl*, vol. 29, no. 6, pp. 83–109, 2018.

[121] K. Bansal and A. Singh, “Automated evolution of CNN with 3PGA for geographical landmark recognition,” *Journal of Intelligent & Fuzzy Systems*, no. Preprint, pp. 1–12.

[122] D. A. Montecino, C. A. Perez, and K. W. Bowyer, “Two-Level Genetic Algorithm for Evolving Convolutional Neural Networks for Pattern Recognition,” *IEEE Access*, vol. 9, pp. 126856–126872, 2021.

[123] B. Fielding and L. Zhang, “Evolving image classification architectures with enhanced particle swarm optimisation,” *IEEE Access*, vol. 6, pp. 68560–68575, 2018.

[124] U. Premaratne, J. Samarabandu, and T. Sidhu, “A new biologically inspired optimization algorithm,” in *2009 international conference on industrial and information systems (ICIIS)*, 2009, pp. 279–284.

[125] A. Lydia and S. Francis, “Adagrad—an optimizer for stochastic gradient descent,” *Int. J. Inf. Comput. Sci*, vol. 6, no. 5, 2019.

[126] T. Zhuangzhuang, Z. Ronghui, H. Jiemin, and Z. Jun, “Adaptive learning rate CNN for SAR ATR,” in *2016 CIE International Conference on Radar (RADAR)*, 2016, pp. 1–5.

[127] Z. Zhang, “Improved adam optimizer for deep neural networks,” in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–2.

[128] L. Bottou, “Stochastic gradient descent tricks,” in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 421–436.

[129] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[130] Y. LeCun, Y. Bengio, and others, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[131] O. K. Erol and I. Eksin, “A new optimization method: big bang–big crunch,” *Advances in Engineering Software*, vol. 37, no. 2, pp. 106–111, 2006.

[132] G. Claeys, “The" survival of the fittest" and the origins of social darwinism,” *J Hist Ideas*, vol. 61, no. 2, pp. 223–240, 2000.

[133] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, “Eco: Efficient convolution operators for tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6638–6646.

[134] “What is the Convolutional Neural Network Architecture?,” Analytics Vidhya.

[135] K. Uchida, M. Tanaka, and M. Okutomi, “Coupled convolution layer for convolutional neural network,” *Neural Networks*, vol. 105, pp. 197–205, 2018.

[136] J. Chang *et al.*, “A mix-pooling CNN architecture with FCRF for brain tumor segmentation,” *J Vis Commun Image Represent*, vol. 58, pp. 316–322, 2019.

[137] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[138] S. Phelps and M. Köksalan, “An interactive evolutionary metaheuristic for multiobjective combinatorial optimization,” *Manage Sci*, vol. 49, no. 12, pp. 1726–1738, 2003.

[139] M. Y. Rafiq, G. Bugmann, and D. J. Easterbrook, “Neural network design for engineering applications,” *Computers & Structures* , vol. 79, no. 17, pp. 1541–1552, 2001.

[140] L. Xie and A. Yuille, “Genetic cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1379–1388.

[141] C.-K. Ting, “An analysis of the effectiveness of multi-parent crossover,” in *International Conference on Parallel Problem Solving from Nature*, 2004, pp. 131–140.

[142] R. Smigrodzki, J. Parks, and W. D. Parker, “High frequency of mitochondrial complex I mutations in Parkinson’s disease and aging,” *Neurobiol Aging*, vol. 25, no. 10, pp. 1273–1281, 2004.

[143] Y. Zhang, M. Sakamoto, and H. Furutani, “Effects of population size and mutation rate on results of genetic algorithm,” in *2008 Fourth International Conference on Natural Computation*, 2008, pp. 70–75.

[144] T.-P. Hong and H.-S. Wang, “A dynamic mutation genetic algorithm,” in *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No. 96CH35929)*, 1996, pp. 2000–2005.

[145] S. Reardon, “Genetic details of controversial’three-parent baby’revealed,” *Nature News*, vol. 544, no. 7648, p. 17, 2017.

[146] A. Singh, S. S. Walia, and S. Kumar, “P3PGA: Multi-Population 3 Parent Genetic Algorithm and its Application to Routing in WMNs.,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, 2017.

[147] F. U. M. Ullah, A. Ullah, K. Muhammad, I. U. Haq, and S. W. Baik, “Violence detection using spatiotemporal features with 3D convolutional neural network,” *Sensors*, vol. 19, no. 11, p. 2472, 2019.

[148] Y. Li *et al.*, “Exploiting kernel sparsity and entropy for interpretable CNN compression,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2800–2809.

[149] P. M. Radiuk and others, “Impact of training set batch size on the performance of convolutional neural networks for diverse datasets,” *Information Technology and Management Science*, vol. 20, no. 1, pp. 20–24, 2017.

[150] N. N. Aizenberg and I. N. Aizenberg, “Fast-convergence learning algorithms for multi-level and binary neurons and solution of some image

processing problems," in *International Workshop on Artificial Neural Networks*, 1993, pp. 230–236.

[151] S. Kwon and others, "A CNN-assisted enhanced audio signal processing for speech emotion recognition," *Sensors*, vol. 20, no. 1, p. 183, 2020.

[152] D. Xu, K. Tu, Y. Wang, C. Liu, B. He, and H. Li, "FCN-engine: Accelerating deconvolutional layers in classic CNN processors," in *Proceedings of the International Conference on Computer-Aided Design*, 2018, pp. 1–6.

[153] D. Wang, X. Wang, M.-K. Kim, and S.-Y. Jung, "Integrated optimization of two design techniques for cogging torque reduction combined with analytical method by a simple gradient descent method," *IEEE Trans Magn*, vol. 48, no. 8, pp. 2265–2276, 2012.

[154] O. Wichrowska *et al.*, "Learned optimizers that scale and generalize," in *International Conference on Machine Learning*, 2017, pp. 3751–3760.

[155] K. Bansal and A. S. Rana, "Landmark Recognition Using Ensemble-Based Machine Learning Models," in *Machine Learning and Data Analytics for Predicting, Managing, and Monitoring Disease*, IGI Global, 2021, pp. 64–74.

[156] S. Mugemanyi, Z. Qu, F. X. Rugema, Y. Dong, C. Bananeza, and L. Wang, "Optimal reactive power dispatch using chaotic bat algorithm," *IEEE Access*, vol. 8, pp. 65830–65867, 2020.

[157] M. C. Rose, B. Styr, T. A. Schmid, J. E. Elie, and M. M. Yartsev, "Cortical representation of group social communication in bats," *Science (1979)*, vol. 374, no. 6566, p. eaba9584, 2021.

[158] J. S. Kanwal and J. P. Rauschecker, "Auditory cortex of bats and primates: managing species-specific calls for social communication," *Front Biosci*, vol. 12, p. 4621, 2007.

[159] H. Wu, L. Gong, T. Jiang, J. Feng, and A. Lin, “Echolocation call frequencies of bats vary with body temperature and weather conditions,” *Anim Behav*, vol. 180, pp. 51–61, 2021.

[160] E. Gillam and M. B. Fenton, “Roles of acoustic social communication in the lives of bats,” *Bat bioacoustics*, pp. 117–139, 2016.

[161] M. Shehab *et al.*, “A Comprehensive Review of Bat Inspired Algorithm: Variants, Applications, and Hybridization,” *Archives of Computational Methods in Engineering*, pp. 1–33, 2022.

[162] S. M. Elsayed, R. A. Sarker, D. L. Essam, and N. M. Hamza, “Testing united multi-operator evolutionary algorithms on the CEC2014 real-parameter numerical optimization,” in *IEEE Congress on Evolutionary Computation (CEC), IEEE*, pp. 1650–1657.

[163] R. Tanabe and A. S. Fukunaga, *Improving the search performance of SHADE using linear population size reduction*. IEEE Congress on Evolutionary Computation (CEC.

[164] C. Xu, H. Huang, and S. Ye, “A differential evolution with replacement strategy for real parameter numerical optimization,” in *IEEE Congress on Evolutionary Computation (CEC*, pp. 1617–1624.

[165] B. Y. Qu, J. J. Liang, J. M. Xiao, and Z. G. Shang, “Memetic differential evolution based on fitness Euclidean-distance ratio,” in *IEEE Congress on Evolutionary Computation (CEC*, pp. 2266–2273.

[166] Z. Hu, Y. Bao, and T. Xiong, “Partial opposition-based adaptive differential evolution algorithms: evaluation on the CEC 2014 benchmark set for real-parameter optimization”,” in *IEEE Congress on Evolutionary Computation (CEC*, pp. 2259–2265.

[167] Z. Li, Z. Shang, B. Y. Qu, and J. J. Liang, *Differential evolution strategy based on the constraint of fitness values classification*. IEEE Congress on Evolutionary Computation (CEC.

[168] I. Erlich, J. L. Rueda, S. Wildenhues, and F. Shewarega, *Evaluating the mean-variance mapping optimization on the IEEE-CEC 2014 test suite*. IEEE Congress on Evolutionary Computation (CEC.

[169] D. Molina, B. Lacroix, and F. Herrera, *Influence of regions on the memetic algorithm for the CEC'2014 Special Session on real-parameter single objective optimization*. IEEE Congress on Evolutionary Computation (CEC.

[170] R. D. Maia, L. N. Castro, and W. M. Caminhas, "Real-parameter optimization with OptBees," in *IEEE Congress on Evolutionary Computation (CEC*, pp. 2649–2655.

[171] P. Preux, R. Munos, and M. Valko, "Bandits attack function optimization," in *IEEE Congress on Evolutionary Computation (CEC*.

[172] C. Yu, L. Kelley, S. Zheng, and Y. Tan, "Fireworks algorithm with differential mutation for solving the CEC 2014 competition problems," in *IEEE Congress on Evolutionary Computation (CEC*, pp. 3238–3245.

[173] L. Chen, Z. Zheng, H. L. Liu, and S., *Xie An evolutionary algorithm based on covariance matrix leaning and searching preference for solving CEC 2014 benchmark problems*. IEEE Congress on Evolutionary Computation (CEC.

[174] R. Mallipeddi, G. Wu, M. Lee, and P. N. Suganthan, "Gaussian adaptation based parameter adaptation for differential evolution," in *IEEE Congress on Evolutionary Computation (CEC*, pp. 1760–1767.

[175] D. Yashesh, K. Deb, and S. Bandaru, "Non-uniform mapping in real-coded genetic algorithms," in *IEEE Congress on Evolutionary Computation (CEC*, pp. 2237–2244.

[176] R. Poláková, J. Tvrdík, and P. Bujok, *Controlled restart in differential evolution applied to CEC 2014 benchmark functions*. IEEE Congress on Evolutionary Computation (CEC.

[177] S. Kumar, S. S. Walia, and A. Singh, “Parallel big bang-big crunch algorithm,” *International Journal of Advanced Computing*, vol. 46, no. 3, pp. 1330–1335, 2013.

[178] X. Wu *et al.*, “Surprisingly popular algorithm-based adaptive Euclidean distance topology learning PSO,” *arXiv preprint arXiv:2108.11173*, 2021.

[179] A. Singh, S. Kumar, A. Singh, and S. S. Walia, “Parallel 3-parent genetic algorithm with application to routing in wireless mesh networks,” *Implementations and Applications of Machine Learning*, pp. 1–28, 2020.

[180] S. Kumar, A. Singh, and S. Walia, “Parallel Big Bang–Big Crunch global optimization algorithm: performance and its applications to routing in WMNs,” *Wirel Pers Commun*, vol. 100, pp. 1601–1618, 2018.

LIST OF PUBLICATIONS

There has been significant output through our research work that has been compiled as under. First, an account of all the accepted and published journal articles is provided. Then, the conference articles are described along with the conferences they were presented in and the conference proceedings, they shall be published in. At the end is the account of a book chapter that too was published. All the research has been compiled in detailed format as well as in a tabulated format below:

Journal Articles

Four journal articles have been published through our research work. The articles are compiled as under:

1. The article entitled “A new parallel bat colony optimization algorithm and its application for evolving CNN architectures with artificial selection” was published in the Journal of “Soft Computing” by Springer Nature. The journal has an impact factor of 3.732 and is indexed by Scopus and Web of Science.
2. The article entitled “Automated evolution of CNN with 3PGA for geographical landmark recognition” was published in the “Journal of Intelligent and Fuzzy Systems” by IOS Press. The journal impact factor is 1.737 and it is indexed under Scopus and Web of Science.
3. The article entitled “Development of VGG-16 transfer learning framework for geographical landmark recognition” was published in the journal “Intelligent Decision Technologies” by IOS Press. The journal has a Scopus indexing and an additional ESCI indexing from the Web of Science.
4. The article entitled “Evolving CNN with Paddy Field Algorithm for Geographical Landmark Recognition” was published in the MDPI journal “Electronics”. The journal impact factor is 2.690 and it is indexed under Scopus and Web of Science.

Conference Articles

Three conference articles have been accepted and presented through our research work. The articles are compiled as under:

1. The conference article entitled “Geographical Landmark Recognition: Using CNN-Based approach to improve accuracy” was presented in the “3rd International Conference (online) on Innovations in Communication Computing and Sciences (ICCS 2021)” held in August 2021. The conference article through it has been published in the American Institute of Physics Conference Proceedings that is a Web of Science Indexed Conference Proceedings. The conference proceedings impact factor is 0.40.
2. The conference article entitled “Landmark Recognition with Artificial Intelligence: A State of the Art” was presented in the “International Conference on Innovations, Research and Challenges in Emerging Technologies” (IRCET 2022) held in November 2021 and the article is currently under process for publication in “Springer CCIS (Communications in Computing and Information Sciences) Conference Proceedings.”. The Conference Proceedings is a Web of Science Indexed Conference Proceedings.
3. The conference article entitled “Analysis of CNNs built using Transfer Learning and Data Augmentation on Landmark Recognition datasets” was presented in the “Recent Advances in Computing Sciences (RACS 2022)” held in November 2022 and the article is currently under process for publication in “Taylor and Francis Conference Proceedings.”. The Conference Proceedings is a Scopus Indexed Conference Proceedings.

Book Chapter

One book chapter entitled “Landmark Recognition Using Ensemble-Based Machine Learning Models” was published in the IGI Global publishers book named “Machine Learning and Data Analytics for Predicting, Managing, and Monitoring Disease” published in June 2021.

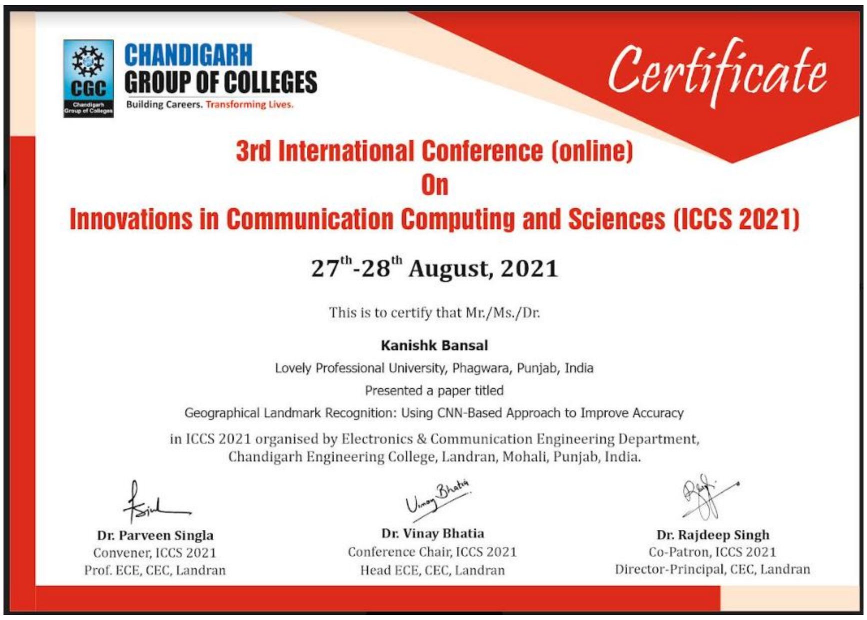
Detail of the journal/ Book / Book chapter/ website link	**Year of Publication**	**Indexing of journal (Scopus/ WOS index etc.)**	**Main findings or conclusion relevant to proposed research work**	**Remarks**
https://link.springer.com/article/10.1007/s00500-023-08846-x	2023	Scopus and WOS	A new parallel bat colony optimization algorithm and its application for evolving CNN architectures with artificial selection	*Journal Article* accepted in a Springer journal "Soft Computing". The Journal impact factor is 3.7
https://content.iospress.com/articles/intelligent-decision-technologies/idt230048	2023	Scopus and WOS	Development of VGG-16 transfer learning framework for geographical landmark recognition	*Journal Article* published in the journal "Intelligent Decision Technologies".
https://content.iospress.com/articles/journal-of-intelligent-and-fuzzy-systems/ifs221473	2022	Scopus and WOS	Automated evolution of CNN with 3PGA for geographical landmark recognition	*Journal Article* published in "Journal of Intelligent and Fuzzy Systems".
https://www.mdpi.com/2079-9292/11/7/1075	2022	Scopus and WOS	Evolving CNN with Paddy Field Algorithm for Geographical Landmark Recognition	*Journal Article* published in MDPI ‘Electronics’

https://aip.scitation.org/doi/abs/10.1063/5.0105666	2021	Scopus and WOS	Geographical Landmark Recognition: Using CNN-Based approach to improve accuracy	*Conference article* presented in ICCS 2021. It is published in the American Institute of Physics Conference Proceedings.
Landmark Recognition with Artificial Intelligence: A State of the Art	2021	Scopus and WOS	Landmark Recognition with Artificial Intelligence: A State of the Art	*Conference article* accepted and presented for CCIS Springer Proceedings and is under process for publication.
Analysis of CNNs built using Transfer Learning and Data Augmentation on Landmark Recognition datasets	2022	Scopus	Analysis of CNNs built using Transfer Learning and Data Augmentation on Landmark Recognition datasets	*Conference article* accepted and presented in RACS 2022 and is under process for publication.
https://www.igi-global.com/chapter/landmark-recognition-using-ensemble-based-machine-learning-models/286243	2021	NA	Landmark Recognition Using Ensemble-Based Machine Learning Models	*Book Chapter* published in IGI Global Book named “Machine Learning and Data Analytics for Predicting, Managing, and Monitoring Disease”

LIST OF CONFERENCES

The certificates to the attended conferences are provided below:

1. International Conference in Innovations in Communication Computing and Sciences (ICCS 2021)



CHANDIGARH GROUP OF COLLEGES
Building Careers. Transforming Lives.

Certificate

3rd International Conference (online)
On
Innovations in Communication Computing and Sciences (ICCS 2021)

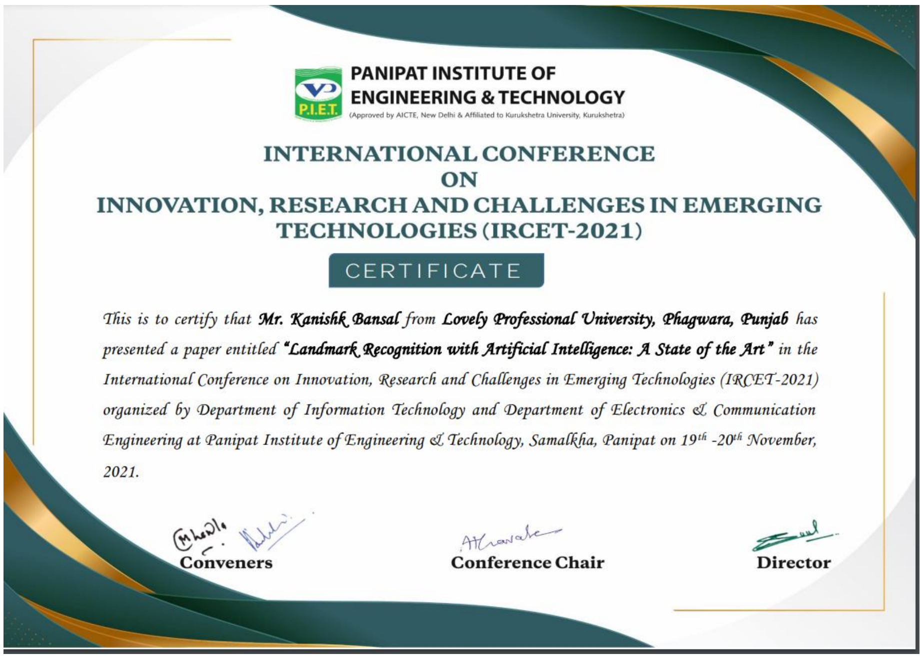
27th-28th August, 2021

This is to certify that Mr./Ms./Dr.

Kanishk Bansal
Lovely Professional University, Phagwara, Punjab, India
Presented a paper titled
Geographical Landmark Recognition: Using CNN-Based Approach to Improve Accuracy
in ICCS 2021 organised by Electronics & Communication Engineering Department,
Chandigarh Engineering College, Landran, Mohali, Punjab, India.

Dr. Parveen Singla	Dr. Vinay Bhatia	Dr. Rajdeep Singh
Convener, ICCS 2021	Conference Chair, ICCS 2021	Co-Patron, ICCS 2021
Prof. ECE, CEC, Landran	Head ECE, CEC, Landran	Director-Principal, CEC, Landran

2. International Conference on Innovation, Research and Challenges in Emerging Technologies (IRCET-2021)



PANIPAT INSTITUTE OF ENGINEERING & TECHNOLOGY
(Approved by AICTE, New Delhi & Affiliated to Kurukshetra University, Kurukshetra)

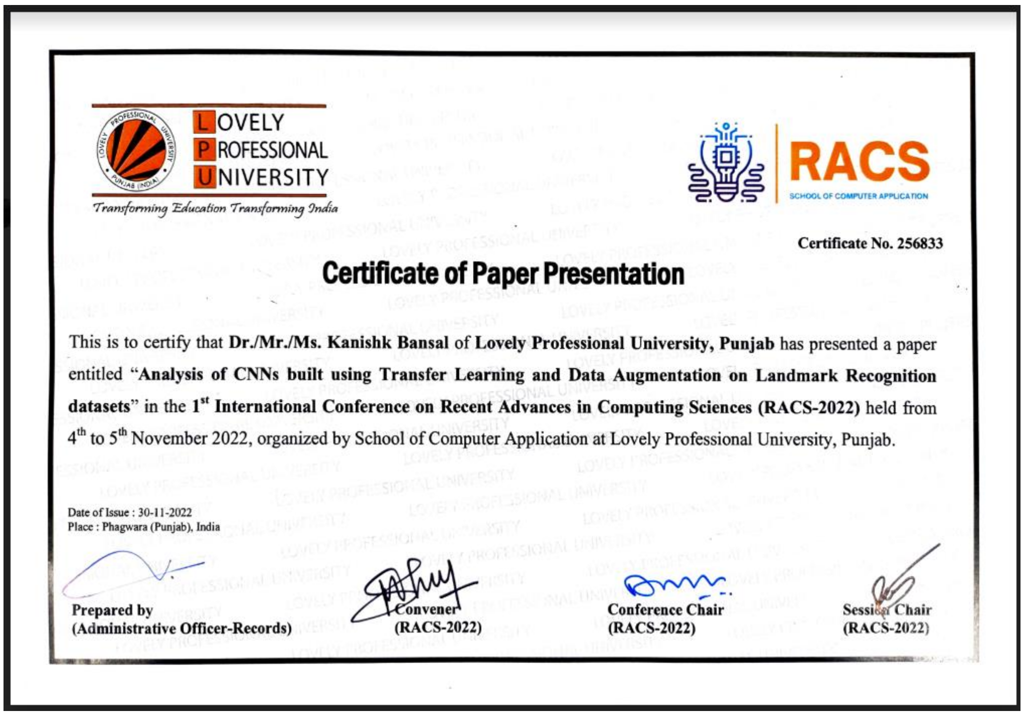
INTERNATIONAL CONFERENCE
ON
INNOVATION, RESEARCH AND CHALLENGES IN EMERGING TECHNOLOGIES (IRCET-2021)

CERTIFICATE

*This is to certify that **Mr. Kanishk Bansal** from **Lovely Professional University, Phagwara, Punjab** has presented a paper entitled **"Landmark Recognition with Artificial Intelligence: A State of the Art"** in the International Conference on Innovation, Research and Challenges in Emerging Technologies (IRCET-2021) organized by Department of Information Technology and Department of Electronics & Communication Engineering at Panipat Institute of Engineering & Technology, Samalkha, Panipat on 19th -20th November, 2021.*

Conveners Conference Chair Director

3. International Conference on Recent Advances in Computing Sciences (RACS-2022)



LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

RACS

SCHOOL OF COMPUTER APPLICATION

Certificate No. 256833

Certificate of Paper Presentation

This is to certify that **Dr./Mr./Ms. Kanishk Bansal** of **Lovely Professional University, Punjab** has presented a paper entitled "**Analysis of CNNs built using Transfer Learning and Data Augmentation on Landmark Recognition datasets**" in the **1st International Conference on Recent Advances in Computing Sciences (RACS-2022)** held from 4th to 5th November 2022, organized by School of Computer Application at Lovely Professional University, Punjab.

Date of Issue : 30-11-2022

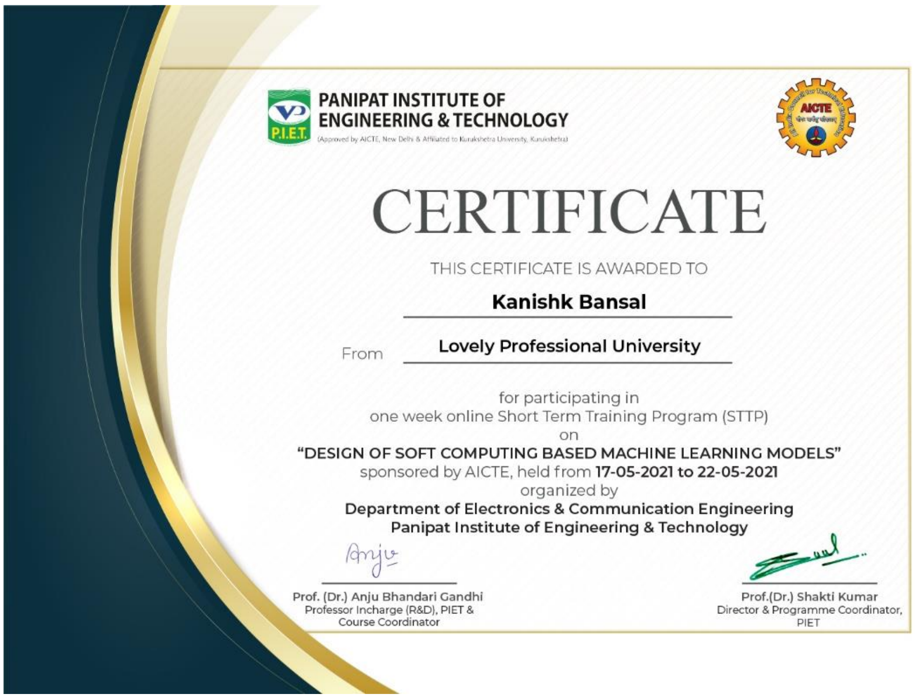
Place : Phagwara (Punjab), India

[illegible] [illegible] [illegible] [illegible]

Prepared by (Administrative Officer-Records) | **Convener (RACS-2022)** | **Conference Chair (RACS-2022)** | **Session Chair (RACS-2022)**

LIST OF WORKSHOPS

1. One-week online Short Term Training Program (STTP) on "Design of Soft Computing Based Machine Learning Models".



PANIPAT INSTITUTE OF ENGINEERING & TECHNOLOGY
(Approved by AICTE, New Delhi & Affiliated to Kurukshetra University, Kurukshetra)

CERTIFICATE

THIS CERTIFICATE IS AWARDED TO

Kanishk Bansal

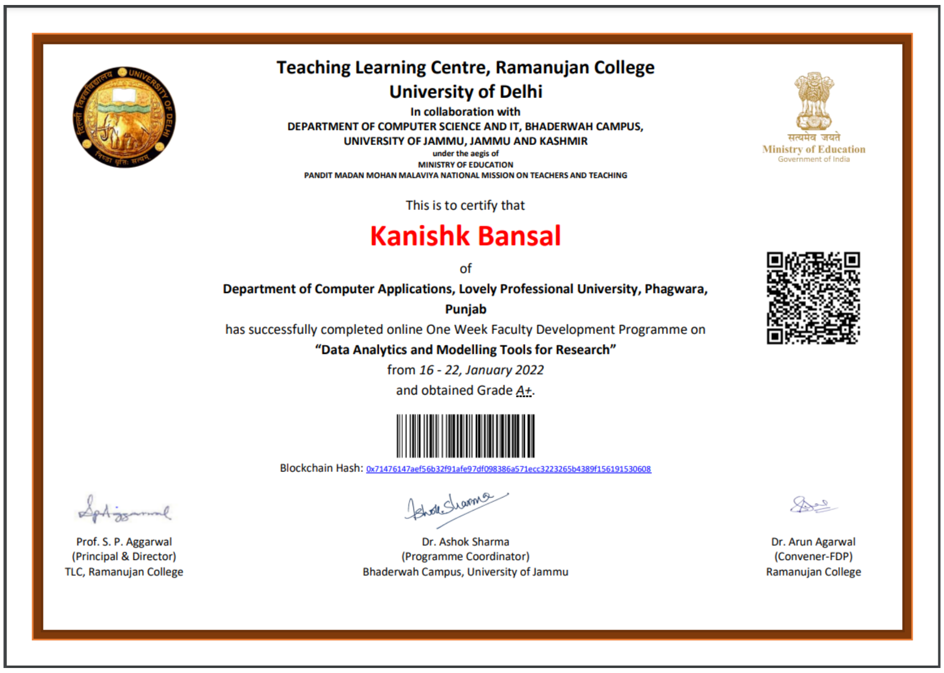
From **Lovely Professional University**

for participating in
one week online Short Term Training Program (STTP)
on
"DESIGN OF SOFT COMPUTING BASED MACHINE LEARNING MODELS"
sponsored by AICTE, held from **17-05-2021 to 22-05-2021**
organized by
Department of Electronics & Communication Engineering
Panipat Institute of Engineering & Technology

Prof. (Dr.) Anju Bhandari Gandhi
Professor Incharge (R&D), PIET &
Course Coordinator

Prof.(Dr.) Shakti Kumar
Director & Programme Coordinator,
PIET

2. A+ grade in the one Week Faculty Development Programme on "Data Analytics and Modelling Tools for Research"



Teaching Learning Centre, Ramanujan College
University of Delhi
In collaboration with
DEPARTMENT OF COMPUTER SCIENCE AND IT, BHADERWAH CAMPUS,
UNIVERSITY OF JAMMU, JAMMU AND KASHMIR
under the aegis of
MINISTRY OF EDUCATION
PANDIT MADAN MOHAN MALAVIYA NATIONAL MISSION ON TEACHERS AND TEACHING

सत्यमेव जयते
Ministry of Education
Government of India

This is to certify that

Kanishk Bansal

of

Department of Computer Applications, Lovely Professional University, Phagwara, Punjab

has successfully completed online One Week Faculty Development Programme on
"Data Analytics and Modelling Tools for Research"
from *16 - 22, January 2022*
and obtained Grade *A+*.

Blockchain Hash: 0x71476147aef56b32f91afe97df098386a571ecc3223265b4389f156191530608

Prof. S. P. Aggarwal
(Principal & Director)
TLC, Ramanujan College

Dr. Ashok Sharma
(Programme Coordinator)
Bhaderwah Campus, University of Jammu

Dr. Arun Agarwal
(Convener-FDP)
Ramanujan College