A

PROJECT REPORT

on

Bird Species Identification From an Image

Submitted in partial fulfilment for the Award of Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

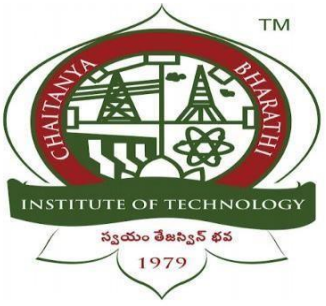
BY

JAGADEESH REDDY (160116737039)

MANIDEEP CHENNA (160116737042)

Under the guidance of

Dr K.RADHIKA
Professor
Dept. of IT, CBIT.



DEPARTMENT OF INFORMATION TECHNOLOGY
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
(Affiliated to Osmania University; Accredited by NBA (AICTE) and NAAC (UGC), ISO Certified 9001:2015), Kokapet (V), GANDIPET(M), HYDERABAD – 500 075
Website: www.cbit.ac.in

2019-2020

CERTIFICATE

This isto certify that the project work entitled "**BIRD SPECIES IDENTIFICATION FROM AN IMAGE**" submitted by **JAGADEESH REDDY (1601- 16-737-039)** and **MANIDEEP CHENNA (1601-16-737-042)** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING** in **INFORMATION TECHNOLOGY** to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(A)**, affiliated to **OSMANIA UNIVERSITY**, Hyderabad, is a record of bonafide work carried out by them under my supervision and guidance. The results embodied in this report have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

Project Guide
Dr.K.Radhika
Professor, Dept. of IT, CBIT,
Hyderabad.

Head of the Department
Dr.K.Radhika
Professor & Head, Dept. of IT,
CBIT, Hyderabad.

DECLARATION

This is to certify that the work reported in the present report titled **"BIRD SPECIES IDENTIFICATION FROM AN IMAGE"** is a record of work done by us in the Department of Information Technology, Chaitanya Bharathi Institute of Technology, Hyderabad.

No part of the project work is copied from books/journals/internet and wherever the partition is taken, the same has been duly referred in the text. The reported are based on the project work done entirely by us and not copied from any other source.

By:

Jagadeesh Reddy(160116737039.)
Manideep Chenna(160116737042.)

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Dr K.Radhika**, Professor, Dept. of IT, our project guide for her valuable guidance and constant support, along with her capable instructions and persistent encouragement.

We are grateful to our Head of the Department, **Dr K.Radhika**, for her steady support and provision of every resource required for the completion of this project.

We would like to take this opportunity to thank our principal, **Dr. P. Ravinder Reddy**, as well as the management of the institute, for having designed an excellent learning atmosphere.

Our thanks to all members of the staff and our lab assistants for providing us with the help required to carry out the groundwork of this project.

ABSTRACT

Now a days some bird species are being found rarely and if found classification of bird species prediction is difficult. Naturally, birds present in various scenarios appear in different sizes, shapes, colors, and angles from human perspective. Besides, the images present strong variations to identify the bird species more than audio classification. Also, human ability to recognize the birds through the images is more understandable. So this method uses the Caltech-UCSD Birds 200 [CUB-200-2011] dataset for training as well as testing purpose. By using deep convolutional neural network (DCNN) algorithm animage converted into grey scale format to generate autograph by using tensor flow, where the multiple nodes of comparison are generated.

By establishing the database of standard images features for bird species and using the algorithm of similarity comparison, this system is proved to achieve good results in practice

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 OVERVIEW

Identification of bird species is a challenging task often resulting in ambiguous labels. Even professional bird watchers sometimes disagree on the species given an image of a bird. It is a difficult problem that pushes the limits of the visual abilities for both humans and computers. Although different bird species share the same basic set of parts, different bird species can vary dramatically in shape and appearance. Intraclass variance is high due to variation in lighting and background and extreme variation in pose (e.g., flying birds, swimming birds, and perched birds that are partially occluded by branches).Our project aims to employ the power of machine learning to help amateur bird watchers identify bird species from the images they capture.

Many people across countries are getting into this interest of bird-watching as a hobby or extra-curricular activity. In modern world, it acts as a great stress buster and a cheap way of getting connected with nature. Another benefit of birdwatching is awareness about natureconservation by observing behavior, migratory pattern, population, and conservation status of bird species. From conservation point of view, it is important that more and more number of people turn towards bird-watching and help collect data that can be used to study birds. Sometimes, bird identification can be difficult for beginners as well.

1.2 MOTIVATION

We were always fascinated by detection of objects using machine learning that led to do this project .Identification of bird species is a challenging task often resulting in ambiguous labels. Even professional bird watchers sometimes disagree on the species given an image of a bird.

1.3 APPLICATIONS

- Helps Ornithologists easily to identify the bird.
- Model can be used for Object Detection on different Objects.

1.4 PROBLEM STATEMENT

The project aims to quantify the qualitative description of different bird species using machine learning techniques and use it as an effective tool for bird species identification from images.This project attempts to design and adopt the bird species identification system based on image feature analysis. By establishingthe database of standard images features for bird species and using the algorithm of similarity comparison, this system is proved to achieve good results inpractice.

1.5 OBJECTIVE

The main objective of our project is to deal with Image processing of bird and to identify the sign using shape and colour analysis and give the sign for classification to get the better output. The project deals with many modules which includeshape analysis, convolutional neural network. The model deals with both detection and recognition which enhances the model performance to detect the bird label.

1.6 ORGANISATION OF REPORT

Chapter 1 deals with the Introduction of the project and gives the details about the Project in an abstract view.

Chapter 2 deals with the information about the technologies used, literature survey of the papers referred and their utilization details are discussed in brief.

Chapter 3 discuss with the Methodology and design of the project.

Chapter 4 deals with the Implementation of the project.

Chapter 5 deals with the results of the work.

Chapter 6 explains the Conclusion.

2. LITERATURE SURVEY

2.1 SURVEY PAPER-1

Title: Bird Species Identification using Deep Learning

Authors: Prof. Pralhad Gavali, Ms. Prachi Abhijeet Mhetre, Ms. Neha Chandrakhant Patil, Ms. Nikita Suresh Bamane, Ms. Harshal Dipak Buva

Year of Publication: 2019

2.1.1 Problem Statement

Now a day some bird species are being found rarely and if found classification of bird species prediction is difficult. Naturally, birds present in various scenarios appear in different sizes, shapes, colors, and angles from human perspective. Besides, the images present strong variations to identify the bird species more than audio classification. Also, human ability to recognize the birds through the images is more understandable. So this method uses the Caltech-UCSD Birds 200 [CUB-200-2011] dataset for training as wellas testing purpose. By using deep convolutional neural network (DCNN) algorithm an image converted into grey scale format to generate autograph by using tensor flow, where the multiple nodes of comparison are generated. These different nodes are compared with the testing dataset and score sheet is obtained from it. After analyzing the score sheet it can predicate the required bird species by using highest score. Experimental analysis on dataset (i.e. Caltech-UCSD Birds 200 [CUB-200-2011]) shows that algorithm achieves an accuracy of bird identification between 80% and 90%.The experimental study is done with the Ubuntu 16.04 operating system using a Tensor flow library.

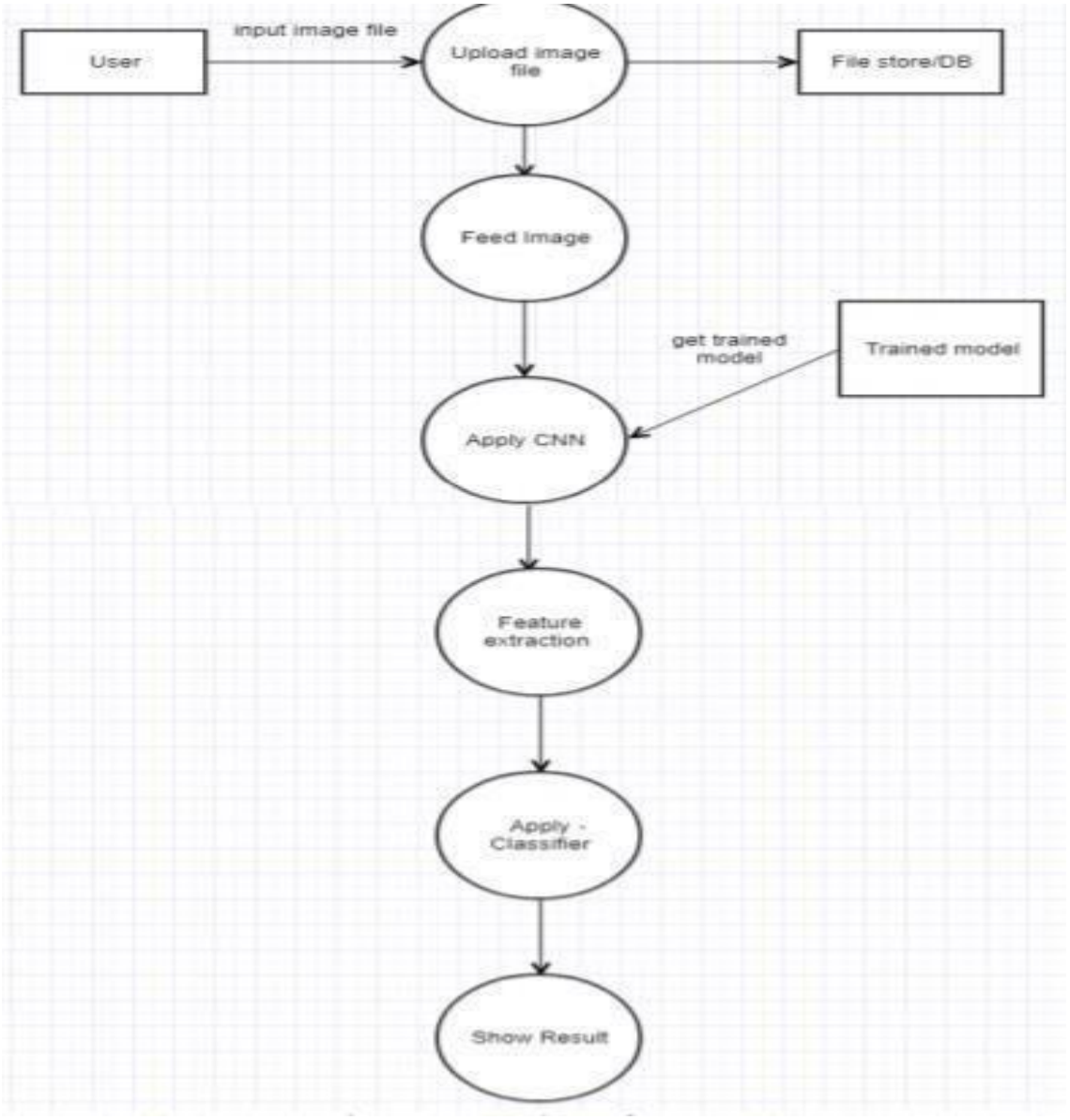


Fig.2.1 Proposed Methodology

2.1.2 Methodology

For developing the system certain methodologies have been used. They are as follows: Dataset (Caltech-UCSD Birds 200), Deep Convolutional Neural Network, Unsupervised learning algorithm, etc. Algorithm: In this experiment, unsupervised learning algorithm has been used for developing the system, because the inputted image defined is not known. Also, the data which is given to unsupervised learning algorithm are not labeled,i.e. only the input variables(X) are given with no corresponding output variables. In unsupervised learning, algorithms discover interesting structures in the data themselves. In detail, clustering is used for dividing the data into several groups. In depth, deep

learning models used to find vast number of neurons. Deep learning algorithms learn more about the image as it goes through each neural network layer. For classifying Neural Network is used. Figure 2 represents layers of neural networks for feature extraction. The neural network is a framework for many machine learning algorithms. Neural networks consist of vector of weights (W) and the bias (B). Figure No. 2: Three layers of Neural Network In deep learning, convolutional neural network (CNN) is a class of deep neural network mostly used for analyzing visual images. It consists of an input layer and output layer as well as multiple hidden layers. Every layer is made up of group of neurons and each layer is fully connected to all neurons of its previous layer. The output layer is responsible for prediction of output. The convolutional layer takes an image as input, and produces a set of feature maps as output. The input image can contain multiple channels such as color, wings, eyes, beak of birds which means that the convolutional layer perform a mapping from 3D volume to another 3D volume. 3D volumes considered are width, height, depth.

The CNN have two components:

1) Feature extraction part: features are detected when network performs a series of convolutional and pooling operation.
2) Classification part: extracted features are given to fully connected layer which acts as classifier.

CNN consists of four layers: convolutional layer, activation layer, pooling layer and fully connected. Convolutional layer allows extracting visual features from an image in small amounts. Pooling is used to reduce the number of neurons from previous convolutional layer but maintaining the important information. Activation layer passes a value through a function which compresses values into range. Fully connected layer connects a neuron from one layer to every neuron in another layer. As CNN classifies each neuron in depth, so it provides more accuracy. Image classification: image classification in machine learning is commonly done in two ways: 1) Gray scale 2) Using RGB values Normally all the data is mostly converted into gray scale. In gray scale algorithm, computer will assign values to each pixel based on how the value of the pixel is it. All the pixel values are put into an array and the computer will perform operation on that array to classify the data.

Dataset

A dataset is a collection of data. For performing action related to birds a dataset named Caltech-UCSD Birds 200 (CUB-200-2011) is used. It is an extended version of the CUB-200 dataset, with roughly double the number of images per class and also has new part location annotations for higher accuracy. The detailed information about the dataset is as follows: Number of categories: 200, Number of images: 11,788, Annotations per image: 15 Part Locations, 312 Binary Attributes, 1 Bounding Box.

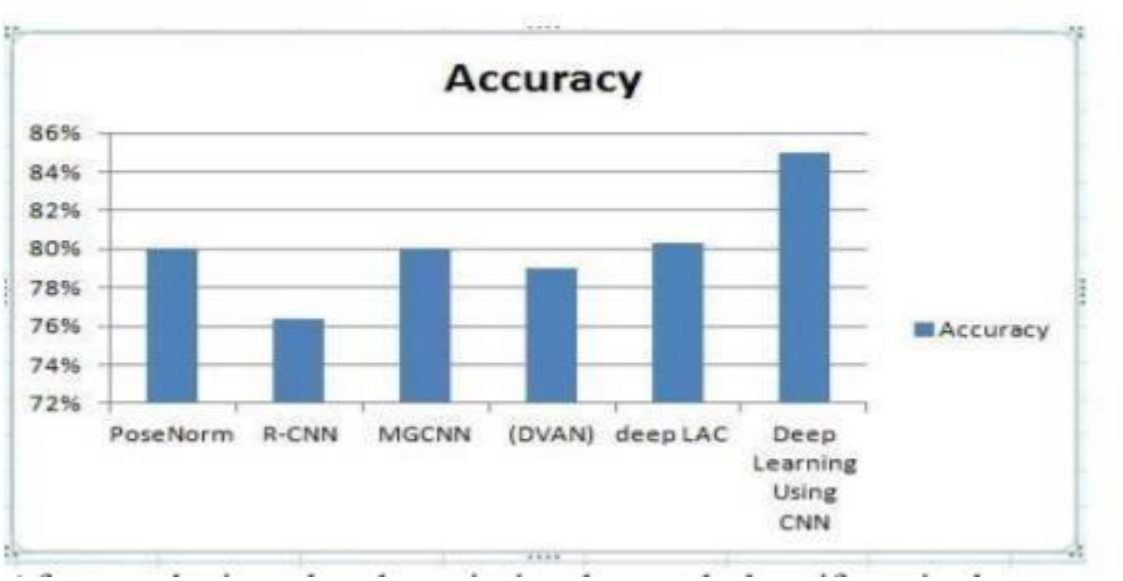


Fig.2.2: Accuracy

Results

The evaluation of the proposed approach for bird species classification by considering color features and parameters such as size, shape, etc. of the bird on the Caltech-UCSD Birds 200 (CUB-200-2011) dataset. This is an image dataset annotated with 200 bird species which includes 11,788 annotated images of birds where each image is annotated with a rough segmentation, a bounding box, and binary attribute annotations. In this the training of dataset is done by using Google-Collab, which is a platform to train dataset by uploading the images from your local machine or from the Google drive.

2.2 SURVEY PAPER – 2

Title: Bird and whale species identification using sound images.

Authors: Loris Nanni , Rafael L. Aguiar , Yandre M.G. Costa , Sheryl Brahnam ,Carlos N. Silla ,Ricky L. Brattin .

Year of Publication: 2019

2.2.1 Problem Statement

Image identification of animals is mostly centred on identifying them based on their appearance, but there are other ways images can be used to identify animals, includingby representing the sounds they make with images. In this study, the authors present a novel and effective approach for automated identification of birds and whales using some of the best texture descriptors in the computer vision literature. The visual features of sounds are built starting from the audio file and are taken from images constructed from different spectrograms and from harmonic and percussion images. These images are divided into sub-windows from which sets of texture descriptors are extracted. The experiments reported in this study using a dataset of Bird vocalisations targeted for species recognition and a dataset of right whale calls targeted for whale detection demonstrate that the fusion of different texture features enhances performance.

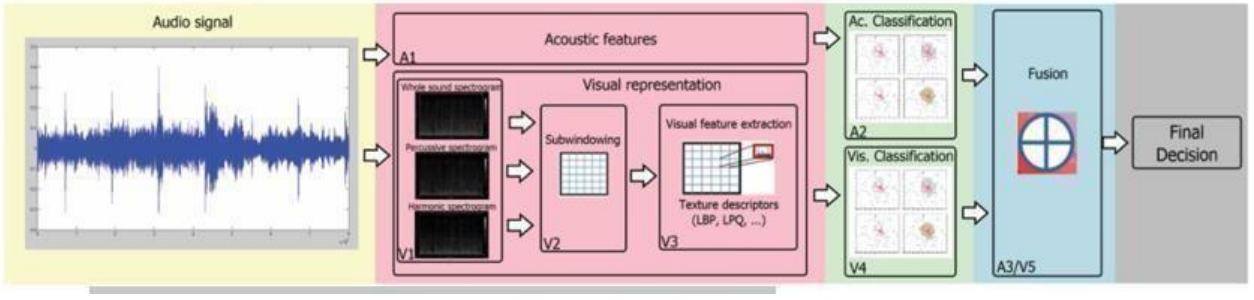


Fig.2.3: Visual and acoustic features extraction and classification steps

2.2.2 Methodology

A) Proposed Work

In Fig. 1, we present a scheme of our proposed approach. In the first step, an audio signal is represented using audio features (A1) and visual features (V1–3). In step 2 (A2 and V4) each of these features are classified. Finally, in steps A3 and V5, the results are combined for a final decision. In steps V1–3, features are extracted from visual representations of an audio signal. Each of these visual features is classified in step V4 using a support vector machine (SVM). As shown in Fig. 1, visual feature extraction is a three step process:

Step V1: The audio signal is represented by three types of audio images:

1. Spectrogram images, which are created with the lower limits of the amplitudes set to −70, −90,

−120 dBFS, respectively, and both grey-scale and colour images are produced.

2. Percussion images.

3. Harmonic images; the latter two types of images are created using the median filtering technique proposed by FitzGerald.

Step V2: Each image is divided into a set of sub-windows.

Step V3: A set of local texture descriptors (described in Section 3.3) are extracted from the sub-windows and each descriptor is classified by a separate SVM.

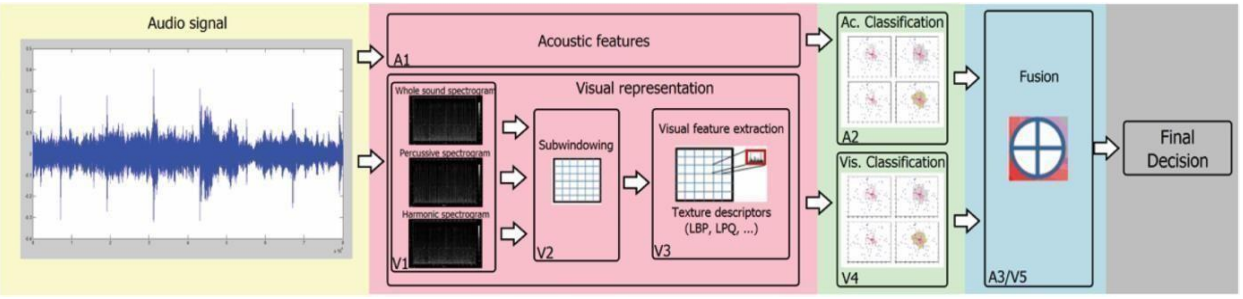


Fig. 2.2 Visual and acoustic features extraction and classification steps

B) Audio image representation

Step V1

In Step V1, different strategies were used for each dataset in order to produce more promising samples from the audio signals. For the bird and whale vocalisation tasks, it is important to extract the most relevant parts of the signal, which in the literature are called shots. For the bird dataset, samples are built from shots that are extracted manually from the original bird samples, and each sample is concatenated with itself until the size is equal to 30 s. This procedure does not impact the results either positively or negatively and was used only to standardise the size of the relevant content. Audio files in the whale dataset are divided into lengths of 2 s. Once the audio files are segmented, the spectrogram, harmonic, and percussion images in step V1 are produced.

Step V2: sub-windowing

As recommended, it is best to employ a zoning mechanism to preserve local information about the extracted features. Zoning consists of simply subdividing the whole image into smaller zones (or windows) in such a way that one can obtain the descriptors and subsequently produce classifiers specialised for the different frequency bands. it was shown that a subwindowing technique based on the Mel scale outperformed a method where features were extracted from the entire image. In

that work, 15 zones were selected for each spectrogram that varied in size as defined by the Mel scale. This produced a total of 45 zones since one spectrogram is created for each segment taken from the original signal. SVMs were trained on each of the zones, and all 45 results were combined by sum rule. In this work, we follow the method of sub-windowing proposed in. However, to reduce the computation time of the expensive ensemble proposed in this study, we have simply divided the spectrogram into three zones.

Step V3: texture descriptors

In step V3, sets of texture descriptors are extracted from the subwindows. The following texture descriptors are evaluated in this work:

• LBP, a multi-scale uniform LBP, where the final descriptor is obtained by concatenating the patterns at different radii R and at different sampling points P: $(R = 1, P = 8)$ and $(R = 2, P = 16)$.

• LPQ, a multi-scale LPQ, an LBP variant that is based on quantising the Fouriertransform phase in local neighbourhoods with radii 3 and 5.

• LBP-HF, a multi-scale LBP HF descriptor that is obtained from the concatenation of LBP-HF with values $(R = 1, P = 8)$ and $(R = 2, P = 16)$.

• RICLBP, a multi-scale rotation invariant co-occurrence of adjacent LBP with values $(R = 1, P = 8)$, $(R = 2, P = 8)$ and $(R = 4, P = 8)$

• Heterogeneous auto-similarities of characteristics (HASC) are applied to heterogeneous dense features maps.

C) Acoustic features

For the acoustic feature sets, we selected the following:

- SSD, which is a set of statistical measures describing the audio content taken from the moments on the Sonogram (the Sone) of each of the 24 critical bands defined according to Bark scale.
- RH, where the magnitudes of each modulation frequency bin of the 24 critical bands defined according to the Bark scale are summed up to form a histogram of 'rhythmic energy' per modulation frequency. This histogram has 60 bins reflecting the modulation frequencies between 0 and 10 Hz, and the feature set is the median of the histograms of each of the 6 s extracted segments.
- Modulation frequency variance descriptor (MVD), a descriptor that measures variations over the critical frequency bands for each modulation frequency. The MVD descriptor for the audio file is the mean of the MVDs taken from the 6 s segments and is a 420-dimensional vector.

• Temporal SSD, a descriptor that incorporates temporal information from the SSD, such as timbre variations and changes in rhythm. Statistical measures are taken across the SSD measures extracted from segments at different time positions in an audio file.
• Temporal rhythm histograms, a descriptor that captures rhythmic changes in music over time.

D) Datasets

BIRD: this is the Bird Songs 46 dataset in, which is freely available [Available at www.din.uem.br/yandre/birds/ bird_songs_46.tar.gz] and developed as a subset of that used. All bird species with less than ten samples were removed. The Bird Songs 46 dataset has been used in [8–10] and is composed of 2814 audio samples of bird vocalisation taken from 46 different species found in the South of Brazil. The protocol used for this dataset is a stratified 10-fold cross validation strategy, which is the same protocol used in [8–10]. The Bird Songs 46 dataset is composed of bird songs only and, in some case, one can find noise related to other bird species in the background

2.3 SURVEY PAPER - 3

Title: Audio hashing for bird species classification

Authors: Anshul Thakur , Pulkit Sharma , Vinayak Abrol , Padmanabhan Rajan

Year of Publication: 2019

2.3.1 Problem Statement

We propose a supervised, convex representation based audio hashing framework for bird species classification. The proposed framework utilizes archetypal analysis, a matrix factorization technique, to obtain convex-sparse representations of a bird vocalization. These convex representations are hashed using Bloom filters with noncryptographic hash functions to obtain compact binary codes, designated as conv-codes. The conv-codes extracted from the training examples are clustered using class-specific k-medoids clustering with Jaccard coefficient as the similarity metric. A hash table is populated using the cluster centers as keys while hash values/slots are pointers to the species identification information. During testing, the hash table is searched to find the species information corresponding to a cluster center that exhibits maximum similarity with the test conv-code. Hence, the proposed framework classifies a bird vocalization in the conv- code space and requires no explicit classifier or reconstruction error calculations. Apart from that, based on min-hash and direct addressing, we also propose a variant of the proposed framework that provides faster and effective classification. The performances of both these frameworks are compared with existing bird species classification frameworks on the audio recordings of 50 different bird species.

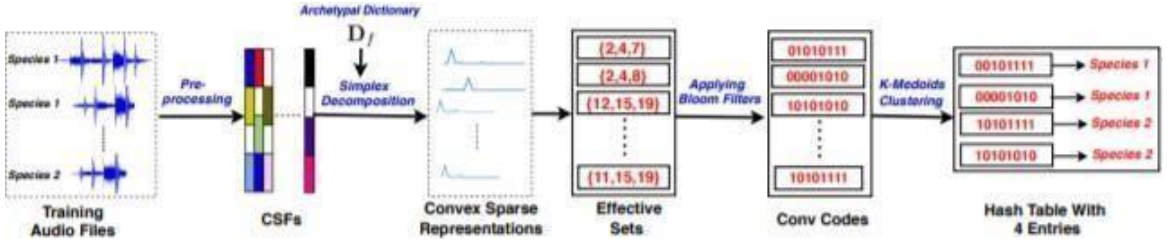


Fig.2.4: Methodology

2.3.2 Methodology

In this section, we describe the convex-sparse audio hashing in detail. First, we explain the process for obtaining archetypal dictionaries and the hash table. Then, the process to classify a given audio recording is described. 3.1. Preprocessing and dictionary learning The audio recordings are converted into compressed super-frame (CSF) representation

by concatenating W neighbouring frames of the spectrogram and projecting the concatenated vector on a random Gaussian matrix. The concatenation helps in effectively capturing the frequency-temporal modulations that give speciesspecific nature to bird vocalizations, while random projections help in compressing the concatenated vectors (Wm-dimensional to Kdimensional, where m is the number of frequency bins in a frame and K << Wm). More details about CSF representation can be found in. Only CSFs corresponding to the vocalization regions of an audio are used for learning dictionaries. A semi-supervised segmentation method proposed in is used here for segmenting bird vocalizations. In this work, archetypal analysis (AA) is employed to obtain the class- specific dictionaries from the CSF representation. The CSFs obtained from vocalizations of a species are pooled together to form a matrix X ∈ R K×l , where K is the dimensionality of the CSFs and l is the number of pooled CSFs. This matrix X is factorized as: X = DA. The dictionary D ∈ R K×d has d archetypes that model the convex hull of the data and characterize the extremal rather than the average behaviour as provided by other acoustic modelling techniques such as Gaussian mixture models. These archetypes are the convex combination of input data points such that D = XB. The archetypal dictionary, D can be obtained by solving the following optimization problem: argmin B,A bj∈∆l ,ai∈∆d kX−DAk 2 F = argmin B,A bj∈∆l ,ai∈∆d kX−XBAk 2 F , ∆l , [bj 0,kbjk1 = 1], ∆d , [ai 0, kaik1 = 1]. (1) Here ai and bj are columns of A ∈ R d×l and B ∈ R l×d , respectively. This optimization objective can be solved using the block-coordinate descent scheme. More details about the implementation of AA can be found in. The final dictionary Df is obtained by concatenating all the class-specific dictionaries: Df = [D1D2 . . . Dq], where Dq is the archetypal dictionary of the qth class. 3.2. Generating conv-codes and populating hash table A convex sparse representation (yi) is obtained by projecting a CSF (xi) on a simplex corresponding to the dictionary, Df ∈ R K×qd (qd is the number of archetypes from all classes): yi = argmin yi yi∈∆qd kxi−Dfyik 2 2 (2) where ∆qd , [yi 0, kyik1 = 1]. In this work, the active-set algorithm proposed in is used to solve equation 2. The convexity constraints on the decomposition leads to sparsity such that only few coefficients of yi are significant. The archetypes corresponding to these significant coefficients have maximum contribution in representing the CSF. The set formed by indices of the top Z coefficients having maximum magnitudes in convex sparse representation is termed as effective-set. An effectiveset is computed for each CSF extracted from training audio recordings. Ideally, a characteristic or fixed set of archetypes contribute to the possible effective-sets for vocalizations of each bird species. Hence, effective-sets obtained for

codes using Bloom filters. Bloom filters are designed to facilitate the set membership queries in an efficient manner by storing the information about the presence/absence of given elements in compact binary strings. To obtain a conv-code for a given effective set using Bloom filters, initially all bits in convcode are set to 0. Then, the non- cryptographic hash functions of Bloom filters hashes each element of an effective set to one of the locations of the conv-code, where the corresponding bit is flipped to 1, marking the presence of that element in the given effective-set. These hash functions are a sequenceof bit-level operations applied on an input element to produce a digest that corresponds to one of the location on the conv-code. In this work, conv-codes extracted from the training audio recordings are used for populating the hash table. The conv- codes of each class are clustered using K-medoids (with Jaccard coefficient as the similarity metric) to obtain T cluster centers. In K-medoids clustering, an input data point is the medoid or the cluster center. Hence, the cluster centers obtained using K- medoids are subset of the training conv-codes. These cluster centers are used as keys, pointing to the respective species or class label, in the hash table. Thus, the hash table has qT entries, where q is the number of classes. It must be noted that qT is significantly less than the number of input CSFs. Figure 1 depicts the process of creating the hash table from training input audio recordings of two bird species. 3.3. Classification In this section, we discuss the classification strategy employed for a test audio recording. Initially, a test input audio recording is processed to segment the bird vocalizations regions from the background. A segmented bird vocalization is represented by a set of CSFs, X = [x t 1x t 2 . . . x t n . . . x t s], where x t n is the nth CSF in the test vocalization. A convex-sparse representation (y t n) is obtained for CSF (x t n) using equation 2. This CSF, x t n, is represented by an effective-set obtained by choosing the indicies of the top Z coefficients having maximum magnitudes in y t n. Finally, x t n is converted into a conv-code (Ct n) using the bloom filter. During classification, this conv- code is matched with all the conv-codes used for populating the hash tables during training. The hash table entry that exhibits maximum similarity withCt n is regarded as its nearest neighbour and Ct n is assigned the species label pointed by the respective hash table entry. In this work, we have used Jaccard coefficient as a metric for this matching. The Jaccard coefficient between the test conv-code and the hash table entry compute the extent of intersection between their corresponding effective-sets. As discussed earlier, the possible effective-sets for each class consist of a particular set of archetypes and ideally, this set of archetypes is unique for each bird species. Hence,two

effective-sets belonging to same species exhibits maximum intersection and their corresponding conv-codes exhibit maximum Jaccard similarity. Each test CSF in X is classified into one of the available bird classes. A voting rule is applied on these CSF wise decisions to get the final species label of the bird vocalization represented by X .

3.4. Reducing computation using min-hashing The retrieval time required to find a best matched conv-code from the hash table is O(qT), where qT is the number of hash table entries. Although bit-manipulation operations are fast, the required computation to find the best matched conv-code can be reduced further. To decrease the retrieval time, we propose a variant of the proposed framework that utilizes min-hashing on the convex- sparse representations. To obtain the min-hash, convex-sparse is randomly permuted and the indices of Z most significant coefficients in this permuted representations are noted. The first of these indices act as min-hash for the convex-sparse representation under processing. It has been shown in the literature that the probability of two sets having same min-hash is equal to the expected Jaccard similarity between these sets. Hence, it can be inferred that two sets having high similarity are most likely to produce the same min-hash.

Based on this, we propose to use direct addressing for creating a hash table whose keys are min-hashes and values are corresponding label information. An array (a data-structure) can be seen as over-simplified example of the direct addressing. The index of an array is a key and the value corresponding to this key can be obtained by directly accessing the memory corresponding to this index. In this work, min-hashes are equivalent to indices of array and value at each index is the corresponding species label. Since min-hashes are indices of coefficients of the convex- sparse representation, the size of this array is equal to the number of archetypes in the dictionary. For testing a CSF, min-hash is obtained for its convex representation. The hash table is directly accessed at the location of this min-hash to obtain the species label. This hash table look-up requires O(1) computation only.

Dataset

The proposed framework is evaluated on a dataset1 containing audio recordings of 50 different bird species, obtained from three different sources. The recordings of 26 bird species were obtained from the Great Himalayan national park (GHNP), in north India. These recordings were used in for performance comparison. The recordings of 7 bird species were obtained from the bird audio database maintained by the Art & Science centre, UCLA. The remaining 17 bird species audio recordings were obtained from the Macaulay Library. All the recording used here are 16-bit mono, sampled at 44.1

kHz and are of durations varying from 15 seconds to 3 minutes. The information about these 50 species along with the total number of recordings and vocalizations per species is available at https://goo.gl/z6UEQa.

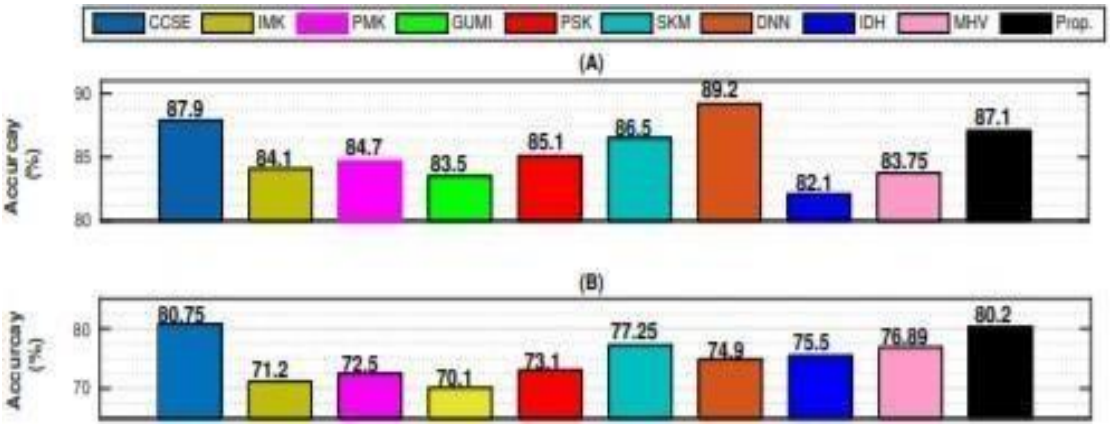


Fig.2.5: Accuracy

Results

The classification performance of the proposed framework along with other comparative methods is shown in Fig. 2(A). Following inferences can be made from this figure:

• The classification performance of the proposed framework is comparable to the DNN i.e. DNN shows a relative improvement of only 2.1%.

• Similar classification performances are observed for CCSE and the proposed framework, with CCSE showing a relative improvement of 0.9% over the proposed approach. This observation indicates that maximum intersection between effective-sets of two super-frames is same as the minimum class/dictionary specific reconstruction error used for classification, as in the CCSE approach.

• The proposed framework outperforms IMK, PMK, GUMI, PSK and SKM by a relative improvement of 3.44%, 2.76%, 4.13%, 2.3% and 0.6%, respectively.

• The incoherent dictionary learning used in IDH could not perform up-to the level of AA used in the proposed framework. This is due to the fact that the proposed framework utilizes class-specific AA dictionaries which providemore discriminative representations.

- Min-hash variant shows comparable performance to SVM based frameworks. However, on comparison with the proposed framework, min-hash variant shows a drop of 3.84% in classification accuracy. As discussed earlier, this drop in classification accuracy occurs due to the loss of information incurred while representing effective-sets by one index i.e. min-hash only. However, using min- hashing decreases the required classification time by approximately three times.

`

3.METHODOLOGY

3.1 Proposed Architecture

The below figure no.3.1 represents the actual flow of the proposed system. To develop such system a trained dataset is required to classify an image. Trained dataset consists of two parts trained result and test result. The dataset has to be retrained to achieve higher accuracy in identification using retrain.py in Google Collab. The training dataset is made using 50000 steps taking into consideration that higher the number of steps higher is its accuracy.

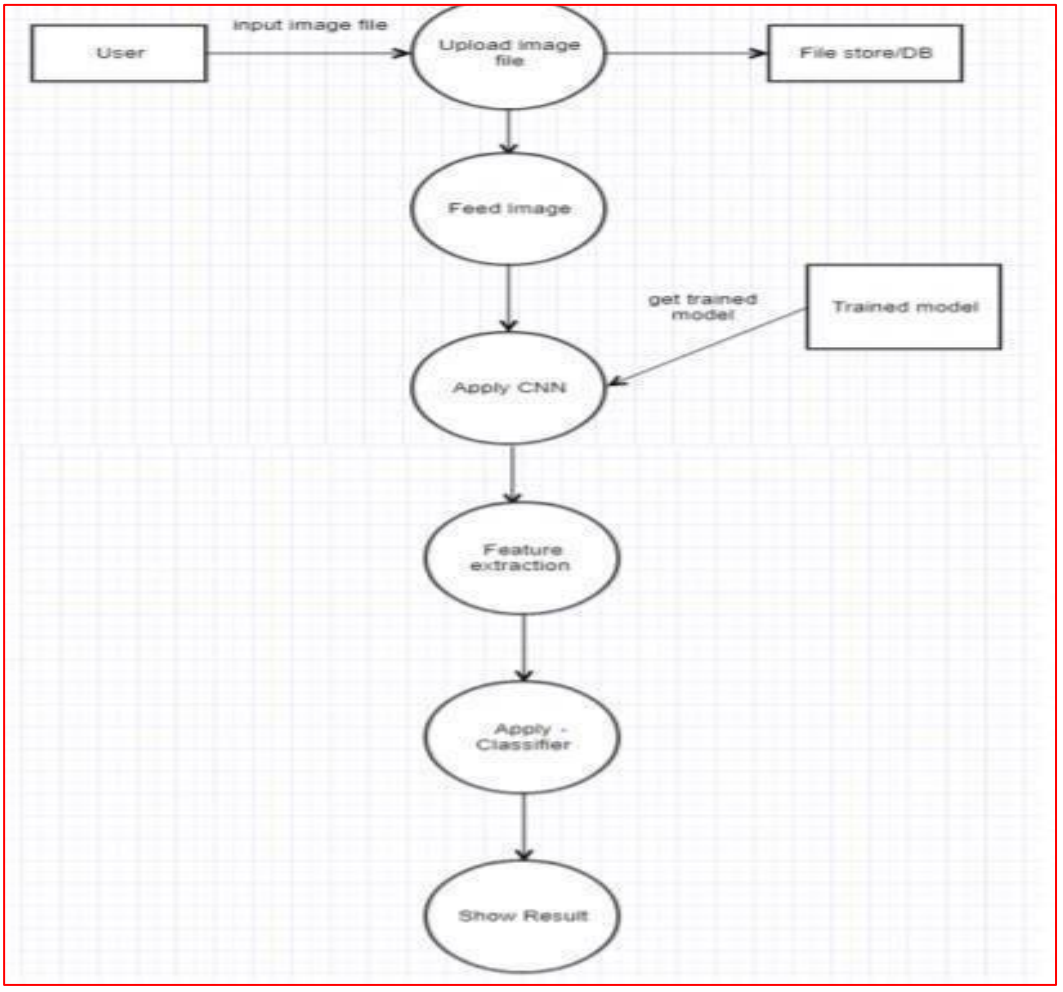


Fig:3.1 Proposed Architecture

3.2 Feed Image

Whenever a user will upload an input file on website, the image is temporarily stored in database. This input file is then feed to system and given to CNN where CNN is coupled with trained dataset.

Image classification: image classification in machine learning is commonly done in two ways:
1) Gray scale 2) Using RGB values Normally all the data is mostly converted into gray scale. In gray scale algorithm, computer will assign values to each pixel based on how the value of the pixel is it. All the pixel values are put into an array and the computer will perform operation on that array to classify the data.

3.3 Convolution Neural Network

CNN consists of four layers: convolutional layer, activation layer, pooling layer and fully connected. Convolutional layer allows extracting visual features from an image in small amounts. Pooling is used to reduce the number of neurons from previous convolutional layer but maintaining the important information. Activation layer passes a value through a function which compresses values into range. Fully connected layer connects a neuron from one layer to every neuron in another layer. As CNN classifies each neuron in depth, so it provides more accuracy.

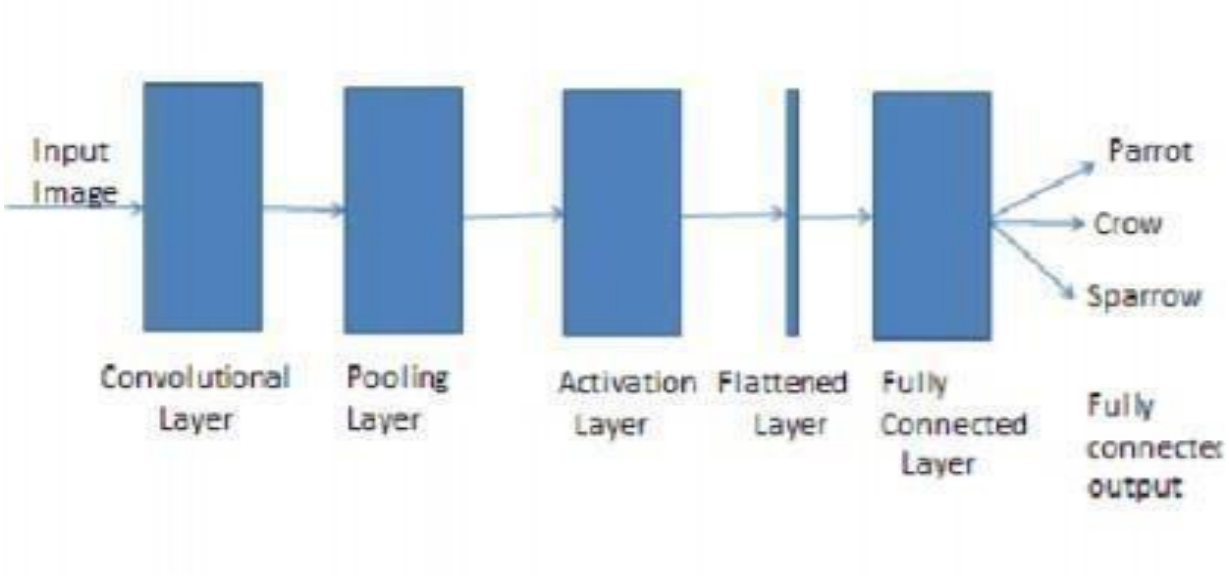


Fig 3.2 CNN Architecture

The CNN have two components:

1) Feature extraction part: Features are detected when network performs a series of convolutional and pooling operation.

2) Classification part: Extracted features are given to fully connected layer which acts as classifier.

3.3.1 Feature Extraction

Features are detected when network performs a series of convolutional and pooling operation.

3.3.2 Convolutional Layer

Convolutional neural network layer types mainly include three types, namely Convolutional layer, pooling layer and fully-connected layer Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field. Although fully connected feed forward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi- layer neural networks with many layers by using back propagation. The aim of Convolutional layer is to learn feature representations of the inputs. As shown in above, Convolutional layer is consists of several feature maps. Each neuron of the same feature map is used to extract local characteristics of different positions in the former layer, but for single neurons, its extraction is local characteristics of same positions in former different feature map. In order to obtain a new feature, the input feature maps are first convolved with a learned kernel and then the results are passed into a nonlinear activation function.

3.3.3 Pooling Layers

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer.For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer Another example is average pooling, which uses the average value from each of a cluster of neurons at the prior layer. The sampling process is equivalent to fuzzy filtering. The pooling layer has the effect of the secondary feature extraction, it can reduce the dimensions of the feature maps and increase the robustness of feature extraction. It is usually placed between two Convolutional layers. The size of feature maps in pooling layer is determined according to the moving step of kernels. The typical pooling operations are average pooling and max pooling. We can extract thehigh level characteristics of inputs by stacking several Convolutional layer and pooling layer.

3.3.4 Fully connected

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images. In general, the classifier of Convolutional neural network is one or more fully-connected layers. They takeall neurons in the previous layer and connect them to every single neuron of current layer. There is no spatial information preserved in fully-connected layers. The last fully-connected layer is followed by an output layer. For classification tasks, softmax regression is commonly used because of it generating a well-performed probability distribution of the outputs. Another commonly used method is SVM, which can be combined with CNNs to solve different classification tasks.

3.3.5 Receptive field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

3.3.6 Weights

Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is

applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a filter and represents some feature of the input. A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

3.4 Dataset

A dataset is a collection of data. For performing action related to birds a dataset named Caltech-UCSD Birds 200 (CUB-200-2011) is used. It is an extended version of the CUB-200 dataset, with roughly double the number of images per class and also has new part location annotations for higher accuracy. The detailed information about the dataset is as follows: Number of categories: 200, Number dataset is validated with an accuracy of 75% to increase the performance of system.

3.5 Library

Tensor Flow:It is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. Build and train ML models easily using intuitive high- level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging. Easily train and deploy models in the cloud, on prem, in the browser, or on device no matter what language you use. A simple and flexible architecture to take new ideas from concept to code, to state-of- the- art models, and to publication faster.

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy. If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

TensorFlow has always provided a direct path to production. Whether it’s on servers, edge devices, or the web, TensorFlow lets you train and deploy your model easily, no matter what

language or platform you use.

Use TensorFlow Extended (TFX) if you need a full production ML pipeline. For running inference on mobile and edge devices, use TensorFlow Lite. Train and deploy models in JavaScript environments using TensorFlow.js.

2) Keras

It is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility). Supports both convolutional networks and recurrent networks, as well as combinations of the two. Runs seamlessly on CPU and GPU.

a) User friendliness.

Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error

b) Modularity.

A model is understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that you can combine to create new models.

c) Easy extensibility.

New modules are simple to add (as new classes and functions), and existingmodules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

d) Work with Python.

No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

3) NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:A powerful N-dimensional array object , Sophisticated (broadcasting).

functions. Tools for integrating C/C++ and Fortran code, Useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi- dimensional container of generic data. Arbitrary data-types can be defined. Inserting or appending entries to an array is not as trivially possible as it is with Python's lists. The np.pad NumPy's np.concatenate([a1,a2]) operation does not actually link the two arrays but returns a new one, filled with the entries from both given arrays in sequence. Reshaping the dimensionality of an array with np.reshape(...) is only possible as long as the number of elements in the array does not change.

3.6 Web Technologies

Flask: Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object- relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

Nowadays, the web frameworks provide routing technique so that user can remember the URLs. It is useful to access the web page directly without navigating from the Home page. It is done through the following route() decorator, to bind the URL to a function.

Flask support various HTTP protocols for data retrieval from the specified URL, these can be defined as:-

METHOD	DESCRIPTION
GET	This is used to send the data in an without encryption of the form to the server.
HEAD	provides response body to the form
POST	Sends the form data to server. Data received by POST method is not cached by server.
PUT	Replaces current representation of target resource with URL.
DELETE	Deletes the target resource of a given URL

Fig 3.3: HTTP Protocals

HTML

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentationor functionality/behavior (JS). "Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.

CSS

Cascading Style Sheets (**CSS**) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is one of the core languages of the **open Web** and is standardized across Web browsers according to the W3C. Developed in levels, CSS1 is now obsolete, CSS2.1 is a recommendation, and CSS3, now split into smaller modules, is progressing on the standardization track.

4. IMPLEMENTATION

The CNN model to recognise bird species is constructed as follows with the help of sequential model.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Fig 4.1 Model Structure

The model accepts four parameters:

1. The image dimensions: height and width
2. Depth
3. Number of Classes in the dataset.

```
print("Loading the training data")
PATH = os.getcwd()
print(" Define data path")
data_path = PATH + '/images'
data_dir_list = os.listdir(data_path)

img_data_list=[]
count=0

for dataset in data_dir_list:
    img_list=os.listdir(data_path+'/'+dataset)
    ##print ('Loaded the images of dataset-'+'{}\n'.format(images))
    for img in img_list:
        img_path = data_path + '/'+ dataset + '/'+ img
        #224,224
        img = image.load_img(img_path, target_size=(224, 224))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)
#       x = x/255
        print('Input image shape:', x.shape)
        img_data_list.append(x)
```

Fig 4.2: Loading Images into Terminal

In the step images are moved to terminal and images are converted in feature images. Each image is converted into 224 *224 pixels size and three different matrices areformed.

Each pixel is divided by 255 so that each value in pixels will be in Integer form which helps in fast computation.

```
img_data = np.array(img_data_list)
#img_data = img_data.astype('float32')
print (img_data.shape)
img_data=np.rollaxis(img_data,1,0)
print (img_data.shape)
print (img_data.shape)
img_data=img_data[0]
```

Fig 4.3:Reshaping the Array

np.array with this function, you can also reshape your array at once like they do in the other answers with reshape (by defining the 'repeats' is more dimensions). np.rollaxis(tensor,axis,start) moves the axis specified by the axis parameter to the position before the axis that is located at start with no exceptions.

```
num_classes = 150
num_of_samples = img_data.shape[0]
labels = np.ones((num_of_samples,),dtype='int64')
labels[0:59]=0
labels[60:119]=1
labels[120:160]=2
```

Fig 4.4: Splitting and Assigning Labels

All images are converted into matrices without considering the label name whenuploading. We have to split the array in 150 different types by arrays(Lists).

Np.ones return a new array of given shape and type of int64.

Int64 is an immutable value type that represents signed integers with values that range from negative 9,223,372,036,854,775,808 (which is represented by the Int64.MinValue constant) through positive 9,223,372,036,854,775,807 (which is represented by the Int64.MaxValue constant.

```
Y = np_utils.to_categorical(labels, num_classes)

x,y = shuffle(img_data,Y, random_state=2)
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)
```

Fig 4.5: Hot Encoding and Splitting

With help of this code np_utils.to_categorical() function converts class labels into Hot- Encoding.

Hot Encoding: Hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

Training set and Testing set are dividing into 80% and 20% of the data.

```
# Block 1
x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv1')(img_input)
x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv2')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool')(x)

# Block 2
x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv1')(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv2')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool')(x)

# Block 3
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv1')(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv2')(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv3')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool')(x)

# Block 4
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv1')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv2')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv3')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool')(x)

# Block 5
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv1')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv2')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv3')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool')(x)
```

Fig 4.6: Feature Extraction Code Snippet

Feature extraction part: Features are detected when network performs a series of convolutional and pooling operation.

- A convolutional layer that extracts features from a source image. Convolution helps with blurring, sharpening, edge detection, noise reduction, or other operations that can help the machine to learn specific characteristics of an image.

- A pooling layer that reduces the image dimensionality without losing important features or patterns.

Code Snippet explaination:

In the first block “Relu” is used for activation layer, 64 different types of filters are used on the image, Feature matrix size of 3*3 is selected and slided on the image matrix and convolution function is applied.

For Pooling 2*2 size of matrix is choosed and slided on feature matrix and maximum of the matrix is choosen as output.

In the Second block “Relu” is used for activation layer, 128 different types of filters are used on the image, Feature matrix size of 3*3 is selected and slided on the image matrix and convolution function is applied.

For Pooling 2*2 size of matrix is choosed and slided on feature matrix and maximum of the matrix is choosen as output.

In the Third block “Relu” is used for activation layer, 256 different types of filters are used on the image, Feature matrix size of 3*3 is selected and slided on the image matrix and convolution function is applied.

For Pooling 2*2 size of matrix is choosed and slided on feature matrix and maximum of the matrix is choosen as output.

In the Fourth block “Relu” is used for activation layer, 512 different types of filters are used on the image, Feature matrix size of 3*3 is selected and slided on the image matrix and convolution function is applied.

For Pooling 2*2 size of matrix is choosed and slided on feature matrix and maximum of the matrix is choosen as output.

In the Fifth block “Relu” is used for activation layer, 512 different types of filters are used on the image, Feature matrix size of 3*3 is selected and slided on the image matrix and convolution function is applied.

For Pooling 2*2 size of matrix is choosed and slided on feature matrix and maximum of the matrix is choosen as output.

ReLu Function: ReLu refers to the Rectifier Linear Unit, the most commonlydeployed activation function for the outputs of the CNN neurons.

```
if include_top:
    # Classification block
    x = Flatten(name='flatten')(x)
    x = Dense(4096, activation='relu', name='fc1')(x)
    x = Dense(4096, activation='relu', name='fc2')(x)
    x = Dense(classes, activation='softmax', name='predictions')(x)
else:
    if pooling == 'avg':
        x = GlobalAveragePooling2D()(x)
    elif pooling == 'max':
        x = GlobalMaxPooling2D()(x)
```

Fig 4.7 :Classification Part

Classification part: Extracted features are given to fully connected layer which acts as classifier.

Keras.layers.flatten(dataformat=none)

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.

In other words, we put all the pixel data in one line and make connections with the finallayer. And once again.

A dense layer is just a regular layer of neurons in a neural network. Each neuron recieves input from all the neurons in the previous layer, thus densely connected.

The layer has a weight matrix **W,** a bias vector **b,** and the activations of previous layer **a.**

The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network model.

```
model.summary()
last_layer = model.get_layer('fc2').output
#x= Flatten(name='flatten')(last_layer)
out = Dense(num_classes, activation='softmax', name='output')(last_layer)
custom_vgg_model = Model(image_input, out)
custom_vgg_model.summary()
#IT will freeze the model
for layer in custom_vgg_model.layers[:-1]:
    layer.trainable = False

custom_vgg_model.layers[3].trainable

custom_vgg_model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])


t=time.time()
#   t = now()
print("Fixed Feature Extraction's Results")
hist = custom_vgg_model.fit(X_train, y_train, batch_size=32, epochs=1, verbose=1, validation_data=(X_test, y_test))
print('Training time: %s' % (t - time.time()))
(loss, accuracy) = custom_vgg_model.evaluate(X_test, y_test, batch_size=10, verbose=1)

print("[INFO] loss={:.4f}, accuracy: {:.4f}%".format(loss,accuracy * 100))
```

Fig 4.9: Final Step

Model.summary() gives the brief short description of all layers. Model.compile() x defines the loss function, the optimizer and the metrics.

It has nothing to do with the weights and you can compile a model as many times as you want without causing any problem to pretrained weights.

Model.fit() runs on the dataset no of epochs and simultaneously accuracy is printed and loss percentage is printed.

Finally atlast Overall accuracy and loss percentages are disclosed.

For Loss Categorical Cross entropy is used and for optimization rmsprop are deployed.

Loss Categorical cross entropy: Also called Softmax Loss. It is a Softmax activation plus a **Cross-Entropy loss**. If we use this loss, we will train a CNN to output a probability over the CC classes for each image. It is used for multi-class classification.

In the specific (and usual) case of Multi-Class classification the labels are one-hot, so only the positive class CpCp keeps its term in the loss. There is only one element of the Target vector tt which is not zero ti=tpti=tp. So discarding the elements of the summation which are zero due to target labels

```
from keras.models import load_model
model.save("birdss.h5")
print("Weights saved")
```

Fig 4.10: Weights of Model

Weights of the model are saved in .h file. So that it can be used other purposes. Now the model is ready now it is gonna deploy in web using Flask.

```
MODEL_PATH = 'birds.h5'

# Load your trained model
model = load_model(MODEL_PATH,compile=False)
model._make_predict_function()          # Necessary
# print('Model loaded. Start serving...')

# You can also use pretrained model from Keras
# Check https://keras.io/applications/
#from keras.applications.resnet50 import ResNet50
#model = ResNet50(weights='imagenet')
#model.save('')
print('Model loaded. Check http://127.0.0.1:5000/')
```

Fig 4.11: Model Deployed in web

Saved weights of the model are now deployed in web.Now modeled on http://127.0.0.1:5000/.

```
{% extends "base.html" %} {% block content %}

<h2>Bird Species identification</h2>

<div>
    <form id="upload-file" method="post" enctype="multipart/form-data">
        <label for="imageUpload" class="upload-label">
            Choose...
        </label>
        <input type="file" name="file" id="imageUpload" accept=".png, .jpg, .jpeg">
    </form>

    <div class="image-section" style="display:none;">
        <div class="img-preview">
            <div id="imagePreview">
            </div>
        </div>
        <div>
            <button type="button" class="btn btn-primary btn-lg " id="btn-predict">Predict!</button>
        </div>
    </div>

    <div class="loader" style="display:none;"></div>

    <h3 id="result">
        <span> </span>
    </h3>

</div>
```

Fig 4.12: Front-Code Snippet

With the help of HTML our model is ready to detect any idea of Bird image uploaded in the model it will predict the which type of bird it is.

This first HTML to be loaded is named as INDEX.html. The first page should always be named as index.html.

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Chinnu's Web</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
</head>

<body>
    <nav class="navbar navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="#">Bird Species Identification</a>
            <button class="btn btn-outline-secondary my-2 my-sm-0" type="submit">Help</button>
        </div>
    </nav>
    <div class="container">
        <div id="content" style="margin-top:2em">{% block content %}{% endblock %}</div>
    </div>
</body>

<footer>
    <script src="{{ url_for('static', filename='js/main.js') }}" type="text/javascript"></script>
</footer>

</html>
```

Fig 4.13: Base page

With the help of this code snippet it will be to load the css which is stored in other file. So the CSS file can be used for index.html page.

```
.img-preview {
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}
```

Fig 4.14: CSS code snippet

With the help of this code it is use to decorate the Html page.The width ,height position,border,box-shadow,margin-top,margin-bottom helps in adjusting the code snippet and code output looks more appealing and interacting to the user.

Background-size used for theme of the web-page.

Width & Height are used for img preview when image is uploaded by the used from uploads folder.

5. RESULTS

5.1 Model Results

```
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
Input image shape: (1, 224, 224, 3)
```

Fig 5.1: Image into Matrix

Images are converted into matrix of size 224*224 into 3 matrices of RGB colors.

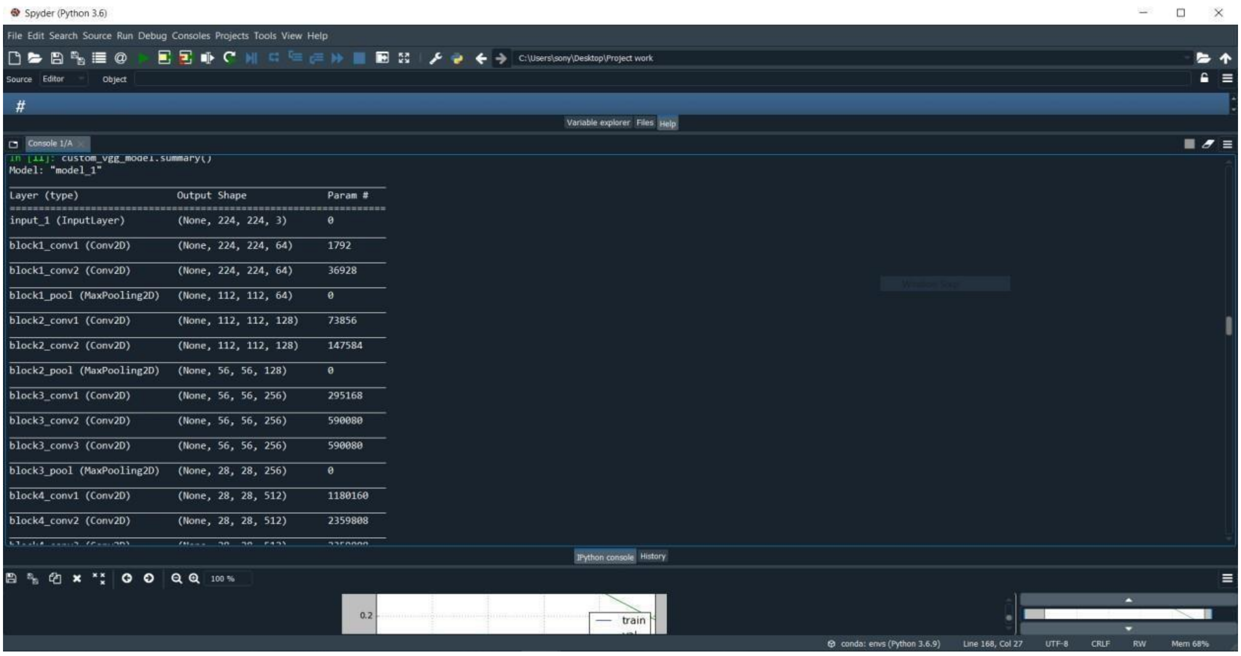


Fig 5.2: Model Summary

The Summary shows output shape of the image from every layer and Param means number of Parameters are considered from the past layer input and parameters considered.

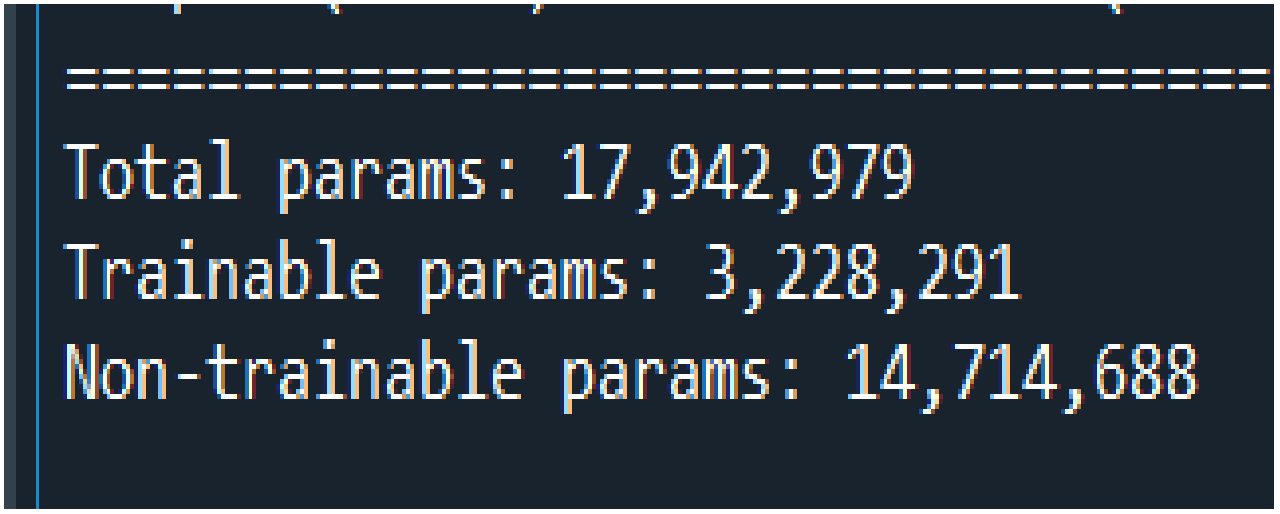


Fig 5.3: Params

Number of params from which parameters which are needed to be trained for the model to be trained for predicting.

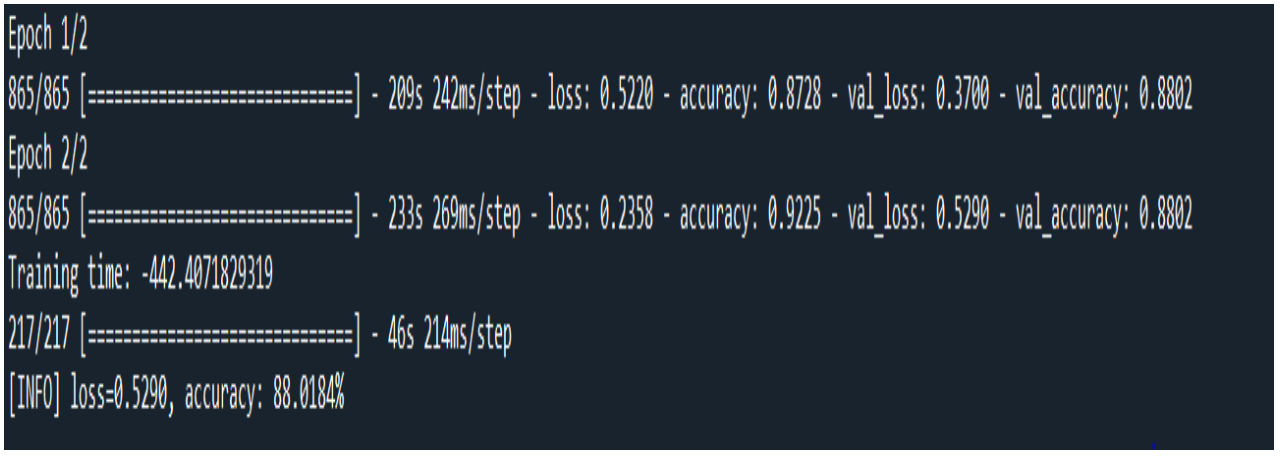


Fig 5.4: Accuracy results

By using model.fit function. We had got accuracy about 88.0184 and loss of about 52.90.

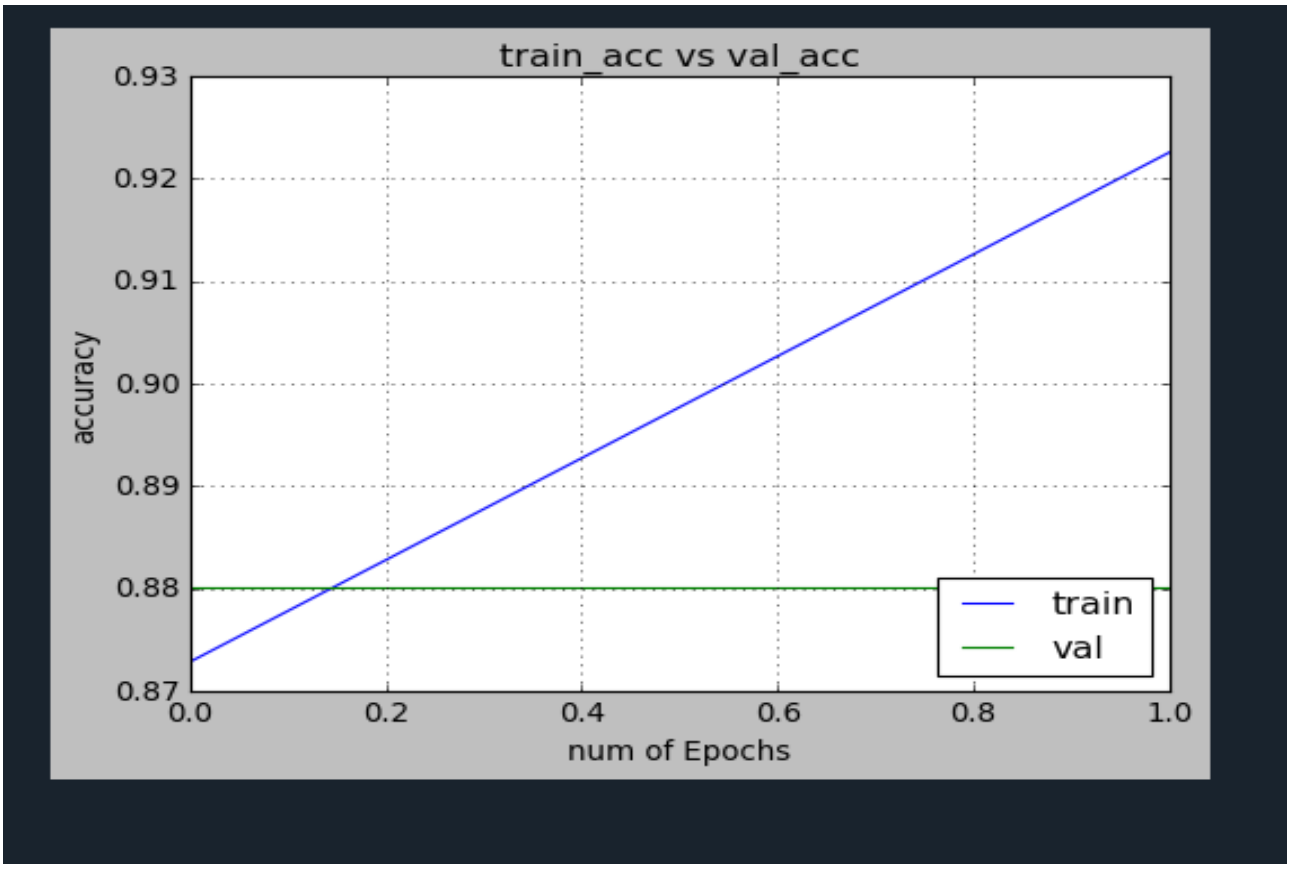


Fig 5.5: Training vs Validation Accuracy

From the above plot training accuracy is about 88% and Validation accuracy is linearly increasing this is a good sign prediction is occurring correctly.

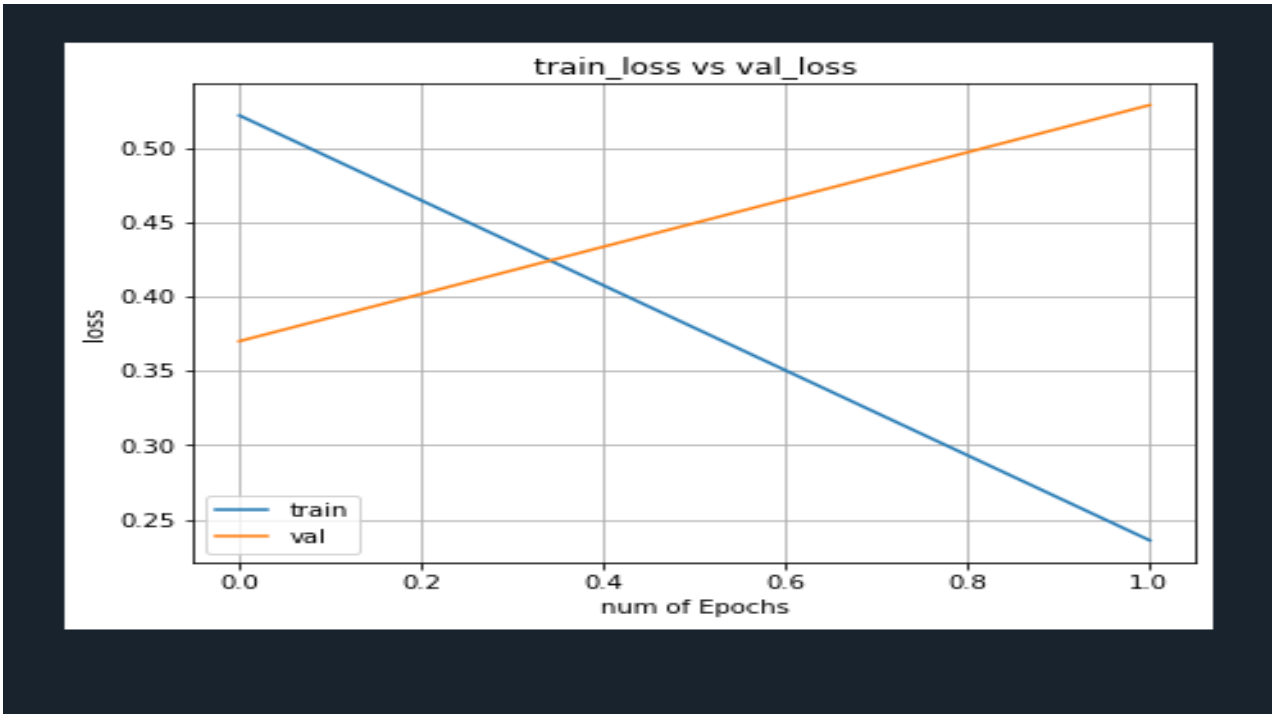


Fig 5.6: Training vs Validation loss

```
from keras.models import load_model
model.save("birdss.h5")
print("Weights saved")
```

Fig 5.7: Saving the Model

With the help of model.save() model is saved .H5 file.

Now model is ready to deploy in Application.

```
In [7]: runcell(0, 'C:/Users/sony/Desktop/Project work/app.py')
Model loaded. Check http://127.0.0.1:5000/
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Feb/2020 00:26:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2020 00:29:47] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2020 00:29:55] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2020 00:30:13] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2020 00:30:49] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2020 00:30:57] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2020 00:31:04] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2020 00:31:11] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2020 00:31:18] "POST /predict HTTP/1.1" 200 -
```

Fig 5.8 : WSGI Server

Model is deployed in 127.0.0.1 which is used to predict the image. It acts as LOG file.

5.1 Application Results

Fig 5.9:Web Application

Model is Deployed in web if we press Choose it will open the Uploads folder in server Click the image.

It gets uploaded into WSGI Server and GET method in invoked in Log file entered in log file.

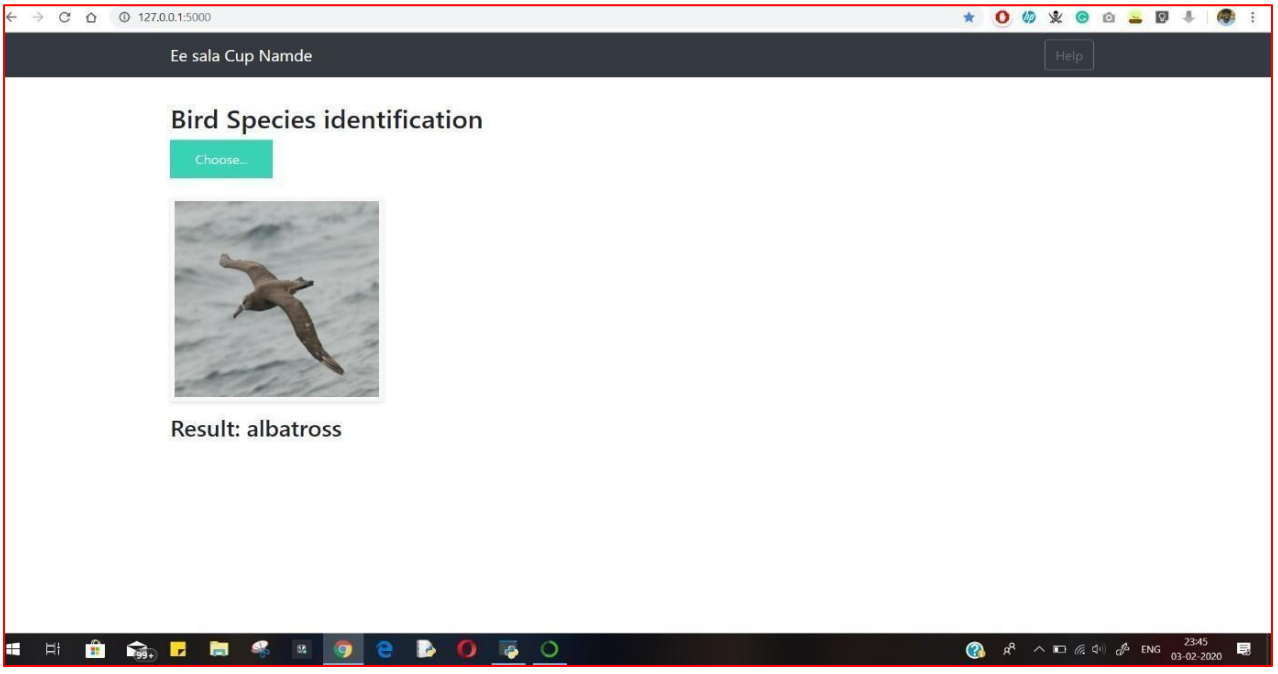


Fig 5.10: Results

When an Image is uploaded in WSGI server the results of the Image is Albatross.

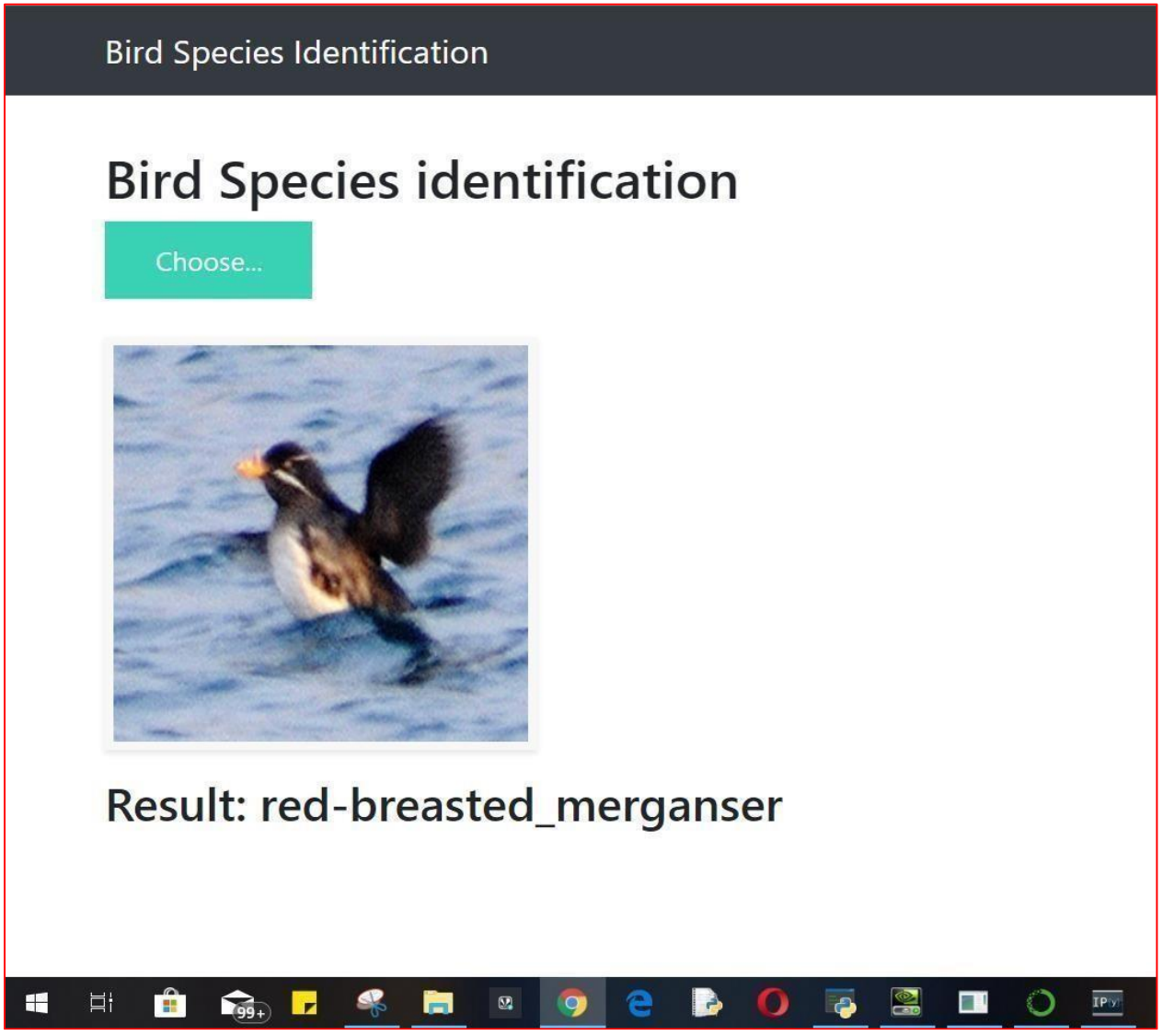


Fig 5.11:Results

When an Image is uploaded in WSGI server the results of the Image is Red- Breasted_merganser.

6. CONCLUSION & FUTURE SCOPE

The main idea behind developing the identification software is to build awareness regarding bird-watching, bird and their identification, especially birds found in India. It also caters to need of simplifying bird identification process and thus making bird-watching easier. The technology used in the experimental setup is Transfer learning in Python on a pretrained algorithm (VGG16). It uses feature extraction for image recognition. The method usedis good enough to extract features and classify images.

The main purpose of the project is to identify the bird species from an image given as input by the user. The technology used is transfer learning and Python. We used Python because it is suitable for implementing advanced algorithm and gives good numerical precision accuracy. It is also general purpose and scientific. We achieved an accuracy of

85%-88%. We believe this project extends a great deal of scope as the purpose meets. In wildlife research and monitoring, this concept can be implemented in camera traps to maintain the record of wildlife movement in specific habitat and behavior of any species.

In Future we want to create a mobile application for both IOS and Android and use Cloud for large computing.

BIBLIOGRAPHY

[1] Tóth, B.P. and Czeba, B., 2016, September. Convolutional Neural Networks for Large- Scale Bird Song Classification in Noisy Environment. In CLEF (Working Notes) (pp. 560- 568).

[2] Fagerlund, S., 2007. Bird species recognition using support vector machines. EURASIP Journal on Applied Signal Processing, 2007(1), pp.64-64.

[3] Pradelle, B., Meister, B., Baskaran, M., Springer, J. and Lethin, R., 2017, November. Polyhedral Optimization of TensorFlow Computation Graphs. In 6th Workshop on Extreme- scale Programming Tools (ESPT-2017) at The International Conference for High Performance Computing, Networking, Storage and Analysis (SC17).

[4] Cireşan, D., Meier, U. and Schmidhuber, J., 2019. Multi-column deep neural networksfor image classification. arXiv preprint arXiv:1202.2745. [5] Andr´eia Marini, Jacques Facon and Alessandro L. Koerich Postgraduate Program in Computer Science (PPGIa) Pontifical Catholic University of Paran´a (PUCPR) Curitiba PR, Brazil 80215–901 Bird Species Classification Based on Color Features

[6] Image Recognition with Deep Learning Techniques ANDREIPETRU BĂRAR, VICTOR-EMIL NEAGOE, NICU SEBE Faculty of Electronics, Telecommunications & Information Technology Polytechnic University of Bucharest.

[7] Xception: Deep Learning with Depthwise Separable Convolutions François Chollet Google, Inc.

[8] Zagoruyko, S. and Komodakis, N., 2016. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928.

[9] Inception-v4,Inception-ResNetand the Impact of Residual Connectionson Learning Christian Szegedy, Sergey Ioffe,Vincent Vanhoucke ,Alexandrr A.Alemi

[10] Stefan Kahl, Thomas Wilhelm-Stein, Hussein Hussein, Holger Klinck, Danny Kowerko, Marc Ritter, and Maximilian Eibl Large-Scale Bird Sound Classification using Convolutional Neural Networks

[11] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur Birdsnap: Large-scaleFine-grainedVisualCategorizationofBirds

[12] Bo Zhao, Jiashi Feng Xiao Wu Shuicheng Yan A Survey on Deep Learning-based Fine-grained Object Classification and Semantic Segmentation

[13] Yuning Chai Electrical Engineering Dept. ETH Zurich,Victor Lempitsky Dept. of Engineering Science University of Oxford, Andrew Zisserman Dept. of Engineering Science University of Oxford BiCoS:A BiSegmentation Method for Image Classification.

[14] Nguwi, Y., Kouzani, A. (2006). Automatic road sign recognition using neural networks. The 2006 IEEE International Joint Conference on Neural Network Proceedings,3955-3962.

[15] Garvila, D. M. (1999). Traffic sign recognition revisited.,Informatik aktuell Mustererkennung 1999,86-93.

[16] T. Wang, D. Wu, A. Coates, and A. Ng, End-to-end text recognition with Convolutionalal neural networks, in International Conference on Pattern Recognition (ICPR), 2012, pp. 33043308.

[17] Y. Boureau, J. Ponce, and Y. Le Cun, A theoretical analysis of feature pooling in visual recognition, in ICML, 2010, pp. 111118.

[18] Y. Tang, Deep learning using linear support vector machines, arXiv prprint arXiv:1306.0239, 2013.

[19] A. Ruta, Y. Li, and X. Liu, “Real-time traffic sign recognition from video by class- specific discriminative features,” Pattern Recognit., vol. 43, no. 1, pp. 416–430, 2010.

[20] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, “Traffic sign recognition— How far are we from the solution?” in Proc. IEEE Int. Joint Conf. Neural Netw., Aug. 2013, pp. 1– 8.