  
**Tribhuvan University**

**Institute of Science and Technology**

**Seminar Report**

**On**

**Bird Species Identification Using Artificial Neural Network**

**Submitted to**

**Central Department of Computer Science & Information Technology**

**Tribhuvan University, Kirtipur**

**Kathmandu, Nepal**

**Submitted by**

**Madan Nath**

Roll No.: 14 / 074

**In partial fulfillment of the requirement for Master's Degree in Computer Science and Information Technology (M.Sc. CSIT), 1st Semester**

**September 02, 2018**



**Tribhuvan University**

**Institute of Science and Technology**

**Student’s Declaration**

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

**Madan Nath**

Date: September 02, 2018

**Supervisor’s Recommendation**

I hereby recommend that this Seminar report is prepared under my supervision by **Mr. Madan Nath** entitled “**Bird Species Identification Using Artificial Neural Network**” be accepted as fulfilment in partial requirement for the degree of Master's of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in computer science.

... ... ... ... ... ... ... ... ... … … …

**Asst. Prof. Lalita Sthapit**

Central Department of Computer Science

and Information Technology

Date: September 02, 2018



**Tribhuvan University**

**Institute of Science and Technology**

**LETTER OF APPROVAL**

This is certify that the seminar report prepared by Mr. Madan Nath entitled “**Bird Species Identification Using Artificial Neural Network**” in partial fulfillment of the requirements for the degree of Master's of Science in Computer Science and Information technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

**Evaluation Committee**

……………………………… ………………………………

**Asst. Prof. Nawaraj Paudel** **Asst. Prof. Lalita Sthapit**

Head of Central Department of Computer (Supervisor)

Science & Information Technology Central Department of Computer Science

& Information Technology

..…………………………………

(External)

Date: September 02, 2018

# ACKNOWLEDGEMENTS

Almost one and half month of preparation for seminar, I’m finally able to succeed on my deeds. Taking idea from workshop, which made me to plan about the seminar and got a chance to perceive new ideas and experience.

With immense please, I submit my deepest gratitude to the Central Department of Computer Science and Technology, Tribhuvan University.

First of all, I would like to thank Asst. Prof. **Nawaraj Paudel**, Head of Central Department of Computer Science and Information Technology, Tribhuvan University for his encouragement and guidance.

Additionally, I would like to thanks Asst. Prof. **Lalita Sthapit** for her valuable guidance in carrying out this work under her effective supervision and enlightenment.

Secondly with deep sense of gratefulness, I express my genuine thanks to my respected supervisor Asst. Prof**. Tej Bhadur Shahi,** for their excellent guidance, patience, trust and correction of my seminar work.

I want to specially thank **Mr. Ashok Kumar Pant**, a Senior Machine Learning and software Engineer and Researcher for his inspiration, insightful discussion, thoughtful guidance and critical comments.

Thanks to all my friends for their support and finally, I want to thank my family, for their love, continuous support and encouragement.

# ABSTRACT

There are different species of birds in the world. Proper identification of the birds is challenging. So, neural network can play crucial role in identification and classification of birds. For the identification of species of birds, physical appearance and posture like size and shape can be taken as input features. Shape is major fact that can computer recognize very well under study. The body structure defines the shape of feather, shape of tail, shape of beak, shape of neck, shape of trunk, shape of claws and different pattern of colors. For classification a good set of features as a descriptor of image is required. Two important categories of the features are described, geometric and statistical features for extracting information from character images. The research primarily concerned with the problem of identification of different species of bird. Multilayer Perceptron (MLP) is used for classification. The principal contributions presented here are preprocessing, feature extraction and MLP classifiers. The Caltech data set contains 200 species (6,033 images) of birds under identification which have been used for this report. The strength of this seminar is efficient feature extraction and the comprehensive classification schemes using Scale Invariant Feature Transform (SIFT).

**Keywords:**

Bird Identification, ANN, MLP, SIFT, Image Preprocessing, Feature Extraction.

# Table of Contents

[ACKNOWLEDGEMENTS iv](#_Toc531588884)

[ABSTRACT v](#_Toc531588885)

[Table of Contents vi](#_Toc531588886)

[LIST OF FIGURES viii](#_Toc531588887)

[LIST OF TABLES ix](#_Toc531588888)

[LIST OF ABBREVIATIONS x](#_Toc531588889)

[Chapter-1 INTRODUCTION 1](#_Toc531588890)

[1.1 Introduction 1](#_Toc531588891)

[1.2 Artificial Neural Networks 2](#_Toc531588892)

[1.3 Real Life Applications of Artificial Neural Networks 3](#_Toc531588893)

[1.4 Challenges 3](#_Toc531588894)

[1.5 History of Artificial Neural Networks 4](#_Toc531588895)

[Chapter- 2 LITERATURE REVIEW 6](#_Toc531588896)

[2.1 Related Work 6](#_Toc531588897)

[2.2 Dataset Overview 7](#_Toc531588898)

[2.3 Preprocessing 8](#_Toc531588899)

[2.4 RBG to Grayscale Conversion 9](#_Toc531588900)

[2.5 Normalization 9](#_Toc531588901)

[2.5.1 Whitening: 10](#_Toc531588902)

[2.5.2 Mean Image Subtraction 11](#_Toc531588903)

[Chapter -3 FEATURE EXTRACTION TECHNIQUES 12](#_Toc531588904)

[3.1 Image Feature Extraction 12](#_Toc531588905)

[3.2 Scale Invariant Feature Transform (SIFT) 12](#_Toc531588906)

[Idea Of SIFT 13](#_Toc531588907)

[Chapter-4 MULTILAYER PERCEPTRON 16](#_Toc531588908)

[4.1 How does it Works 17](#_Toc531588909)

[4.2 Evaluation Metrics 18](#_Toc531588910)

[4.2.1 Confusion Matrix: 18](#_Toc531588911)

[4.2.2 Accuracy 20](#_Toc531588912)

[4.2.3 Precision 20](#_Toc531588913)

[4.2.4 Recall 20](#_Toc531588914)

[4.2.5 F-Score 21](#_Toc531588915)

[Chapter-5 EXPERIMENT RESULT AND ANALYSIS 23](#_Toc531588916)

[Chapter-6 CONCLUSION AND FUTURE SCOPE 24](#_Toc531588917)

[6.1 Conclusion 24](#_Toc531588918)

[6.2 Future scope 24](#_Toc531588919)

[Reference 25](#_Toc531588920)

[Appendix: 26](#_Toc531588921)

[A: Data preprocessing 26](#_Toc531588922)

[B: Feature Extracting (SIFT) 29](#_Toc531588923)

[C: MLP(Confusion Matrix) 31](#_Toc531588924)

# LIST OF FIGURES

[Figure 1 Physical neuron Figure 2 An Artificial Neuron 2](#_Toc531588512)

[Figure 3 Bird data set. 7](#_Toc531588513)

[Figure 4 Image processing Stages 8](#_Toc531588514)

[Figure 5 Image processing Stages Figure 6 Grayscale Image 9](#_Toc531588515)

[Figure 7 Whitening 10](#_Toc531588516)

[Figure 9 Image transforming to local features 13](#_Toc531588517)

[Figure 10 Gaussian Pre-filtering 14](#_Toc531588518)

[Figure 11 For each max or min found, output is the location and the scale. 14](#_Toc531588519)

[Figure 19 Architecture of MLP 16](#_Toc531588520)

[Figure 20 Working Phases of MLP 18](#_Toc531588521)

[Figure 21 Confusion Matrix 19](#_Toc531588522)

[Figure 22 Precision and Recall 21](#_Toc531588523)

# LIST OF TABLES

[Table 1: Multi-valued bird attributes. 8](#_Toc531422276)

# LIST OF ABBREVIATIONS

**ANN** Artificial Neural Network

**CR** Character Recognition

**GDMA** Gradient Descent with Momentum & Adaptive Learning Rate

**IPT** Image Processing Toolbox

**LMS** Least Mean Square

**LS** Least Square

**MAT** Medial Axis Transformation

**MLP** Multilayer Perceptron

**NNT** Neural Network Toolbox

**OCR** Optical Character Recognition

**RBF** Radial Basis Function

**RBFNN** Radial Basis Function Neural Network

**SVM** Support Vector Machine

**VLSI** Very Large-Scale Integration

**SIFT** Scale Invariant Feature Transform

**HOG** Histogram of oriented gradients

**LBP** Local Binary Pattern

**SURE** Speed-Up Robust Features

# Chapter-1 INTRODUCTION

## Introduction

Birds are unique living beings with their ability to fly. There are many types of birds and we see only a few of them around the place of human residence. Almost 9,000 to 10,000 species of birds are found in the world. Bird species identiﬁcation from images is an important and challenging problem with many applications in the real world such as environment protection and endangered animal rescue. There are also some other practical reasons to monitor birds. In order to evaluate the quality of our living environment it is important to obtain reliable information about the population of wild animals. Birds are numerous and sensitive to environmental changes also, and are easier to monitor than other species. The proper identification of the birds using Neural network is very challenging. A Neural Network is a computing structure consisting of a massively parallel interconnection of artificial adaptive neurons. The main advantages of neural networks are its ability to be trained automatically from examples, good performance with noisy data, possible parallel implementation, and efficient tools for learning large databases. There are many neural network-based recognition techniques like multilayer perceptron, radial basis function, recurrent networks, self-organizing maps, etc.

Bird identiﬁcation is a well-known problem to ornithologists, and is considered as a scientiﬁc task since antiquity. Ornithologists study birds; their existence in nature, their biology, their songs, their distribution, and their ecological impact. Besides the interesting results achieved with bioacoustics signals, this problem has been tackled also based on bird images. Compared with sound identiﬁcation, visual features are not well studied for bird classiﬁcation. The classiﬁcation problem can be stated as given a bird image, classify its species among a ﬁxed but large number of possibilities. The challenge of such a classiﬁcation task is due to the variation in the background and illumination since most of the images are gathered on birds’ natural habitat and in the birds pose since it's not possible to control rotation, scale, and angle of view while acquiring images. Visual properties (e.g., shape, color, marks, etc.) are important keys for bird recognition. Some researchers utilize these features to automatically identify birds. Local thresholding was applied to the HSV, GRAY, and RGB color spaces. Next, template matching using normal correlation and artiﬁcial neural networks (ANN) were developed in addition to image morphology. For this seminar data is taken from Caltech.edu [1]. It has altogether 6,033 number of bird’s image (200 categories) under study. The main objective of this Report is to investigate various features using SIFT feature extraction technique and to compare Neural Network based pattern recognition techniques namely Multilayer Feed-forward Network. Comparative Performance matrices are analyzed. The sub-problem field of bird identification is also addressed. The main objective of this seminar is to evaluate simple feature descriptors in bird images, and what is the expected classiﬁcation performance that can be achieved when dealing with a great number of bird species.

## Artificial Neural Networks

Artificial neural network is non-linear, parallel, distributed, highly connected network having capability of adaptivity, self-organization, fault tolerance, evidential response and Very Large Scale Integration (VLSI) implementation, which closely resembles with physical nervous system. Physical nervous system is highly parallel, distributed information processing system having high degree of connectivity with capability of self-learning. Human nervous system contains about 10 billion neurons with 60 trillion of interconnections. These connections are modified based on experience. It is something rough mathematical cartoon of how a biological neural network works. In biological brain we have individual cells called neurons, each neuron looks what other neuron looks at to what it’s neighbor has to say then it decides what it wants to say. In artificial neural network we have little mathematical functions we put them in some organized structure.

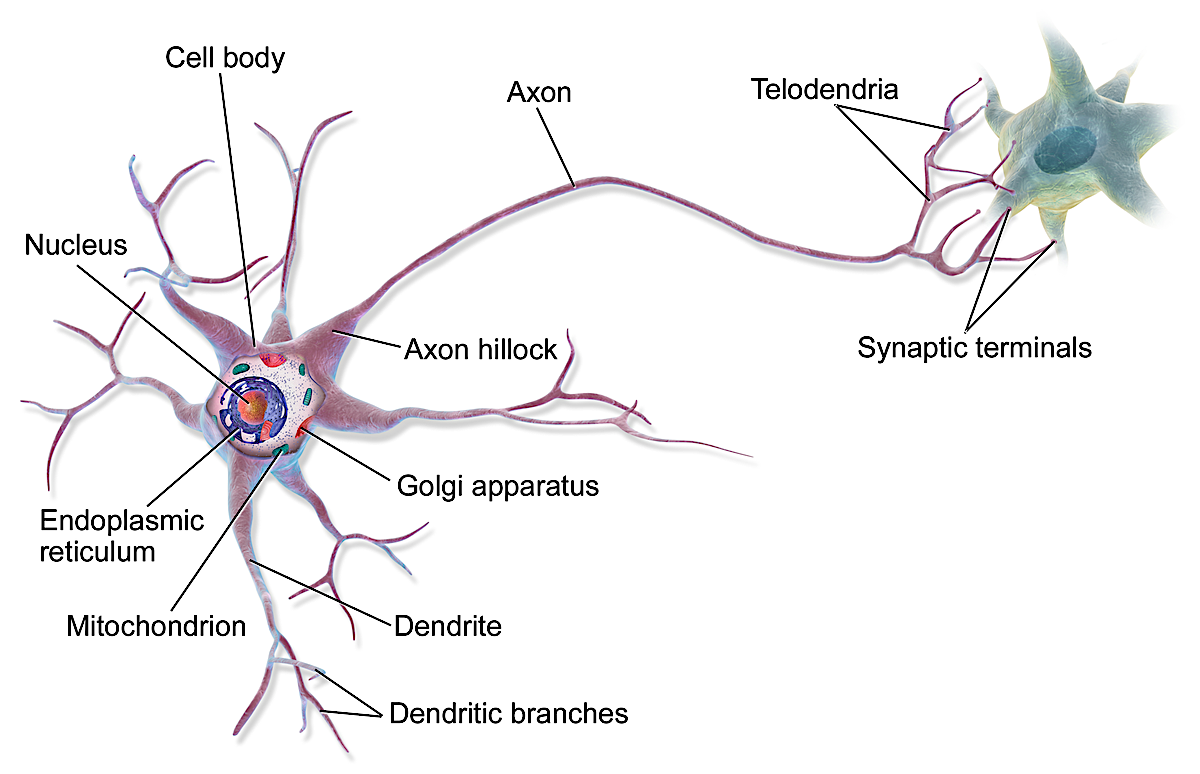
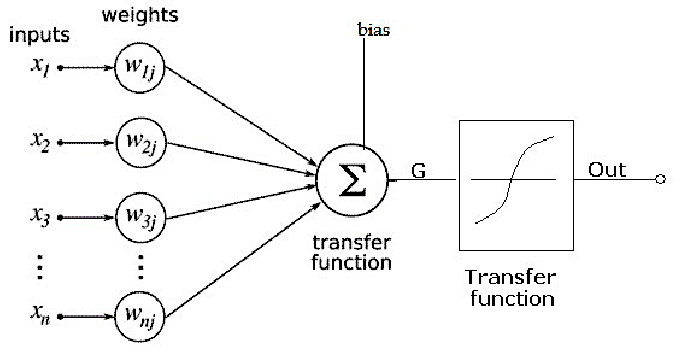
 

Figure 1 Physical neuron  Figure 2 An Artificial Neuron

Artificial neural networks are composed of interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Artificial neural networks may either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system. The real, biological nervous system is highly complex. Artificial neural network algorithms attempt to abstract this complexity and focus on what may hypothetically matter most from an information processing point of view. Another incentive for these abstractions, is to reduce the amount of computation required to simulate artificial neural networks, so as to allow one to experiment with larger networks and train them on larger data sets.

## Real Life Applications of Artificial Neural Networks

Some real-life applications of Artificial Neural Network are given below.

* Signal processing (e.g. adaptive echo cancelling).
* Control (e.g. manufacturing plants for controlling automated machines).
* Robotics (e.g. vision recognition).
* Pattern recognition (e.g. recognizing handwritten characters).
* Medicine (e.g. storing medical records based on case information, Medical Diagnostics).
* Speech production (e.g. reading text aloud).
* Speech recognition.
* Vision (e.g. face recognition, Autonomous Driving).
* Business (e.g. rules for mortgage decisions).
* Financial Applications (e.g. stock market prediction).
* Data Compression (e.g. images).
* Game Playing (e.g. chess, Tic-Tac-Toe).
* Spam Filtering and Fraud Detection.
* Virtual Personal Assistance.
* Weather Forecasting. etc.

## Challenges

Several bird species classification challenges with closely related, but different, task descriptions have been held during the last few years. The interest and participation in these challenges have been high which indicates that these are relevant problems and that there is a need to solve them. The challenges are usually to predict which species are present in a set of hidden labels, called the test set, and to submit the predicted species for each test data point for evaluation against the ground truth labels.

The main problem when training neural networks is deciding what features to use as input to the network. In particular, how to extract abstract features, which contain as much information about the original input as possible, but with a lower dimensionality to enable efficient training. The recognition task is carried out with Artificial Neural Network. Many geometric and statistical features are extracted from images so that the performance and accuracy of recognition system is achieved in the range of human ability of recognition. The system performs picture recognition by quantification of the picture into a mathematical vector entity using the geometrical and statistical properties of the character image. The sub problems in the domain of bird identification such as, noise removal, image binarization, object skeletonizing, size normalization, etc. have great impact on recognition procedure. These sub-problems are also addressed with the most suitable solutions in the literature for this type of research work.

For the Caltech dataset the annotation are as:

* **Same**: The bird in the image looks like it is of the same species as the exemplar.
* **Similar**: the bird in the image and the exemplar look similar, maybe of the same species.
* **Different**: the bird in image differs from the one in the exemplar
* **Difficult**: chosen if occlusion or scale differences make the comparison difficult.

## History of Artificial Neural Networks

The modern view of neural network began in the 1940s with the work of Warren McCulloch

and Walter Pitts, who showed that networks of artificial neurons could, in principle, compute

any arithmetic or logical function. McCulloch and Pitts were followed by Donald Hebb, who proposed that classical conditioning (as described by Pavlov in the experiment of food presenting to the dog) is present because of the properties of individual neurons.

Hebb proposed a mechanism for learning artificial neurons in the way of biological neurons. The first practical application of the artificial neural networks came in the late 1950s, with the invention of the perceptron network and associative learning rule by Frank Rosenblatt. Around 1956, Bernard Widrow and Ted Hoff introduced a new learning algorithm and used to train adaptive linear neural networks, which is similar in structure and capability to Rosan-4 blatt’s perceptron.

Another system besides perceptron was the Adaptive Linear Element (ADALINE) which was developed in 1960 by Widrow and Hoff (of Stanford University). Rosenblatt’s and Widrow’s network both suffer from limitation of applicability of networks only to linearly separable classes of problems. Limitations of these networks were publicized in the book by Marvin Minsky and Seymour Papert with the fact that there were no powerful

digital computers to do experiments.

So, for a decade, neural network research was largely suspended. Development of neural network dramatically increased from 1980, as there were powerful personal computers to do experiments and the development of multilayer backpropagation perceptron network. The most influenced publication of the backpropagation algorithm was by Devid Rumelhart and James Mclelland. This algorithm was the answer to the criticisms Minsky and Papert had made in the 1960s. Significant progress has been made in the field of neural networks-enough to attract a great deal of attention and fund further research. Advancement beyond current commercial applications appears to be possible, and research is advancing the field on many fronts. Neutrally based chips are emerging and applications to complex problems developing. Clearly, today is a period of transition for neural network technology. [2]

# Chapter- 2 LITERATURE REVIEW

## 2.1 Related Work

It is an ancient dream to make machines able to perform tasks like humans. In this seminar paper existing bird image classification journals commonly used in computer vision are reviewed. Various techniques have been proposed to predict the bird identification. Bird species are classified before but it becomes easiest after the use of Neural network with the pioneering work of McCulloch and Pitts on Nervous system. After 1990, image processing techniques and pattern recognition techniques were combined using artificial intelligence. Along with powerful computers and more accurate electronic equipment's such as scanners, cameras and electronic tablets, there came in efficient, modern use of methodologies such as artificial neural networks (ANNs), hidden Markov models (HMMs), fuzzy set reasoning etc.

In [3], Authors goal was to compare two different methods of classification: linear discriminant analysis and multinomial logistic regression to make the choice between the two methods easier, and to understand how do the two models behave under different data and group characteristics. The measure used to compare the performance of the two techniques was the overall classification accuracy. And to investigate the quality of prediction in terms of sensitivity and specificity, area under the ROC curve (AUC) is also examined.

In [4], Authors suggested that syllables are elementary building blocks of bird song. In sounds of many songbirds a large class of syllables can be approximated as amplitude and frequency varying brief sinusoidal pulses. In this article we test how well bird species can be recognized by comparing simple sinusoidal representations of isolated syllables. Results are encouraging and show that with limited sets of bird species a recognizer based on this signal model may already be sufficient.

Paper [5] presents a novel approach for bird species classification based on color features extracted from unconstrained images. This means that the birds may appear in different scenarios as well may present different poses, sizes and angles of view. The histogram bins are used as feature vectors to by a learning algorithm to try to distinguish between the different numbers of bird species. Experimental results on the CUB-200 dataset show that the segmentation algorithm achieves 75% of correct segmentation rate. Furthermore, the bird species classification rate varies between 90% and

8% depending on the number of classes considered.

## 2.2 Dataset Overview

To prepare this report, **Caltech-UCSD Birds 200** (CUB-200) data set is used. It contains 200 bird species with different classes and 6,033 images. Mostly the photos of different species of birds are found in North America. Ii is a challenging image dataset a was created to enable the study of subordinate categorization hat focus on basic level categories. The images were downloaded from the website Flickr and filtered by workers on Amazon Mechanical Turk. Each image is annotated with a bounding box, a rough bird segmentation, and a set of attribute labels. [1]

The large number of categories should make it an interesting dataset for subordinate categorization. Moreover, since it is annotated with bounding boxes, rough segmentations and attribute labels, it is also ideally suited for benchmarking systems where the users take an active part in the recognition process, as demonstrated below.

****

Figure 3 Bird data set.

Training and testing samples for recognition system are selected from each dataset. From total data set 70% for training and 30% for testing would be separated. [1]

Table 1: Multi-valued bird attributes.

|  |  |
| --- | --- |
| **Attribute** | **Values** |
| Bill shape | cone, all-purpose, dagger, hooked seabird, hooked, curved (up or down), spatulate, needle, specialized |
| Head Pattern | malar, eyebrow, capped, eye ring, unique pattern, striped, spotted, crested, masked, plain, eyeline |
| Belly Pattern | solid, striped, spotted, multi-colored |
| Wing shape | pointed-wings, tapered-wings, long-wings, rounded-wings, broad-wings |
| Shape | perching-like, tree-clinging-like, gull-like, duck-like, swallow-like, upright-perching water-like, sandpiper-like, upland-ground-like, chicken-like-marsh, pigeon-like, long-legged-like, hummingbird-like, hawk-like, owl-like |
| Size | small (5 - 9 in), very small (3 - 5 in), medium (9 - 16 in), very large (32 - 72 in), large (16 - 32 in) |
| Back color | buff, white, black, grey, brown, purple, pink, blue, iridescent, olive, rufous, yellow, green, red, orange |
| Tail pattern | striped, solid, spotted, multi-colored |
| Wing Pattern | striped, spotted, solid, multi-colored |

## 2.3 Preprocessing

Data preprocessing is applied to improve the quality of data. Data preprocessing includes data cleaning, data integration, data transformation and data reduction techniques. Cleaning is used to remove noisy data and missing values. Integration is used to extract data from multiple sources and storing as a single repository. Transformation transforms and normalizes the data in a consolidated form suitable for mining. Reduction reduces the data by adopting various techniques like dimensionality reduction, numerosity reduction and so on. Here data is being cleaned.

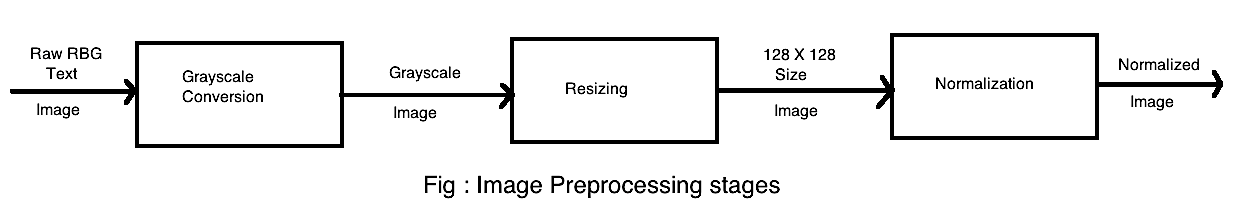


Figure 4 Image processing Stages

The large number of categories should make it an interesting dataset for subordinate categorization. Moreover, since it is annotated with bounding boxes, rough segmentations and attribute labels, it is also ideally suited for benchmarking systems. The raw images which are input to the system are preprocessed by subjecting to the preliminary operators such that it can be fed to the SIFT for further processing. The input cropped image are converted into 128 X 128. After resizing, to each image the sample mean image subtraction and standard division normalization is performed.

## 2.4 RBG to Grayscale Conversion

An RBG image is an image in which each pixel is specified by three values one each for the red, blue, and green components of the pixel scalar. The image will be converted to greyscale (range of gray shades from white to black) the computer will assign each pixel a value based on how dark it is. All the numbers are put into an array and the computer does computations on that array. Here, the RBG image with 24-bit true color is converted into 8-bit grayscale image using python code (importing os, sys, traceback, numpy library) as demonstration in Appendix.

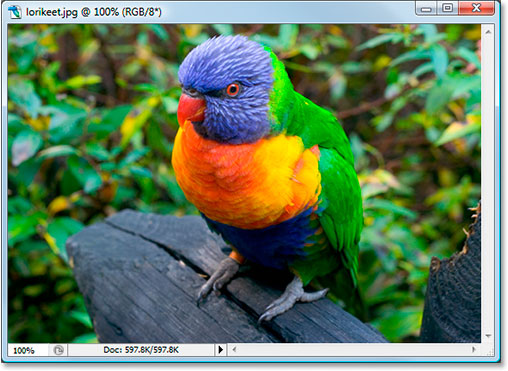
 

Figure 5 Image processing Stages  Figure 6 Grayscale Image

## 2.5 Normalization

It maps the original value of attribute in a predefined range. For this normalization process there are many normalization techniques like Min-Max normalization, Standard Scaler normalization, Decimal Scaling, Standard Deviation Method and by eliminating outliers. In Neural Network, we use continually by adding gradient error vector which is multiplied by learning rate. In our training data, if we don’t scale our features, the distributions of feature values would likely be different for each feature, and thus the learning rate would cause corrections in each dimension that would differ from one another.

### 2.5.1 Whitening:

A whitening transformation or sphering transformation is a [linear transformation](https://en.wikipedia.org/wiki/Linear_transformation) that transforms a vector of [random variables](https://en.wikipedia.org/wiki/Random_variables) with a known [covariance matrix](https://en.wikipedia.org/wiki/Covariance_matrix) into a set of new variables whose covariance is the [identity matrix](https://en.wikipedia.org/wiki/Identity_matrix), meaning that they are [uncorrelated](https://en.wikipedia.org/wiki/Uncorrelated) and each have [variance](https://en.wikipedia.org/wiki/Variance) 1. The transformation is called "whitening" because it changes the input vector into a [white noise vector](https://en.wikipedia.org/wiki/White_noise).

The whitening transformation does its operation as: The first operation decorrelates the data by pre-multiplying the data with the eigenvector matrix Et, calculated from the data co-variance. This decorrelation can be thought of as a rotation that reorients the data so that the principal axes of the data are aligned with the axes along which the data has the largest (orthogonal) variance. This rotation is essentially the same procedure as the oft-used Principal Components Analysis (PCA) and is shown in the middle row.

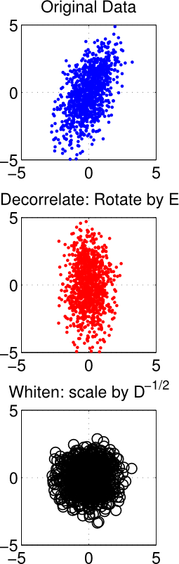


Figure 7 Whitening

The second operation, scaling by*D−1/2 D−1/2*can be thought of squeezing the data if the variance along a dimension is larger than one or stretching the data, If the variance along a dimension is less than one. The stretching and squeezing forms the data into a sphere about the origin. This scaling operation is depicted in the bottom row in the plot above.

### 2.5.2 Mean Image Subtraction

Image subtraction or pixel subtraction is a process whereby the digital numeric value of one pixel or whole image is subtracted from another image. This is primarily done for one of two reasons levelling uneven sections of an image such as half an image having a shadow on it, or detecting changes between two images. This detection of changes can be used to tell if something in the image moved.

# Chapter -3 FEATURE EXTRACTION TECHNIQUES

## 3.1 Image Feature Extraction

Feature extraction is one of the important stages of the image identification, it plays a very important role in the area of image processing. Before getting features, various image preprocessing techniques like binarization, thresholding, resizing, normalization etc. are applied on the sampled image. After that, feature extraction techniques are applied to get features that will be useful in classifying and recognition of images. Image feature is a simple image pattern, based on which we can describe what we see on the image. For example, bird eye will be a feature on an image of a bird. The main role of features in computer vision is to transform visual information into the vector space. This gives us possibility to perform mathematical operations on them.

In the field of images, features might be raw pixels for simple problems like digit recognition, however, in natural images, usage of simple image pixels are not descriptive enough. Instead there are two main steam to follow. One is to use hand engineered feature extraction methods (e.g. SIFT, VLAD, HOG, GIST, LBP). One of the prevalent hand engineered method is SIFT. The algorithm Scale-invariant feature transform (SIFT) is frequently applied in computer vision and a variety of multimedia tasks. It has been used for widely varying vision tasks from finding objects in images and videos on the one hand to the creation of point clouds of 3D scenes on the other hand. In this way SIFT is the best way of extraction features. The biggest problem seen with SIFT is that it is very complex and time consuming but gives better performance.

## 3.2 Scale Invariant Feature Transform (SIFT)

SIFT starts by detecting edges and corners in the image. On the resulted image, SIFT tries to find interesting points that are differentiating that image from the others. Then, out of each extracts a histograms where each of the bins is count of particular edge or corner orientation. These histograms can be concatenated or quantized into some smaller number of groups with a clustering method like K-means. SIFT is explained here very naively since the exact formulation is more complicated and requires some level of Computer Vision and Calculus knowledge.

**The main steps involved in the calculation of SIFT features are as follows:**

1. Extrema detection in a Laplacian-of-Gaussian (LoG) scale space to locate potential interest points.

2. Key point refinement by fitting a continuous model to determine precise location and scale.

3. Orientation assignment by the dominant orientation of the feature point from the directions of the surrounding image gradients.

4. Formation of the feature descriptor by normalizing the local gradient histogram.

### Idea Of SIFT

On SIFT operation Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters.

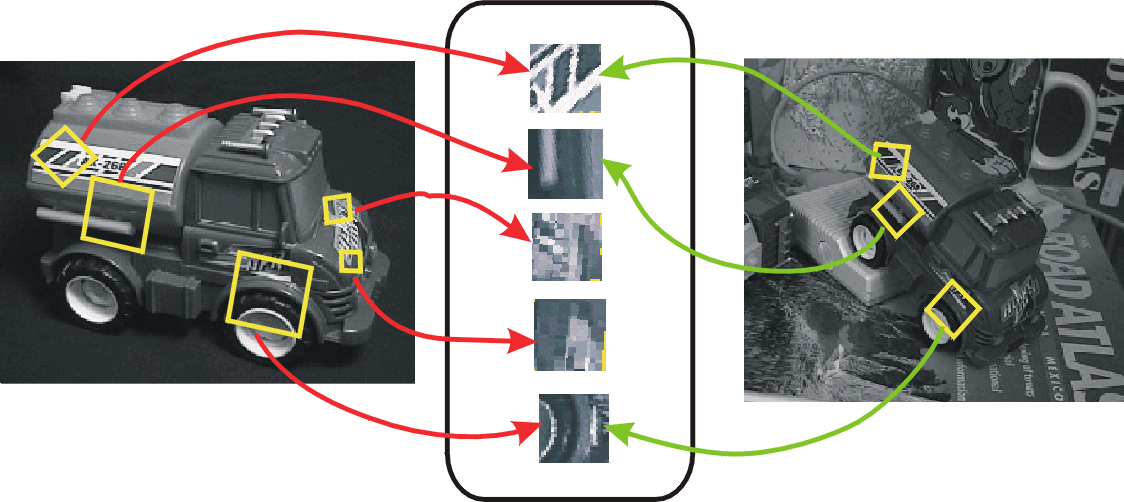


Figure 9 Image transforming to local features

**Step: 1 Scale-space extrema detection**

On this step Identification of locations and scales that can be repeatably assigned under different views of the same scene or object. It searches for stable features across multiple scales using a continuous function of scale. Prior work has shown that under a variety of assumptions, the best function is a Gaussian function. The scale space of an image is a function *L(x, y,σ)* that is produced from the convolution of a Gaussian kernel (at different scales) with the input image.

For Example: Subsampling with Gaussian pre-filtering.

Gaussian ½ Gaussian ¼ Gaussian 1/8

Figure 10 Gaussian Pre-filtering

|  |
| --- |
|  |

**Step: 2 Key point localization**

It Detect maxima and minima of difference-of-Gaussian in scale space. Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below.

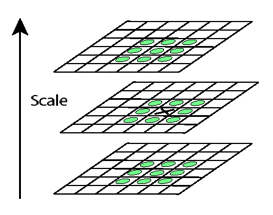
****

Figure 11 For each max or min found, output is the location and the scale.

* Sampling in scale for efficiency
* More scales evaluated, more key points found
* S < 3, stable key points increased too
* S > 3, stable key points decreased
* S = 3, maximum stable key points found

**Step: 3 Orientation assignment**

First, create histogram of local gradient directions at selected scale then Assign canonical orientation at peak of smoothed histogram and Each key specifies stable 2D coordinates (x, y, scale, orientation)

**Step: 4 Key point Descriptors**

* At this point, each key point has
  + location
  + scale
  + orientation
* Next is to compute a descriptor for the local image region about each key point that is
  + highly distinctive
  + invariant as possible to variations such as changes in viewpoint and illumination

# Chapter-4 MULTILAYER PERCEPTRON

A multilayer perceptron (MLP) is a [deep, artificial neural network](https://deeplearning4j.org/neuralnet-overview). They are composed of an input layer to receive the signal, an output layer that decides or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer are capable of approximating any continuous function. Except for the input nodes, each node is a neuron that uses a nonlinear [activation function](https://en.wikipedia.org/wiki/Activation_function).

Multilayer perceptron’s are often applied to supervised learning problems. They train on a set of input-output pairs and learn to model the correlation (or dependencies) between those inputs and outputs. Training involves adjusting the parameters, or the weights and biases, of the model in order to minimize error. Backpropagation is used to make those weigh and bias adjustments relative to the error, and the error itself can be measured in a variety of ways, including by root mean squared error (RMSE). In this seminar, we focus on the problem of constructing an optimal multilayer perceptron network architecture.

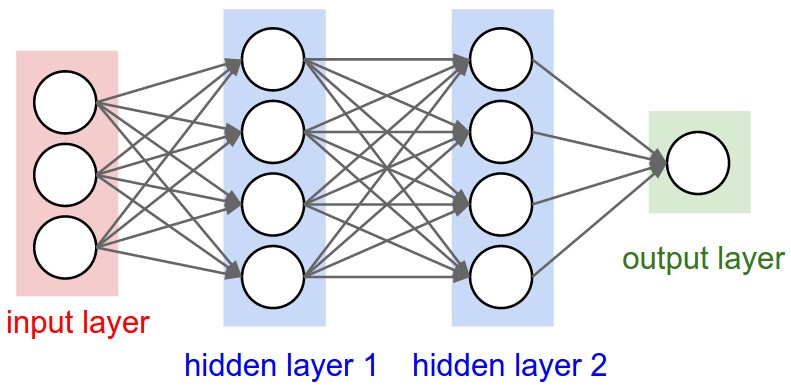


Figure 19 Architecture of MLP

1. **Input Layer**: How many features/dimensions are in your data. i.e. how many columns in each data row. Add one to this (for the bias node) and that is the number of nodes for the first input layer.
2. **Output Layer**: MLP is running in 'machine' mode or 'regression' mode, 'regression' is used in the machine learning rather than the statistical sense i.e. does my MLP return a class label or a predicted value?
3. **Hidden Layer**: In between these two input and output layer are obviously the hidden layers. Always start with a single hidden layer. So, set the initial size of the hidden layer to some number of nodes just slightly greater than the number of nodes in the input layer. Compared with having fewer nodes than the input layer, this excess capacity will help your numerical optimization routine (e.g. gradient descent) converge.

## 4.1 How does it Works

Let we have *m*input data (*x1, x2, ………. xm*), we call this *m features*. A feature is just one variable we consider as having an influence to a specific outcome.

Then we multiply each of the m features with a weight (*w1, w2, ……… wm*) and sum them all together, this is a dot product.

W.X = w1.x1 + w2.x2 + ………... wm.xm

z + bias

So here,

* With m features in input X,we need m weights to perform a dot product.
* With n hidden neurons in the hidden layer, we need n sets of weights (W1, W2, … Wn) for performing dot products
* With 1 hidden layer, we can perform 'n' dot products to get the hidden output h:(h1, h2, …, hn)
* Then it’s just like a single-layer perceptron, we use hidden output h:(h1, h2, …, hn) as input data that has 'n' features**,** perform dot product with 1 set of n weights (w1, w2, …, wn) to get your final output **y'.**

Now, feed z into an activation function ƒ(z) so that we get an output for all hidden layers. After calculated, we use them as input to calculate the final output.

i.e. y' = ƒ(z)

It is just like a single layer perceptron except that It have many more weights in process. When training neural networks on larger dataset with many more features it takes lots of memory in our computer.

Error (e) =

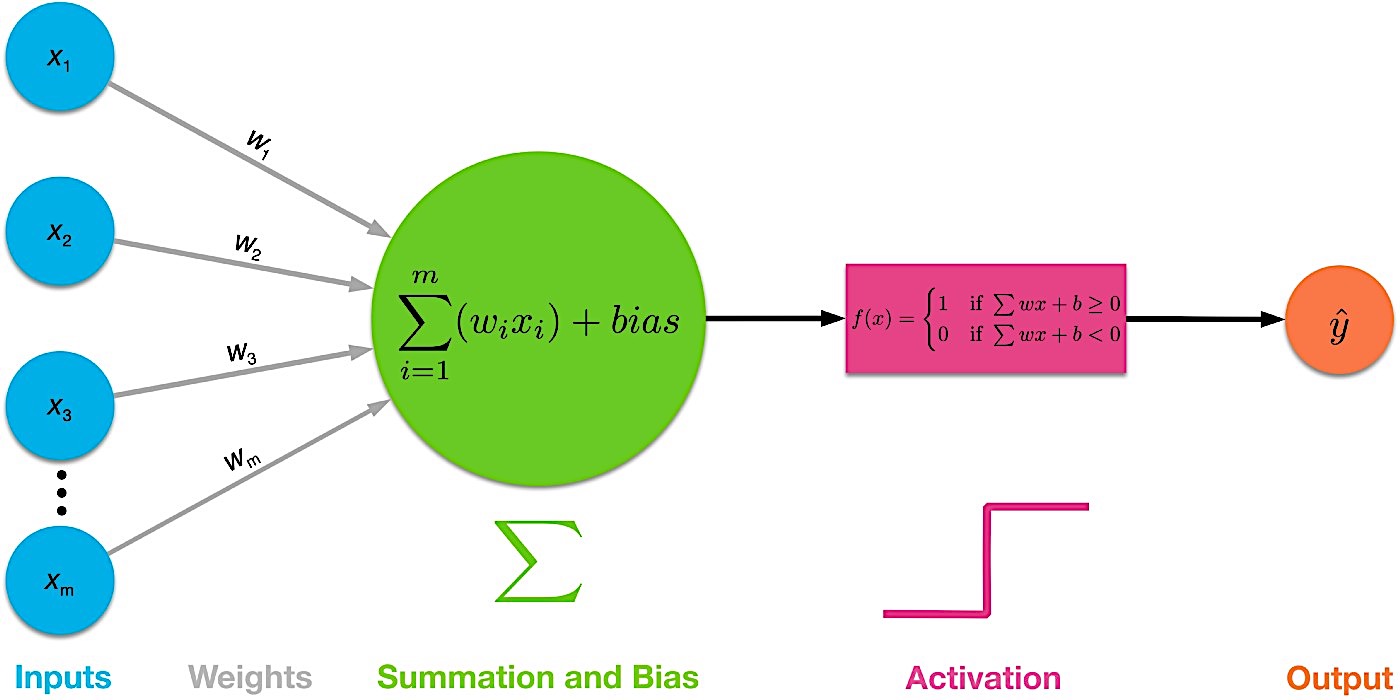


Figure 20 Working Phases of MLP

## 4.2 Evaluation Metrics

Evaluation Matrix can be defined as: Confusion Matrix, AUC-ROC Curve, Log loss, F-Beta score etc. In this project we use Confusion matrix and is classified on following terms.

### 4.2.1 Confusion Matrix:

The Confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. A confusion matrix is a technique for summarizing the performance of a classification algorithm. On this the number of correct and incorrect predictions are summarized with count values and broken down by each class. It is a table with two dimensions (“Actual” and “Predicted”), and sets of “classes” in both dimensions. It is calculated as:

1. We need a test dataset or a validation dataset with expected outcome values.
2. Make a prediction for each row in your test dataset.
3. From the expected outcomes and predictions count:

* The number of correct predictions for each class.
* The number of incorrect predictions for each class, organized by the class that was predicted.

These numbers are then organized into a table, or a matrix as follows:

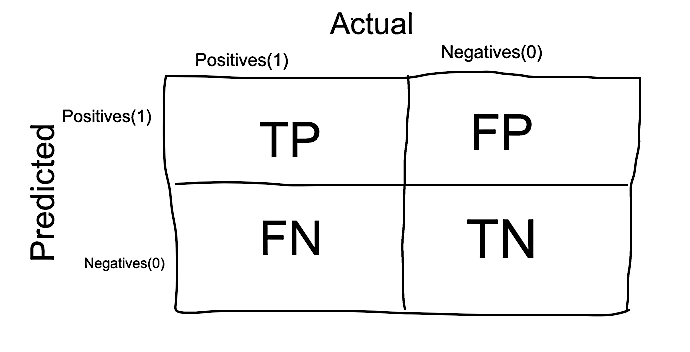


Figure 21 Confusion Matrix

1. **True Positives (TP):** True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True)
2. **True Negatives (TN):** True negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0(False).
3. **False Positives (FP):** False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one. (1)
4. **False Negatives (FN):**False negatives are the cases when the actual class of the data point was 1(True) and the predicted is 0(False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one.

For this report the total data set is partitioned into two halves, 70% of total dataset for training purpose and 30% for testing from every dataset out of 6,033 images with 200 species. Every set has completely different samples, which are selected randomly in uniformly manner from the pool of given dataset. Then system is trained using different random samples from each dataset in supervised manner. And then system is tested by testing data and accuracy is measured. Total dataset of 1,810 images tested and found the records with following outcomes.

|  |  |
| --- | --- |
| True positive = 1176 | False positive = 22 |
| False negative = 405 | True negative = 207 |

### 4.2.2 Accuracy

Accuracy in classification problems is the number of correct predictions made by the model over all kind's predictions made.

= = 76.403%

Accuracy is a good measure when the target variable classes in the data are nearly balanced. But It should not be used as a measure when the target variable classes in the data are a majority of one class.

### 4.2.3 Precision

In [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval) and binary classification, precision (also called [positive predictive value](https://en.wikipedia.org/wiki/Positive_predictive_value)) is the fraction of relevant instances among the retrieved instances. The precision is defined as:

= 98.163%

Where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

### 4.2.4 Recall

While recall (also known as [sensitivity](https://en.wikipedia.org/wiki/Sensitivity_and_specificity)) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Both precision and recall are therefore based on an understanding and measure of [relevance](https://en.wikipedia.org/wiki/Relevance). The recall is defined as:

= =74.383%

Where tp is the number of true positives and fn is the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

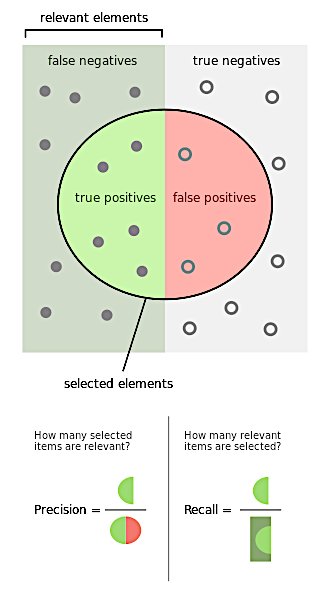


Figure 22 Precision and Recall

So, in this case, precision is "how useful the search results are", and recall is "how complete the results are" experiments here.

### 4.2.5 F-Score

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0. The F-beta score weights recall more than precision by a factor of beta. Beta value = 1.0 means recall and precision are equally important. It is calculated as:

= = 0.8462

This measure is approximately the average of the two when they are close, and is more generally the [harmonic mean](https://en.wikipedia.org/wiki/Harmonic_mean), which for the case of two numbers, coincides with the square of the [geometric mean](https://en.wikipedia.org/wiki/Geometric_mean) divided by the [arithmetic mean](https://en.wikipedia.org/wiki/Arithmetic_mean). There are several reasons that the F-score can be criticized in particular circumstances due to its bias as an evaluation metric.

In this way we determine classification report (i.e. precision, recall and f1 score) from the confusion matrix for the training sets. We also calculate the average accuracy.

# Chapter-5 EXPERIMENT RESULT AND ANALYSIS

The aim of this project work was to evaluate the Bird Species Identification Using Artificial Neural Network and Multilayer Perceptron is to classify it. There is septate dataset for 200 bird species and experiment is done 6,033 number of images over all. For the preprocessing the raw image dataset is converted into a Grayscale image dataset and resized it into 128X128 size. For this report SIFT is used for feature extraction. It is very simplest and efficient way technique to implement. It starts by detecting edges and corners in the image. On the resulted image, It tries to find interesting points that are differentiating that image from the others. It is found to be well and sufficient feature extraction technique.

Confusion matrix is used for finding accuracy and efficiency for each dataset. This experiment gives the following results:

|  |  |
| --- | --- |
| Accuracy | 76.403% |
| Precision | 98.163% |
| Recall | 74.383% |
| F-Score | 0.8462 |

# Chapter-6 CONCLUSION AND FUTURE SCOPE

## 6.1 Conclusion

This work attempt to investigate the performance of Scale Invariant Feature Extraction for feature extraction, Multi-layer perceptron for identification and Confusion matrix for evaluate the performance. The important preprocessing steps and feature extraction techniques for Bird identification is described in full detail. Bird Identification is a difficult problem, not only because of the huge number of birds, but also because of the various image of birds for single species. Recognition accuracy of the system greatly depends upon the nature of the data to be recognized.

## 6.2 Future scope

There are several ways in which I think it would be interesting to extend this work. The first would be to perform additional experiments in the current simulation environment. The second would be extend the dataset to increase accuracy. The third would be using deep learning for better experimental result. Finally, it would be more affective and interesting if we collect dataset our self.

Moreover, it can be extended it for calculating the bird migration rate and behavior of different species of bird. It would be more beneficial for preventing the bird species that are on the list of endangered species. It might play a vital role to predict about bird's species in endangered list.

# Reference

|  |  |
| --- | --- |
| [1] | P. Welinder, "Caltech-UCSD Birds 200," Caltech-UCSD Birds 200, October 2008. [Online]. Available: http://www.vision.caltech.edu/visipedia/CUB-200.html. |
| [2] | S. Haykin, Multilayer Preceptron, India, Patparjung Delhi: Person Education, 2005. |
| [3] | S. Manel, "Comparing discriminant analysis, neural networks and logistic regression for predicting species distributions," [Online]. Available: https://www.researchgate.net/publication/223753841\_Comparing\_discriminant\_analysis\_neural\_networks\_and\_logistic\_regression\_for\_predicting\_species\_distributions\_A\_case\_study\_with\_a\_Himalayan\_river\_bird. |
| [4] | A. Harma, "Automatic Identification Of Birds Species Based On Sinusoidal Modeling Of Syllabus," Helsinki University of Technology, [Online]. Available: https://www.researchgate.net/profile/Aki\_Haermae2/publication/4015246\_Automatic\_identification\_of\_bird\_species\_based\_on\_sinusoidal\_modeling\_of\_syllables/links/0046352a756329a778000000/Automatic-identification-of-bird-species-based-on-sinusoidal-modeling-o. |
| [5] | J. F. A. L. K. Andr eia Marini, "Bird Species Classification Based on Color Features," Department of Electrical EngineeringFederal University of Paran a (UFPR), 23 7 2018. [Online]. Available: https://www.researchgate.net/publication/261857309\_Bird\_Species\_Classification\_Based\_on\_Color\_Features. |
| [6] | S. Fagerlund, "Bird Species Recognition Using Support Vector Machines," 22 5 2007. [Online]. Available: https://asp-eurasipjournals.springeropen.com/articles/10.1155/2007/38637. |
| [7] | "Histogram of Oriented Gradients (HOG) Descriptor," Developer Reference for Intel, 29 May 2018. [Online]. Available: https://software.intel.com/en-us/ipp-dev-reference-histogram-of-oriented-gradients-hog-descriptor. |
| [8] | H. Z. a. N. E. M. Esfahaniana, "Using local binary patterns as features for classification," The Journal of the Acoustical Society of America, 14 June 2013. [Online]. Available: https://asa.scitation.org/doi/full/10.1121/1.4811162. |
| [9] | J. O. R. S.-C. H. A.-G. Wilson Castro, "Multylayer Preceptron Architecture," *Multilayer perceptron architecture optimization using parallel computing techniques,* 13 December 2017. |
| [10] | M. Sunasra, "Performance Metrics for Classification problems in Machine Learning," 11 11 2017. [Online]. Available: https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b. |
| [11] | N. Kang, "Multi-Layer Neural Networks with Sigmoid Function," Product Marketing, 27 6 2017. [Online]. Available: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f. |

# Appendix:

## A: Data preprocessing

import os  
import sys  
import traceback  
import numpy as np  
import time  
from imutils import get\_filename\_and\_class, normalize\_array, pil2array, imread, imshow, show\_samples  
import cv2 as cv2  
  
def create\_dataset(data\_path, test\_ratio=0.2, hot\_labels=True):  
 files, id2label, label2id = get\_filename\_and\_class(data\_path=data\_path)  
 np.random.shuffle(files)  
 num\_test = int(test\_ratio \* len(files))  
  
 np.random.shuffle(files)  
 train\_files = files[num\_test:]  
 test\_files = files[:num\_test]  
  
 train\_data = []  
 train\_labels = []  
  
 test\_data = []  
 test\_labels = []  
  
 num\_classes = len(id2label)  
  
 for f in train\_files:  
 try:  
 image = normalize\_array(pil2array(image=imread(f[0])))  
 image = np.reshape(image, (image.shape[0] \* image.shape[1]))  
  
 train\_data.append(image)  
 if hot\_labels:  
 y = np.zeros(shape=num\_classes, dtype=np.float32)  
 y[int(f[1])] = 1.0  
 train\_labels.append(y)  
 else:  
 train\_labels.append(int(f[1]))  
 except Exception as \_:  
 traceback.print\_exc(file=sys.stdout)  
 continue  
  
 for f in test\_files:  
 try:  
 image = normalize\_array(pil2array(image=imread(f[0])))  
 image = np.reshape(image, (image.shape[0] \* image.shape[1]))  
 test\_data.append(image)  
 if hot\_labels:  
 y = np.zeros(shape=num\_classes, dtype=np.float32)  
 y[int(f[1])] = 1.0  
 test\_labels.append(y)  
 else:  
 test\_labels.append(int(f[1]))  
 except Exception as \_:  
 traceback.print\_exc(file=sys.stdout)  
 continue  
  
 train\_data = np.array(train\_data)  
 train\_labels = np.array(train\_labels)  
 test\_data = np.array(test\_data)  
 test\_labels = np.array(test\_labels)  
  
 if len(train\_data) == 0:  
 train\_data = None  
 train\_labels = None  
 if len(test\_data) == 0:  
 test\_data = None  
 test\_labels = None  
 return train\_data, train\_labels, test\_data, test\_labels, id2label  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 data\_dir = "../caltechdataset/"  
 files = get\_filename\_and\_class(data\_path=data\_dir)  
 show\_samples(data\_path=data\_dir,row=10,col=20)  
 print("files")  
 for f in files[0]:  
 print(f[0], f[1])  
 img = imread(f[0])  
  
 imshow(img)  
 cv2.waitKey(100)  
  
 # train\_data, train\_labels, test\_data, test\_labels, label\_map = create\_dataset(data\_path=data\_dir)  
 # print("Classes: {}={}".format(len(label\_map), label\_map))  
 # print("Train samples: {}, Test samples: {}".format(len(train\_labels), len(test\_labels)))  
 # print(train\_data[0])

import os  
import traceback  
import matplotlib.pyplot as plt  
import numpy as np  
from PIL import Image  
import sys  
import cv2  
  
def pil2array(image):  
 return np.asarray(image)  
  
def array2pil(image):  
 return Image.fromarray(np.uint8(image)).convert('RGB')  
  
def imread(filename, cv=True):  
 if cv:  
 image = cv2.imread(filename)  
 else:  
 image = Image.open(filename)  
 return image  
  
def im2bw(image):  
 return image.convert('1')  
  
def rgb2gray(image):  
 return image.convert('L')  
  
def imresize(image, size):  
 return image.resize(size, Image.ANTIALIAS)  
  
  
def normalize\_array(image):  
 return image / 255.0  
  
def imshow\_array(image):  
 plt.imshow(image, cmap='gray')  
 plt.show()  
  
def imshow(image, winname="Image", cv=True):  
 if cv:  
 cv2.imshow(winname, image)  
 else:  
 plt.imshow(np.asarray(image), cmap='gray')  
 plt.show()  
  
  
def show\_samples(data\_path, row=3, col=10):  
 files, id2label, label2id = get\_filename\_and\_class(data\_path=data\_path)  
 np.random.shuffle(files)  
  
 v = None  
 i = 0  
 for r in range(row):  
 h = None  
 for c in range(col):  
 image = rgb2gray(imresize(image=imread(files[i][0], cv=False), size=(128, 128)))  
 image = normalize\_array(pil2array(image))  
 i += 1  
 if h is None:  
 h = image.copy()  
 h = np.hstack((h, np.zeros((128, 1))))  
 else:  
 h = np.hstack((h, image, np.zeros((128, 1))))  
  
 if v is None:  
 v = h.copy()  
 v = np.vstack((v, np.zeros((1, h.shape[1]))))  
 else:  
 v = np.vstack((v, h, np.zeros((1, h.shape[1]))))  
 imshow\_array(v)  
 return v  
  
  
def get\_filename\_and\_class(data\_path, max\_classes=0, min\_samples\_per\_class=0):  
 *"""Returns a list of filename and inferred class names.  
 Args:* ***:param*** *data\_path: A directory containing a set of subdirectories representing class names. Each subdirectory should contain PNG or JPG encoded images.* ***:param*** *min\_samples\_per\_class:* ***:param*** *max\_classes:  
 data\_path:  
 Returns:  
 A list of image file paths, relative to `data\_path` and the list of  
 subdirectories, representing class names.  
  
 """* folders = [name for name in os.listdir(data\_path) if  
 os.path.isdir(os.path.join(data\_path, name))]  
  
 if len(folders) == 0:  
 raise ValueError(data\_path + " does not contain valid sub directories.")  
 directories = []  
 for folder in folders:  
 directories.append(os.path.join(data\_path, folder))  
  
 folders = sorted(folders)  
 label2id = {}  
  
 i = 0  
 c = 0  
 total\_files = []  
 for folder in folders:  
 dir = os.path.join(data\_path, folder)  
 files = os.listdir(dir)  
 if min\_samples\_per\_class > 0 and len(files) < min\_samples\_per\_class:  
 continue  
  
 for file in files:  
 path = os.path.join(dir, file)  
 total\_files.append([path, i])  
 label2id[folder] = i  
 i += 1  
  
 if 0 < max\_classes <= c:  
 break  
 c += 1  
  
 id2label = {v: k for k, v in label2id.items()}  
 return np.array(total\_files), id2label, label2id

## B: Feature Extracting (SIFT)

import cv2  
import numpy as np  
import scipy  
from scipy.misc import imread  
import cPickle as pickle  
import random  
import os  
import matplotlib.pyplot as plt  
  
  
# Feature extractor  
def extract\_features(image\_path, vector\_size=32):  
 image = imread(image\_path, mode="RGB")  
 try:  
 # Using SIFT  
 alg = cv2.SIFT\_create()  
 # Dinding image keypoints  
 kps = alg.detect(image)  
 # Getting first 32 of them.  
 # Number of keypoints is varies depend on image size and color pallet  
 # Sorting them based on keypoint response value(bigger is better)  
 kps = sorted(kps, key=lambda x: -x.response)[:vector\_size]  
 # computing descriptors vector  
 kps, dsc = alg.compute(image, kps)  
 # Flatten all of them in one big vector - our feature vector  
 dsc = dsc.flatten()  
 # Making descriptor of same size  
 # Descriptor vector size is 64  
 needed\_size = (vector\_size \* 64)  
 if dsc.size < needed\_size:  
 # if we have less the 32 descriptors then just adding zeros at the  
 # end of our feature vector  
 dsc = np.concatenate([dsc, np.zeros(needed\_size - dsc.size)])  
 except cv2.error as e:  
 print 'Error: ', e  
 return None  
  
 return dsc  
  
  
def batch\_extractor(images\_path, pickled\_db\_path="features.pck"):  
 files = [os.path.join(images\_path, p) for p in sorted(os.listdir(images\_path))]  
  
 result = {}  
 for f in files:  
 print 'Extracting features from image %s' % f  
 name = f.split('/')[-1].lower()  
 result[name] = extract\_features(f)  
  
 # saving all our feature vectors in pickled file  
 with open(pickled\_db\_path, 'w') as fp:  
 pickle.dump(result, fp)  
  
  
  
 class Matcher(object):  
  
 def \_\_init\_\_(self, pickled\_db\_path="features.pck"):  
 with open(pickled\_db\_path) as fp:  
 self.data = pickle.load(fp)  
 self.names = []  
 self.matrix = []  
 for k, v in self.data.iteritems():  
 self.names.append(k)  
 self.matrix.append(v)  
 self.matrix = np.array(self.matrix)  
 self.names = np.array(self.names)  
  
 def cos\_cdist(self, vector):  
 # getting cosine distance between search image and images database  
 v = vector.reshape(1, -1)  
 return scipy.spatial.distance.cdist(self.matrix, v, 'cosine').reshape(-1)  
  
 def match(self, image\_path, topn=5):  
 features = extract\_features(image\_path)  
 img\_distances = self.cos\_cdist(features)  
 # getting top 5 records  
 nearest\_ids = np.argsort(img\_distances)[:topn].tolist()  
 nearest\_img\_paths = self.names[nearest\_ids].tolist()  
  
 return nearest\_img\_paths, img\_distances[nearest\_ids].tolist()  
  
  
 def show\_img(path):  
 img = imread(path, mode="RGB")  
 plt.imshow(img)  
 plt.show()  
  
 def run():  
 images\_path = 'resources/images/'  
 files = [os.path.join(images\_path, p) for p in sorted(os.listdir(images\_path))]  
 # getting 3 random images  
 sample = random.sample(files, 3)  
  
 batch\_extractor(images\_path)  
  
 ma = Matcher('features.pck')  
  
 for s in sample:  
 print 'Query image =========================================='  
 show\_img(s)  
 names, match = ma.match(s, topn=3)  
 print 'Result images ========================================'  
 for i in range(3):  
 # we got cosine distance, less cosine distance between vectors  
 # more they similar, thus we subtruct it from 1 to get match value  
 print 'Match %s' % (1 - match[i])  
 show\_img(os.path.join(images\_path, names[i]))  
  
 run()

## C: MLP(Confusion Matrix)

from \_\_future\_\_ import print\_function  
  
from time import time  
import logging  
import matplotlib.pyplot as plt  
  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import GridSearchCV  
from sklearn.datasets import fetch\_lfw\_people  
from sklearn.metrics import classification\_report  
from sklearn.metrics import confusion\_matrix  
from sklearn.decomposition import PCA  
from sklearn.svm import SVC  
  
  
print(\_\_doc\_\_)  
  
# Display progress logs on stdout  
logging.basicConfig(level=logging.INFO, format='%(asctime)s %(message)s')  
  
  
# #############################################################################  
# Download the data, if not already on disk and load it as numpy arrays  
  
lfw\_bird = fetch\_lfw\_bird(min\_faces\_per\_person=70, resize=0.4)  
  
# introspect the images arrays to find the shapes (for plotting)  
n\_samples, h, w = lfw\_bird.images.shape  
  
# for machine learning we use the 2 data directly (as relative pixel  
# positions info is ignored by this model)  
X = lfw\_bird.data  
n\_features = X.shape[1]  
  
# the label to predict is the id of the person  
y = lfw\_bird.target  
target\_names = lfw\_bird.target\_names  
n\_classes = target\_names.shape[0]  
  
print("Total dataset size:")  
print("n\_samples: %d" % n\_samples)  
print("n\_features: %d" % n\_features)  
print("n\_classes: %d" % n\_classes)  
  
  
# #############################################################################  
# Split into a training set and a test set using a stratified k fold  
  
# split into a training and testing set  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.30, random\_state=42)  
  
  
# #############################################################################  
# Compute a PCA (eigenfaces) on the face dataset (treated as unlabeled  
# dataset): unsupervised feature extraction / dimensionality reduction  
n\_components = 150  
  
print("Extracting the top %d eigenfaces from %d faces"  
 % (n\_components, X\_train.shape[0]))  
t0 = time()  
pca = PCA(n\_components=n\_components, svd\_solver='randomized',  
 whiten=True).fit(X\_train)  
print("done in %0.3fs" % (time() - t0))  
  
eigenfaces = pca.components\_.reshape((n\_components, h, w))  
  
print("Projecting the input data on the eigenfaces orthonormal basis")  
t0 = time()  
X\_train\_pca = pca.transform(X\_train)  
X\_test\_pca = pca.transform(X\_test)  
print("done in %0.3fs" % (time() - t0))  
  
  
# #############################################################################  
# Train a SVM classification model  
  
print("Fitting the classifier to the training set")  
t0 = time()  
param\_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],  
 'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], }  
clf = GridSearchCV(SVC(kernel='rbf', class\_weight='balanced'), param\_grid)  
clf = clf.fit(X\_train\_pca, y\_train)  
print("done in %0.3fs" % (time() - t0))  
print("Best estimator found by grid search:")  
print(clf.best\_estimator\_)  
  
  
# #############################################################################  
# Quantitative evaluation of the model quality on the test set  
  
print("Predicting bird's names on the test set")  
t0 = time()  
y\_pred = clf.predict(X\_test\_pca)  
print("done in %0.3fs" % (time() - t0))  
  
print(classification\_report(y\_test, y\_pred, target\_names=target\_names))  
print(confusion\_matrix(y\_test, y\_pred, labels=range(n\_classes)))  
  
  
# #############################################################################  
# Qualitative evaluation of the predictions using matplotlib  
  
def plot\_gallery(images, titles, h, w, n\_row=3, n\_col=4):  
 *"""Helper function to plot a gallery of portraits"""* plt.figure(figsize=(1.8 \* n\_col, 2.4 \* n\_row))  
 plt.subplots\_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)  
 for i in range(n\_row \* n\_col):  
 plt.subplot(n\_row, n\_col, i + 1)  
 plt.imshow(images[i].reshape((h, w)), cmap=plt.cm.gray)  
 plt.title(titles[i], size=12)  
 plt.xticks(())  
 plt.yticks(())  
  
  
# plot the result of the prediction on a portion of the test set  
  
def title(y\_pred, y\_test, target\_names, i):  
 pred\_name = target\_names[y\_pred[i]].rsplit(' ', 1)[-1]  
 true\_name = target\_names[y\_test[i]].rsplit(' ', 1)[-1]  
 return 'predicted: %s\ntrue: %s' % (pred\_name, true\_name)  
  
prediction\_titles = [title(y\_pred, y\_test, target\_names, i)  
 for i in range(y\_pred.shape[0])]  
  
plot\_gallery(X\_test, prediction\_titles, h, w)  
  
# plot the gallery of the most significative eigenfaces  
  
eigenface\_titles = ["eigenface %d" % i for i in range(eigenfaces.shape[0])]  
plot\_gallery(eigenfaces, eigenface\_titles, h, w)  
  
plt.show()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 #Seed the random number generator  
 random.seed(1)  
  
 # Create layer 1 (4 neurons, each with 3 inputs)  
 layer1 = NeuronLayer(4, 3)  
  
 # Create layer 2 (a single neuron with 4 inputs)  
 layer2 = NeuronLayer(1, 4)  
  
 # Combine the layers to create a neural network  
 neural\_network = NeuralNetwork(layer1, layer2)  
  
 print "Stage 1) Random starting synaptic weights: "  
 neural\_network.print\_weights()  
  
 # The training set. We have 7 examples, each consisting of 3 input values  
 # and 1 output value.  
 training\_set\_inputs = array([[0, 0, 1], [0, 1, 1], [1, 0, 1], [0, 1, 0], [1, 0, 0], [1, 1, 1], [0, 0, 0]])  
 training\_set\_outputs = array([[0, 1, 1, 1, 1, 0, 0]]).T  
  
 # Train the neural network using the training set.  
 # Do it 60,000 times and make small adjustments each time.  
 neural\_network.train(training\_set\_inputs, training\_set\_outputs, 60000)  
  
 print "Stage 2) New synaptic weights after training: "  
 neural\_network.print\_weights()  
  
 # Test the neural network with a new situation.  
 print "Stage 3) Considering a new situation [1, 1, 0] -> ?: "  
 hidden\_state, output = neural\_network.think(array([1, 1, 0]))  
 print output