**BIRD SPICES CLASSIFICATION**

**ABSTRACT**

The process of identifying bird species involves a systematic approach combining observation, analysis, and documentation. It begins with field observation, where the bird's physical characteristics, such as size, color, beak shape, and plumage patterns, are noted. Behavioural traits, such as flight patterns, feeding habits, and vocalizations, are also observed. The habitat plays a crucial role, as specific bird species are often found in particular environments like forests, wetlands, or urban areas. Photographs or sketches may be captured to aid in later identification. Using field guides or bird identification apps, these observations are compared against documented species. With the vast diversity of bird species on Earth, identifying each one by name becomes a challenging task, especially for non-experts. The variations in appearance, habitat, and behavior further complicate accurate recognition. There is a lot of process to classify the bird spices. This project focuses on the classification of bird species using a Convolutional Neural Network (CNN) algorithm, a type of deep learning model particularly effective for image recognition tasks. The primary process is to identify and categorize bird species based on images provided as input. Automate the process of species identification, which traditionally requires expert knowledge and manual effort. Preprocessing techniques, such as resizing, normalization, and augmentation, are applied to prepare the images for training. The convolutional layers detect patterns and features such as edges, shapes, and textures, while pooling layers reduce dimensionality, making the computation efficient. By integrating modern Deep learning techniques, this project aims to provide a powerful tool for ornithologists and bird enthusiasts, facilitating the study and preservation of avian species.

1. **INTRODUCTION**
   1. **OVERVIEW**

The process of bird species identification traditionally involves careful observation and analysis of physical characteristics, behaviors, and habitat. This includes noting features such as size, color, beak shape, plumage patterns, flight habits, and vocalizations, often with the help of field guides or identification apps. However, given the vast diversity of bird species, accurately identifying each one, especially for non-experts, can be a complex and challenging task. This project addresses these challenges by leveraging modern deep learning techniques, specifically using a Convolutional Neural Network (CNN) algorithm, to automate bird species classification. The primary goal is to identify and categorize bird species from images, removing the need for expert knowledge and manual effort. Through preprocessing techniques such as resizing, normalization, and data augmentation, the bird images are prepared for training. The CNN model then analyzes the images by detecting key patterns, shapes, and textures, with convolutional layers identifying relevant features and pooling layers enhancing computation efficiency. This automated approach not only streamlines species identification but also serves as a valuable tool for ornithologists and bird enthusiasts, supporting the study, monitoring, and conservation of avian species.

* 1. **DOMAIN INTRODUCTION**

**DEEP LEARNING**

*Deep learning* which is a hot buzz nowadays and has firmly put down its roots in a vast multitude of industries that are investing in fields like Artificial Intelligence, Big Data and Analytics. For example, Google is using deep learning in its voice and image recognition algorithms whereas Netflix and Amazon are using it to understand the behavior of their customer. In fact, you won’t believe it, but researchers at MIT are trying to predict future using deep learning. Deep learning can be considered as a subset of machine learning. It is a field that is based on learning and improving on its own by examining computer algorithms. While machine learning uses simpler concepts, deep learning works with artificial neural networks, which are designed to imitate how humans think and learn. Neural Network is the biological neurons, which is nothing but a brain cell. Until recently, neural networks were limited by computing power and thus were limited in complexity.

Deep learning has aided image classification, language translation, speech recognition. It can be used to solve any pattern recognition problem and without human intervention. Deep learning models are capable enough to focus on the accurate features themselves by requiring a little guidance from the programmer and are very helpful in solving out the problem of dimensionality. Deep learning algorithms are used, especially when we have a huge no of inputs and outputs.

**HOW DEEP LEARNING WORKS?**

Neural Networks are layers of nodes, much like the human brain is made up of neurons. Nodes within individual layers are connected to adjacent layers. The network is said to be deeper based on the number of layers it has. A single neuron in the human brain receives thousands of signals from other neurons. The network is said to be deeper based on the number of layers it has. A single neuron in the human brain receives thousands of signals from other neurons. The final layer compiles the weighted inputs to produce an output. Deep learning systems require powerful hardware because they have a large amount of data being processed and involve several complex mathematical calculations. Even with such advanced hardware, however, training a neural network can take weeks.

Deep learning systems require large amounts of data to return accurate results; accordingly, information is fed as huge data sets. When processing the data, artificial neural networks are able to classify data with the answers received from a series of binary true or false questions involving highly complex mathematical calculations. Deep learning takes this one step ahead. Deep learning automatically finds out the features which are important for classification because of deep neural networks, whereas in case of Machine Learning we had to manually define these features.

**WHY DEEP LEARNING IS POPULAR?**

The first advantage of deep learning over machine learning is the needlessness of the so-called feature extraction. Long before deep learning was used, traditional machine learning methods were mainly used such as Decision Trees, SVM, Naïve Bayes Classifier and Logistic Regression. These algorithms are also called flat algorithms. Flat here means that these algorithms cannot normally be applied directly to the raw data (such as .csv, images, text, etc.). We need a pre-processing step called Feature Extraction. The result of Feature Extraction is a representation of the given raw data that can now be used by these classic machine learning algorithms to perform a task. Feature Extraction is usually quite complex and requires detailed knowledge of the problem domain. This pre-processing layer must be adapted, tested and refined over several iterations for optimal results. The feature extraction step is already part of the process that takes place in an artificial neural network*.* During the training process, this step is also optimized by the neural network to obtain the best possible abstract representation of the input data. This means that the models of deep learning thus require little to no manual effort to perform and optimize the feature extraction process.

**TYPES OF DEEP LEARNING**

* Feed forward neural network
* Radial basis function neural networks
* Multi-layer Perceptron
* Convolution neural network (CNN)
* Recurrent neural network
* Modular neural network

**FEED FORWARD NEURAL NETWORK**

This type of neural network is the very basic neural network where the flow control occurs from the input layer and goes towards the output layer. These kinds of networks are only having single layers or only 1 hidden layer since the data moves only in 1 direction there is no back propagation technique in this network. In the feed-forward neural network, there are not any feedback loops or connections in the network. There can be multiple hidden layers which depend on what kind of data you are dealing with. The number of hidden layers is known as the depth of the neural network. The deep neural network can learn from more functions. Input layer first provides the neural network with data and the output layer then make predictions on that data which is based on a series of functions. ReLU Function is the most commonly used activation function in the deep neural network.

**RADIAL BASIS FUNCTION NEURAL NETWORK**

This kind of neural network has generally more than 1 layer preferably two layers Radial basis networks are generally used in power restoration systems to restore the power in the shortest span of time to avoid blackouts. The popular type of feed-forward network is the radial basis function (RBF) network. It has two layers, not counting the input layer, and contrasts from a multilayer perceptron in the method that the hidden units implement computations. Each hidden unit significantly defines a specific point in input space, and its output, or activation, for a given instance based on the distance between its point and the instance, which is only a different point. The closer these two points, the better the activation.

The parameters that such a network understands are the centres and widths of the RBFs and the weights used to design the linear set of the outputs acquired from the hidden layer. An essential benefit over multilayer perceptrons is that the first group of parameters can be decided independently of the second group and make accurate classifiers. One method to decide the first group of parameters is to use clustering. The simple k-means clustering algorithm can be applied, clustering each class independently to obtain k-basis functions for each class. The second group of parameters is understood by keeping the first parameters constant. This includes learning a simple linear classifier using one of the approaches such as linear or logistic regression. If there are long fewer hidden units than training instances, this can be done fast.

The limitation of RBF networks is that they provide each attribute with a similar weight because all are considered equally in the distance computation unless attribute weight parameters are contained in the complete optimization process. Therefore, they cannot deal efficiently with inappropriate attributes, against multilayer perceptrons. Support vector machines share similar issues. Support vector machines with Gaussian kernels (i.e., “RBF kernels”) are a definite method of RBF network, in which one function is centered on each training instance, all basis functions have a similar width, and the outputs are merged linearly by calculating the maximum-margin hyperplane. This has the result that some of the RBFs have a nonzero weight the ones that define the support vectors.

**MULTI LAYER PERCEPTRON**

This type of network are having more than 3 layers and its used to classify the data which is not linear. These networks are extensively used for speech recognition and other machine learning technologies. Multilayer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.

There are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.  Every node in the multi-layer perception uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula.

**CONVOLUTION NEURAL NETWORK**

CNN is one of the variations of the multilayer Perceptron.CNN can contain more than 1 convolution layer and since it contains a convolution layer the network is very deep with fewer parameters.CNN is very effective for image recognition and identifying different image patterns. It is assumed that the reader knows the concept of neural networks.  
When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN. Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

1. **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or soft max which converts the output of each class into the probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feed forward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. After that, we back propagate into the model by calculating the derivatives. This step is called Back propagation which basically is used to minimize the loss.

**RECURRENT NEURAL NETWORK**

RNN is a type of neural network where the output of a particular neuron is fed back as an input to the same node. This method helps the network to predict the output. This kind of network is useful in maintaining a small state of memory which is very useful for developing the Chatbot and text-to-speech technologies. The **output from previous step is fed as input to the current step**. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is **Hidden state,** which remembers some information about a sequence.

RNN have a **“memory”** which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks. RNN will do the following:

* RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer.
* Hence these three layers can be joined together such that the weights and bias of all the hidden layers is the same, into a single recurrent layer.

**MODULAR NEURAL NETWORK**

A modular neural network is made up of several neural network models that are linked together via an intermediate. Modular neural networks allow for more complex management and handling of more basic neural network systems. In this case, the multiple neural networks act as modules, each solving a portion of the issue. An integrator is responsible for dividing the problem into multiple modules as well as integrating the answers of the modules to create the system's final output. Modular neural networks have been studied in various methods since the 1980s. A collection of "simple" or "weak" learners can outperform a single deep learning model, according to the idea of ensemble learning. Modular neural networks, in general, allow engineers to expand the possibilities of employing these technologies to push the limits of what neural networks can do. Each network is converted into a module that may be freely combined with modules of different sorts.

Factors leading to Modular Neural Network's development

* **Reducing model complexity:**Controlling the degrees of freedom of the system is one method to minimize training time.
* **Data fusion and prediction averaging:**Network committees may be thought of as composite systems consisting of comparable parts.
* **Combination of techniques:** As a building block, more than one method or network class can be utilized.
* **Learning several tasks at the same time:**Trained modules can be transferred between systems that are built for various tasks.
* **Robustness and incrementality:**The integrated network may be fault-tolerant and develop progressively.

**APPLICATIONS OF DEEP LEARNING**

**NATURAL LANGUAGE PROCESSING (NLP)** - Understanding the complexities associated with language whether it is syntax, semantics, tonal nuances, expressions, or even sarcasm, is one of the hardest tasks for humans to learn. Constant training since birth and exposure to different social settings help humans develop appropriate responses and a personalized form of expression to every scenario. Natural Language Processing through Deep Learning is trying to achieve the same thing by training machines to catch linguistic nuances and frame appropriate responses.

**HEALTHCARE** - Medical professionals use a CNN or Convolution Neural Network, a Deep learning method, to grade different types of cancer cells. The deep CNN models then demarcate various cellular features within the sample and detect carcinogenic elements. Helping early, accurate and speedy diagnosis of life-threatening diseases, augmented clinicians addressing the shortage of quality physicians and healthcare providers, pathology results and treatment course standardization, and understanding genetics to predict future risk of diseases and negative health episodes are some of the Deep Learning projects picking up speed in the Healthcare domain

IMAGE – LANGUAGE TRANSLATIONS - A fascination application of Deep Learning includes the Image – Language translations. With the Google Translate app, it is now possible to automatically translate photographic images with text into a real-time language of your choice. All you need to do is to hold the camera on top of the object and your phone runs a deep learning network to read the image, OCR it (i.e. convert it to text) and then translate it into a text in the preferred language. This is an extremely useful application considering that languages will gradually stop being a barrier, allowing universal human communication.

Pixel Restoration - The concept of zooming into videos beyond its actual resolution was unrealistic until Deep Learning came into play. In 2017, Google Brain researchers trained a Deep Learning network to take very low resolution images of faces and predict the person’s face through it. This method was known as the Pixel Recursive Super Resolution. It enhances the resolution of photos significantly, pinpointing prominent features in order that is just enough for personality identification.

**NEWS AGGREGATION AND FAKE NEWS DETECTION** - Deep Learning allows you to customize news depending on the readers’ persona. Neural Networks help develop classifiers that can detect fake and biased news and remove it from your feed.

**ROBOTICS** - Deep Learning is heavily used for building  robots to perform human-like tasks. Robots powered by Deep Learning use real-time updates to sense obstacles in their path and pre-plan their journey instantly. Boston Dynamics robots react to people when someone pushes them around, they can unload a dishwasher, get up when they fall, and do other tasks as well.

**SELF DRIVING CARS** - Deep Learning is the force that is bringing autonomous driving to life. A million sets of data are fed to a system to build a model, to train the machines to learn, and then test the results in a safe environment. The Uber Artificial Intelligence Labs at Pittsburg is not only working on making driverless cars humdrum but also integrating several smart features such as food delivery options with the use of driverless cars. The major concern for autonomous car developers is handling unprecedented scenarios. A regular cycle of testing and implementation typical to deep learning algorithms is ensuring safe driving with more and more exposure to millions of scenarios. Data from cameras, sensors, geo-mapping is helping create succinct and sophisticated models to navigate through traffic, identify paths, signage, pedestrian-only routes, and real-time elements like traffic volume and road blockages.

**ADVANTAGE OF DEEP LEARNING**

* No need to label the data
* Effective at Producing High-Quality Results
* There Is No Need for Feature Engineering
* Scalability
* The Cost-Effectiveness
* Advanced Analytics
* Support Parallel and Distributed algorithms

**DISADVANTAGE OF DEEP LEARNING**

* Massive Data Requirement
* High Processing Power
* Struggles With Real-Life Data and Not have strong theoretical groundwork
  1. **PROBLEM STATEMENT**

The identification of bird species is a task that traditionally requires extensive expertise and manual effort, making it time-consuming and prone to human error, especially for non-experts. With the vast number of bird species existing on Earth, distinguishing between similar-looking species or recognizing rare species can be challenging, further complicating conservation efforts, research, and educational initiatives. The existing methods for bird species identification are often inefficient and not scalable, particularly in large-scale wildlife monitoring or field studies. This project aims to address these challenges by utilizing a Convolutional Neural Network (CNN) algorithm to automate the bird species classification process. By training the CNN model on a diverse set of bird images, this project seeks to provide an accurate, fast, and scalable solution for bird identification. The goal is to reduce the reliance on expert knowledge and manual labor, improving the accessibility and efficiency of bird species classification for researchers, conservationists, and enthusiasts alike.

* 1. **OBJECTIVE**

The primary objective of this project is to develop an automated system for bird species classification using deep learning techniques, specifically a Convolutional Neural Network (CNN). The goal is to accurately identify and categorize bird species from images, minimizing the need for manual identification and expert knowledge. By leveraging CNN's ability to detect intricate patterns and features in images, the project aims to create a model that can efficiently recognize various bird species with high accuracy. Additionally, the project seeks to apply preprocessing techniques such as resizing, normalization, and data augmentation to enhance the quality and diversity of the training dataset. Ultimately, the system aims to serve as a powerful tool for ornithologists, researchers, and bird enthusiasts, enabling faster and more reliable bird species identification, supporting conservation efforts, and advancing the study of avian biodiversity.

* 1. **EXISTING SYSTEM**

This existing process offers to build on using SVMs as a classifier. Similar to the previous approach, this one uses support vector machines to classify birds after extracting their characteristics. The SoftMax layer is swapped out for SVM in the suggested techniques. Using images of birds, the performance of the employed models is evaluated. Support vector machines are utilized to categorize the appearances of birds in images after a ResNet50 architecture is developed to simulate their looks. Instead of using the SoftMax approach, this technique supports vector machines for a more reliable and accurate categorization of bird images. The suggested approaches offer improved accuracy compared to conventional methods because support vector machines are more potent classifiers than SoftMax. The performance results demonstrate that the suggested models perform better than the standard deep learning based SVM models. Using images of birds, the performance of the employed models is evaluated. Support vector machines are utilized to categorize the appearances of birds in images after a ResNet50 architecture is developed to simulate their looks. Instead of using the SoftMax approach, this technique supports vector machines for a more reliable and accurate categorization of bird images.

* 1. **DRAWBACKS OF EXISTING SYSTEM**
* It can over fit to the training data if not properly tuned or if the dataset is too small.
* Overfitting can reduce the model's ability to generalize to new, unseen bird images.
* This lack of transparency could be a disadvantage in applications where interpretability and understanding of the decision-making process are important.
* Leading to longer training times or less efficient predictions compared to other classifiers.
  1. **PROPOSED SYSTEM ARCHITECTURE**

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs).

**Various organizations define systems architecture in different ways, including:**

* An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.
* Architecture comprises the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.
* If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software
* The composite of the design architectures for products and their life-cycle processes.

Feature Extraction

Preprocessing

Admin

Bird spices Dataset

Median Filtering

Database

User

Register

Login

Upload Bird image

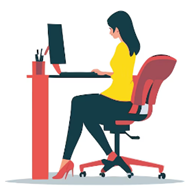
Classification

Bird spices prediction

Bird spices name & details

Testing Phase

Training Phase



1. **FEASIBILITY STUDY**

Depending on the results of the initial investigation the survey is now expanded to a more detailed feasibility study. “FEASIBILITY STUDY” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. It focuses on these major questions:

* What are the user’s demonstrable needs and how does a candidate system meet them?
* What resources are available for given candidate system?
* What are the likely impacts of the candidate system on the organization?
* Whether it is worth to solve the problem?

During feasibility analysis for this project, events and alerts are to be considered. Investigation and generating ideas about a new system does this.

**TECHNICAL FEASIBILITY**

A study of resource availability that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not.

* + Can the work for the project be done with current equipment existing software technology & available personal?
  + Can the system be upgraded if developed?
  + If new technology is needed then what can be developed?

**ECONOMICAL FEASIBILITY**

Economic justification is generally the “Bottom Line” consideration for most systems. Economic justification includes a broad range of concerns that includes cost benefit analysis. In this we weight the cost and the benefits associated with the candidate system and if it suits the basic purpose of the organization i.e. profit making, the project is making to the analysis and design phase. The financial and the economic questions during the preliminary investigation are verified to estimate the following:

• The cost to conduct a full system investigation.

• The cost of hardware and software for the class of application being considered.

• The benefits in the form of reduced cost.

• The proposed system will give the minute information, as a result the performance is improved which in turn may be expected to provide increased profits.

• This feasibility checks whether the system can be developed with events and alert monitoring does not require the manual work. This can be done economically if planned judicially, so it is economically feasible. The cost of project depends upon the number of man hours required.

**OPERATIONAL FEASIBILITY**

It is mainly related to human organizations and political aspects. The points to be considered are:

* + What changes will be brought with the system?
  + What organization structures are disturbed?
  + What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?

The system is operationally feasible as it very easy for the End users to operate it. It only needs basic information about Windows platform.

1. **PROJECT METHODOLOGY**

**CONVOLUTIONAL NEURAL NETWORK (CNN)**

A convolutional neural network (CNN or convnet) is a subset of machine learning. It is one of the various types of artificial neural networks which are used for different applications and data types. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data. There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. This makes them highly suitable for computer vision (CV) tasks and for applications where object recognition is vital, such as self-driving cars and facial recognition.

**PSEUDOCODE**

* Initialize the weights and biases of the convolutional layers randomly.
* Set the learning rate and the number of epochs.
* For each epoch do:
  + Shuffle the training data.
  + For each training example do:
* Feedforward the input through the convolutional layers and compute the output.
* ii. Apply a pooling operation to reduce the dimensionality of the output.
* iii. Flatten the output and pass it through one or more fully connected layers.
* iv. Calculate the error between the predicted output and the actual output.
* v. Backpropagate the error through the network to update the weights and biases of the convolutional layers and fully connected layers.
  + Calculate the average error for the epoch.
* Test the network on the validation set to evaluate its performance.
* Repeat steps 3 and 4 until the network achieves satisfactory performance on the validation set.
* Test the final network on the test set to evaluate its generalization performance.
  1. **MODULE DESCRIPTION**

**MODULE LIST**

* Data Acquisition
* Pre-processing
* Feature extraction
* Classification
* Bird spices prediction

**MODULES DESCRIPTION**

**Data Acquisition**

In this module the bird species first step is to collect bird images from publicly available datasets such as those found on platforms like Kaggle. Kaggle offers a variety of bird image datasets, such as the "CUB-200-2011" (Caltech-UCSD Birds 200) dataset, which contains thousands of labeled images from over 200 bird species. These datasets provide a rich source of data, offering a diverse range of bird species in different environments and conditions, making them ideal for training deep learning models. The admin collects bird species data by sourcing images and related information from various reliable datasets or field observations. This data is then organized and stored in a structured database, ensuring proper labeling and categorization for easy retrieval.

**Pre-processing**

The preprocessing process the collection of diverse and high-quality images representing various bird species. These images can be sourced from platforms like Kaggle, which offer labeled datasets such as the "CUB-200-2011" bird species dataset, or through field photography and image repositories. Once the images are collected, the first step is image resizing. This step reduces computational complexity and ensures that the model processes images of the same scale. Next, image normalization is applied, which involves scaling pixel values to a standard range, typically between 0 and 1. This process ensures that the model can learn effectively by preventing issues arising from large variations in pixel intensity. This allows the model to generalize better by making it less sensitive to specific image orientations or backgrounds. Finally, label encoding is performed to convert bird species names into numerical labels and details, enabling the model to process the data efficiently.

**Feature extraction**

The feature extraction process in image classification involves identifying and isolating important visual patterns or features within the images that can help a model distinguish between different categories. It identifying key patterns and structures in the bird images that can be used by the model to distinguish between different species. These filters scan the image in small regions and detect basic features such as edges, textures, and corners, which are essential for understanding the shape and structure of the bird. The network extracts increasingly complex features, such as shapes of wings, beaks, and feathers, as well as more abstract patterns that are unique to each bird species.

**Classification**

The classification process in bird species classification using a Convolutional Neural Network (CNN) begins after the feature extraction phase, where the network has learned to detect key visual patterns in the bird images. Once the features are extracted through convolutional and pooling layers, the next step is to pass the extracted features through **fully connected layers**. These layers are designed to combine the lower-level features into higher-level, more abstract representations, allowing the model to understand the complex relationships between various parts of the bird image, such as its shape, color patterns, and unique characteristics. n the fully connected layers, the network learns to map these features to specific output categories, which, in this case, are the different bird species. Each output corresponds to a specific bird species label, and the network assigns a probability score to each species. The model is trained using labeled bird image data, where the correct species is provided for each image. Finally the image classified by using the CNN algorithm efficiently.

**Bird spices prediction**

In this module after the classification process in bird species prediction, the model provides the predicted bird species name along with detailed information about the species. Once the image is processed through the CNN Algorithm, and the most likely bird species is identified, the output includes not only the species label but also relevant details associated with that bird. These details may include the bird's common name, scientific name, habitat, diet, and other distinguishing characteristics such as size, color, and behavior. The prediction and corresponding details offer a comprehensive view of the identified species, making it useful for applications like wildlife monitoring, educational tools, and ecological research. This combination of accurate classification and informative output enhances the user experience and the utility of the bird species prediction model.

1. **RESULTS AND DISCUSSION**

The results of this project demonstrate the effectiveness of using Convolutional Neural Networks (CNNs) for bird species classification. After training the CNN model with a diverse dataset of bird images, the system showed promising accuracy in identifying various bird species based on visual features such as plumage patterns, size, and shape. The preprocessing steps, including resizing, normalization, and data augmentation, significantly enhanced the model's ability to generalize across different image conditions, such as lighting and background variations. In terms of performance, the model exhibited robust results on test datasets, with accurate predictions for common and rare bird species alike. However, challenges remain in dealing with images of birds that have similar physical traits or those taken under poor lighting conditions, which occasionally led to misclassification. Additionally, the model's accuracy could be further improved with a larger and more diverse dataset, including various poses, environmental factors, and more detailed annotations. Overall, this project successfully automates the bird species classification process, providing a valuable tool for researchers, conservationists, and bird enthusiasts, with potential for further refinement and application in wildlife monitoring and biodiversity studies.

**4.1 SOURCE CODE**

from tkinter import \*

import os

from tkinter import filedialog

import cv2

from tkinter import messagebox

def file\_sucess():

global file\_success\_screen

file\_success\_screen = Toplevel(training\_screen)

file\_success\_screen.title("File Upload Success")

file\_success\_screen.geometry("150x100")

Label(file\_success\_screen, text="File Upload Success").pack()

Button(file\_success\_screen, text='''ok''', font=(

'Palatino Linotype', 15), height="2", width="30").pack()

global ttype

def training():

global training\_screen

global clicked

training\_screen = Toplevel(main\_screen)

training\_screen.title("Training")

# login\_screen.geometry("400x300")

training\_screen.geometry("600x450+650+150")

training\_screen.minsize(120, 1)

training\_screen.maxsize(1604, 881)

training\_screen.resizable(1, 1)

training\_screen.configure(bg='lightblue')

# login\_screen.title("New Toplevel")

Label(training\_screen, text='''Upload Image ''',

foreground="#000000", width="300", height="2", bg='lightblue', font=("Palatino Linotype", 16)).pack()

Label(training\_screen, text="").pack()

options = [

'Asian-Green-Bee-Eater', 'Brown-Headed-Barbet', 'Cattle-Egret', 'Common-Kingfisher', 'Common-Myna',

'Common-Rosefinch', 'Common-Tailorbird', 'Coppersmith-Barbet', 'Forest-Wagtail', 'Gray-Wagtail', 'Hoopoe',

'House-Crow', 'Indian-Grey-Hornbill', 'Indian-Peacock', 'Indian-Pitta', 'Indian-Roller', 'Jungle-Babbler',

'Northern-Lapwing', 'Red-Wattled-Lapwing', 'Ruddy-Shelduck', 'Rufous-Treepie', 'Sarus-Crane',

'White-Breasted-Kingfisher', 'White-Breasted-Waterhen', 'White-Wagtail'

]

# datatype of menu text

clicked = StringVar()

# initial menu text

clicked.set("Adposhel")

# Create Dropdown menu

drop = OptionMenu(training\_screen, clicked, \*options)

drop.config(width="30", bg='lightblue')

drop.pack()

ttype = clicked.get()

Button(training\_screen, text='''Upload Image''', bg='lightblue', font=(

'Palatino Linotype', 15), height="2", width="30", command=imgtraining).pack()

def imgtraining():

name1 = clicked.get()

print(name1)

import\_file\_path = filedialog.askopenfilename()

import os

s = import\_file\_path

os.path.split(s)

os.path.split(s)[1]

splname = os.path.split(s)[1]

image = cv2.imread(import\_file\_path)

# filename = 'Test.jpg'

filename = 'Dataset/' + name1 + '/' + splname

cv2.imwrite(filename, image)

print("After saving image:")

image = cv2.resize(image, (780, 540))

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cv2.imshow('Original image', image)

cv2.imshow('Gray image', gray)

# import\_file\_path = filedialog.askopenfilename()

print(import\_file\_path)

fnm = os.path.basename(import\_file\_path)

print(os.path.basename(import\_file\_path))

from PIL import Image, ImageOps

im = Image.open(import\_file\_path)

im\_invert = ImageOps.invert(im)

im\_invert.save('lena\_invert.jpg', quality=95)

im = Image.open(import\_file\_path).convert('RGB')

im\_invert = ImageOps.invert(im)

im\_invert.save('tt.png')

image2 = cv2.imread('tt.png')

image2 = cv2.resize(image2, (780, 540))

cv2.imshow("Invert", image2)

""""-----------------------------------------------"""

img = image

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow('Original image', img)

dst = cv2.medianBlur(img, 7)

cv2.imshow("Noise Removal", dst)

def fulltraining():

import Model as mm

def testing():

global testing\_screen

testing\_screen = Toplevel(main\_screen)

testing\_screen.title("Testing")

# login\_screen.geometry("400x300")

testing\_screen.geometry("600x450+650+150")

testing\_screen.minsize(120, 1)

testing\_screen.maxsize(1604, 881)

testing\_screen.resizable(1, 1)

testing\_screen.configure(bg='lightblue')

# login\_screen.title("New Toplevel")

Label(testing\_screen, text='''Upload Image''', bg='lightblue', width="300", height="2",

font=("Palatino Linotype", 16)).pack()

Label(testing\_screen, text="", bg='lightblue').pack()

Label(testing\_screen, text="", bg='lightblue').pack()

Label(testing\_screen, text="", bg='lightblue').pack()

Button(testing\_screen, text='''Upload Image''', font=(

'Palatino Linotype', 15), height="2", bg='lightblue', width="30", command=imgtest).pack()

global affect

def imgtest():

import\_file\_path = filedialog.askopenfilename()

image = cv2.imread(import\_file\_path)

print(import\_file\_path)

filename = 'Test.jpg'

cv2.imwrite(filename, image)

print("After saving image:")

# result()

# import\_file\_path = filedialog.askopenfilename()

print(import\_file\_path)

fnm = os.path.basename(import\_file\_path)

print(os.path.basename(import\_file\_path))

# file\_sucess()

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nImage : " + fnm + "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

img = cv2.imread(import\_file\_path)

if img is None:

print('no data')

img1 = cv2.imread(import\_file\_path)

print(img.shape)

img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))

original = img.copy()

neworiginal = img.copy()

img1 = cv2.resize(img1, (960, 540))

cv2.imshow('original', img1)

gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

img1S = cv2.resize(img1, (960, 540))

cv2.imshow('Original image', img1S)

grayS = cv2.resize(gray, (960, 540))

cv2.imshow('Gray image', grayS)

dst = cv2.fastNlMeansDenoisingColored(img1, None, 10, 10, 7, 21)

dst = cv2.resize(dst, (960, 540))

cv2.imshow("Noise Removal", dst)

result()

def result():

import warnings

warnings.filterwarnings('ignore')

import tensorflow as tf

classifierLoad = tf.keras.models.load\_model('model.h5')

base\_dir = 'Dataset/'

catgo = os.listdir(base\_dir)

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img('Test.jpg', target\_size=(200, 200))

x = image.img\_to\_array(test\_image)

x = x / 255.

x = np.expand\_dims(x, axis=0)

images = np.vstack([x])

result = classifierLoad.predict(images)

out = ''

pre = ''

print(result)

ind = np.argmax(result)

print(catgo[ind])

out = catgo[ind]

if out == "Asian-Green-Bee-Eater":

pre = "The Asian green bee-eater, also known as little green bee-eater, and green bee-eater in Sri Lanka, " \

"is a near passerine bird in the bee-eater family. It is resident but prone to seasonal movements and " \

"is found widely distributed across Asia from coastal southern Iran east through the Indian " \

"subcontinent to Vietnam "

elif out == "Brown-Headed-Barbet":

pre = "The brown-headed barbet is an Asian barbet species native to the Indian subcontinent, where it " \

"inhabits tropical and subtropical moist broadleaf forests "

elif out == "Cattle-Egret":

pre = "They are white birds adorned with buff plumes in the breeding season. They nest in colonies, " \

"usually near bodies of water and often with other wading birds. "

elif out == "Common-Kingfisher":

pre = "The common kingfisher, also known as the Eurasian kingfisher and river kingfisher, is a small " \

"kingfisher with seven subspecies recognized within its wide distribution across Eurasia and North " \

"Africa. It is resident in much of its range, but migrates from areas where rivers freeze in winter. "

elif out == "Common-Myna":

pre = "The common myna or Indian myna, sometimes spelled mynah, is a bird in the family Sturnidae, native to " \

"Asia. An omnivorous open woodland bird with a strong territorial instinct, the common myna has adapted " \

"extremely well to urban environments. "

elif out == "Common-Rosefinch":

pre = "The Common-Rosefinch is an Asian barbet species native to the Indian subcontinent, where it inhabits " \

"tropical and subtropical moist broadleaf forests "

elif out == "Common-Tailorbird":

pre = "The common tailorbird is a songbird found across tropical Asia. Popular for its nest made of leaves " \

"sewn together and immortalized by Rudyard Kipling as Darzee in his Jungle Book, it is a common " \

"resident in urban gardens. "

elif out == "Coppersmith-Barbet":

pre = "The coppersmith barbet, also called crimson-breasted barbet and coppersmith, is an Asian barbet with " \

"crimson forehead and throat, known for its metronomic call that sounds similar to a coppersmith " \

"striking metal with a hammer. It is a resident bird in the Indian subcontinent and parts of Southeast " \

"Asia "

elif out == "Forest-Wagtail":

pre = "The forest wagtail is a medium-sized passerine bird in the wagtail family Motacillidae. It has a " \

"distinctive plumage that sets it apart from other wagtails and has the habit of wagging its tail " \

"sideways unlike the usual up and down movements of the other wagtail species "

elif out == "Brown-Headed-Barbet":

pre = "The brown-headed barbet is an Asian barbet species native to the Indian subcontinent, where it " \

"inhabits tropical and subtropical moist broadleaf forests "

elif out == "Gray-Wagtail":

pre = "The grey wagtail is a member of the wagtail family, Motacillidae, measuring around 18–19 cm overall " \

"length. The species looks somewhat similar to the yellow wagtail but has the yellow on its underside " \

"restricted to the throat and vent. Breeding males have a black throat. "

elif out == "Hoopoe":

pre = "Hoopoes are colourful birds found across Africa, Asia, and Europe, notable for their distinctive crown " \

"of feathers which can be raised or lowered at will. Three living and one extinct species are " \

"recognized, though for many years all of the extant species were lumped as a single species—Upupa " \

"epops. "

elif out == "House-Crow":

pre = "The house crow, also known as the Indian, greynecked, Ceylon or Colombo crow, is a common bird of the " \

"crow family that is of Asian origin but now found in many parts of the world, where they arrived " \

"assisted by shipping. It is between the jackdaw and the carrion crow in size but is slimmer than " \

"either. "

elif out == "Indian-Grey-Hornbill":

pre = "The Indian gray hornbill is a common hornbill found on the Indian subcontinent. It is mostly arboreal " \

"and is commonly sighted in pairs. It has grey feathers all over the body with a light grey or dull " \

"white belly. The horn is black or dark grey with a casque extending to the point of curvature of the " \

"horn. "

elif out == "Indian-Peacock":

pre = "The Indian peafowl (Pavo cristatus), also known as the common peafowl or blue peafowl, is a peafowl " \

"species native to the Indian subcontinent. "

elif out == "Indian-Pitta":

pre = "The Indian pitta is a passerine bird native to the Indian subcontinent. It inhabits scrub jungle, " \

"deciduous and dense evergreen forest. It breeds in the forests of the Himalayas, hills of central and " \

"western India, and migrates to other parts of the peninsula in winter. "

elif out == "Indian-Roller":

pre = "The Indian roller is a bird of the family Coraciidae. It is 30–34 cm long with a wingspan of 65–74 cm " \

"and weighs 166–176 g. The face and throat are pinkish, the head and back are brown, with blue on the " \

"rump and contrasting light and dark blue on the wings and tail. "

elif out == "Jungle-Babbler":

pre = "The jungle babbler is a member of the family Leiothrichidae found in the Indian subcontinent. Jungle " \

"babblers are gregarious birds that forage in small groups of six to ten birds, a habit that has given "

elif out == "Northern-Lapwing":

pre = "The northern lapwing, also known as the peewit or pewit, tuit or tewit, green plover, or pyewipe or " \

"just lapwing, is a bird in the lapwing subfamily. It is common through temperate Eurosiberia. "

elif out == "Red-Wattled-Lapwing":

pre = "The red-wattled lapwing is an Asian lapwing or large plover, a wader in the family Charadriidae. Like " \

"other lapwings they are ground birds that are incapable of perching. "

elif out == "Ruddy-Shelduck":

pre = "The ruddy shelduck, known in India as the Brahminy duck, is a member of the family Anatidae. It is a " \

"distinctive waterfowl, 58 to 70 cm in length with a wingspan of 110 to 135 cm. "

elif out == "Rufous-Treepie":

pre = "The rufous treepie is a treepie, native to the Indian Subcontinent and adjoining parts of Southeast " \

"Asia. It is a member of the crow family, Corvidae. It is long tailed and has loud musical calls making " \

"it very conspicuous. It is found commonly in open scrub, agricultural areas, forests as well as urban " \

"gardens. "

elif out == "Sarus-Crane":

pre = "The sarus crane is a large nonmigratory crane found in parts of the Indian subcontinent, Southeast " \

"Asia, and northern Australia "

elif out == "White-Breasted-Kingfisher":

pre = "The white-throated kingfisher also known as the white-breasted kingfisher is a tree kingfisher, " \

"widely distributed in Asia from the Sinai east through the Indian subcontinent to China and Indonesia. " \

"This kingfisher is a resident over much of its range, although some populations may make short " \

"distance movements. "

elif out == "White-Breasted-Waterhen":

pre = "The white-breasted waterhen is a waterbird of the rail and crake family, Rallidae, that is widely " \

"distributed across South and Southeast Asia. They are dark slaty birds with a clean white face, " \

"breast and belly. "

elif out == "White-Wagtail":

pre = "The white wagtail is a small passerine bird in the family Motacillidae, which also includes pipits and " \

"longclaws. The species breeds in the Palearctic zone in most of Europe and Asia and parts of North " \

"Africa; it also has a toehold in western Alaska as a scarce breeder "

messagebox.showinfo("Result", "Classification Result : " + str(out))

messagebox.showinfo("Info", "Bird Information : " + str(pre))

def main\_account\_screen():

global main\_screen

main\_screen = Tk()

width = 600

height = 500

screen\_width = main\_screen.winfo\_screenwidth()

screen\_height = main\_screen.winfo\_screenheight()

x = (screen\_width / 2) - (width / 2)

y = (screen\_height / 2) - (height / 2)

main\_screen.geometry("%dx%d+%d+%d" % (width, height, x, y))

main\_screen.resizable(0, 0)

# main\_screen.geometry("300x250")

main\_screen.configure(bg='lightblue')

main\_screen.title("Bird Image classification ")

Label(text="Bird Image classification ", width="300", height="5", bg='lightblue',

font=("Palatino Linotype", 16)).pack()

Button(text="UploadImage", font=(

'Palatino Linotype', 15), height="2", width="20", bg='lightblue', command=training,

highlightcolor="black").pack(side=TOP)

Label(text="", bg='lightblue').pack()

Button(text="Training", font=(

'Palatino Linotype', 15), height="2", width="20", bg='lightblue', command=fulltraining,

highlightcolor="black").pack(side=TOP)

Label(text="", bg='lightblue').pack()

Button(text="Testing", font=(

'Palatino Linotype', 15), height="2", width="20", bg='lightblue', command=testing).pack(side=TOP)

Label(text="", bg='lightblue').pack()

main\_screen.mainloop()

main\_account\_screen()

import matplotlib.pyplot as plt

import warnings

import seaborn as sns

import numpy

warnings.filterwarnings('ignore')

batch\_size = 32

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale=1/255)

train\_generator = train\_datagen.flow\_from\_directory('Dataset',target\_size=(200, 200), batch\_size=batch\_size,

classes = ['Asian-Green-Bee-Eater', 'Brown-Headed-Barbet', 'Cattle-Egret', 'Common-Kingfisher', 'Common-Myna', 'Common-Rosefinch', 'Common-Tailorbird', 'Coppersmith-Barbet', 'Forest-Wagtail', 'Gray-Wagtail', 'Hoopoe', 'House-Crow', 'Indian-Grey-Hornbill', 'Indian-Peacock', 'Indian-Pitta', 'Indian-Roller', 'Jungle-Babbler', 'Northern-Lapwing', 'Red-Wattled-Lapwing', 'Ruddy-Shelduck', 'Rufous-Treepie', 'Sarus-Crane', 'White-Breasted-Kingfisher', 'White-Breasted-Waterhen', 'White-Wagtail'],class\_mode='categorical')

test\_datagen = ImageDataGenerator(rescale=1/255)

test\_generator = test\_datagen.flow\_from\_directory('Dataset', target\_size=(200, 200), batch\_size=batch\_size,

classes = ['Asian-Green-Bee-Eater', 'Brown-Headed-Barbet', 'Cattle-Egret', 'Common-Kingfisher', 'Common-Myna', 'Common-Rosefinch', 'Common-Tailorbird', 'Coppersmith-Barbet', 'Forest-Wagtail', 'Gray-Wagtail', 'Hoopoe', 'House-Crow', 'Indian-Grey-Hornbill', 'Indian-Peacock', 'Indian-Pitta', 'Indian-Roller', 'Jungle-Babbler', 'Northern-Lapwing', 'Red-Wattled-Lapwing', 'Ruddy-Shelduck', 'Rufous-Treepie', 'Sarus-Crane', 'White-Breasted-Kingfisher', 'White-Breasted-Waterhen', 'White-Wagtail'],

class\_mode='categorical',shuffle=False)

import tensorflow as tf

model = tf.keras.models.Sequential([

# Note the input shape is the desired size of the image 200x 200 with 3 bytes color

# The first convolution

tf.keras.layers.Conv2D(16, (3,3), activation='relu', input\_shape=(200, 200, 3)),

tf.keras.layers.MaxPooling2D(2, 2),

# The second convolution

tf.keras.layers.Conv2D(32, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

# The third convolution

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

# The fourth convolution

tf.keras.layers.Conv2D(128, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

# The fifth convolution

tf.keras.layers.Conv2D(128, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

# Flatten the results to feed into a dense layer

tf.keras.layers.Flatten(),

# 128 neuron in the fully-connected layer

tf.keras.layers.Dense(256, activation='relu'),

# 5 output neurons for 5 classes with the softmax activation

tf.keras.layers.Dense(25, activation='softmax')

])

model.summary()

from tensorflow.keras.optimizers import RMSprop

early = tf.keras.callbacks.EarlyStopping(monitor='val\_loss',patience=5)

model.compile(loss='categorical\_crossentropy', optimizer=RMSprop(lr=0.001),metrics=['accuracy'])

total\_sample=train\_generator.n

n\_epochs = 10

history = model.fit\_generator(train\_generator,steps\_per\_epoch=int(total\_sample/batch\_size),epochs=n\_epochs, verbose=1)

model.save('model.h5')

acc = history.history['accuracy']

loss = history.history['loss']

epochs = range(1, len(acc) + 1)

# Train and validation accuracy

plt.plot(epochs, acc, 'b', label=' accurarcy')

plt.title('accurarcy')

plt.legend()

plt.figure()

# Train and validation loss

plt.plot(epochs, loss, 'b', label=' loss')

plt.title(' loss')

plt.legend()

plt.show()

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

test\_steps\_per\_epoch = numpy.math.ceil(test\_generator.samples / test\_generator.batch\_size)

predictions = model.predict\_generator(test\_generator, steps=test\_steps\_per\_epoch)

# Get most likely class

predicted\_classes = numpy.argmax(predictions, axis=1)

true\_classes = test\_generator.classes

class\_labels = list(test\_generator.class\_indices.keys())

print('Classification Report')

report = classification\_report(true\_classes, predicted\_classes, target\_names=class\_labels)

print(report)

print('confusion matrix')

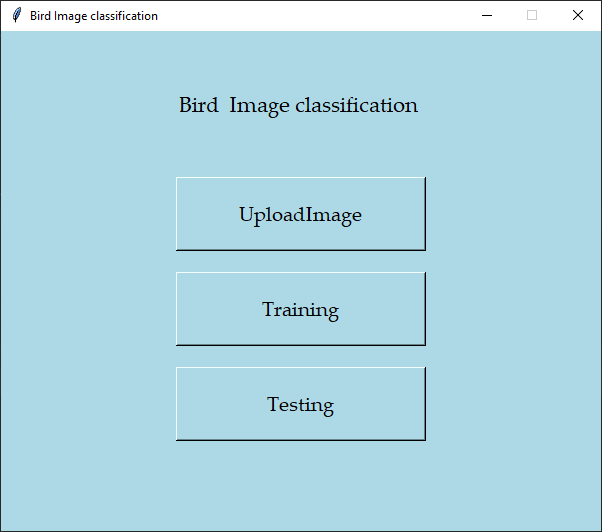
confusion\_matrix= confusion\_matrix(true\_classes, predicted\_classes)

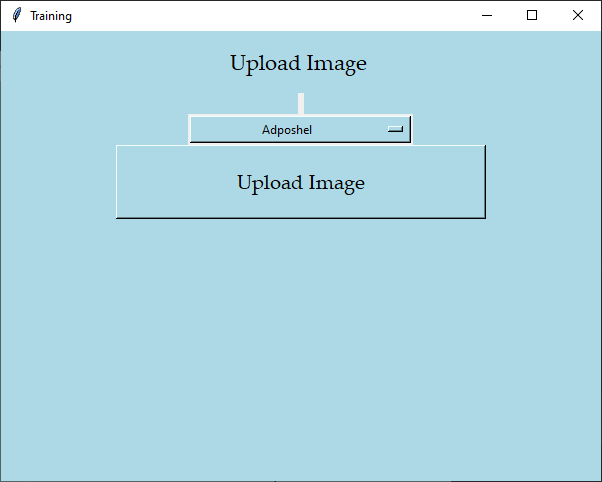
print(confusion\_matrix)

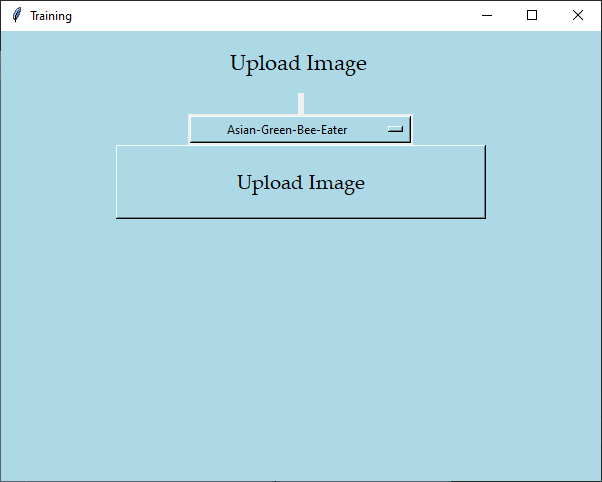
sns.heatmap(confusion\_matrix, annot = True)

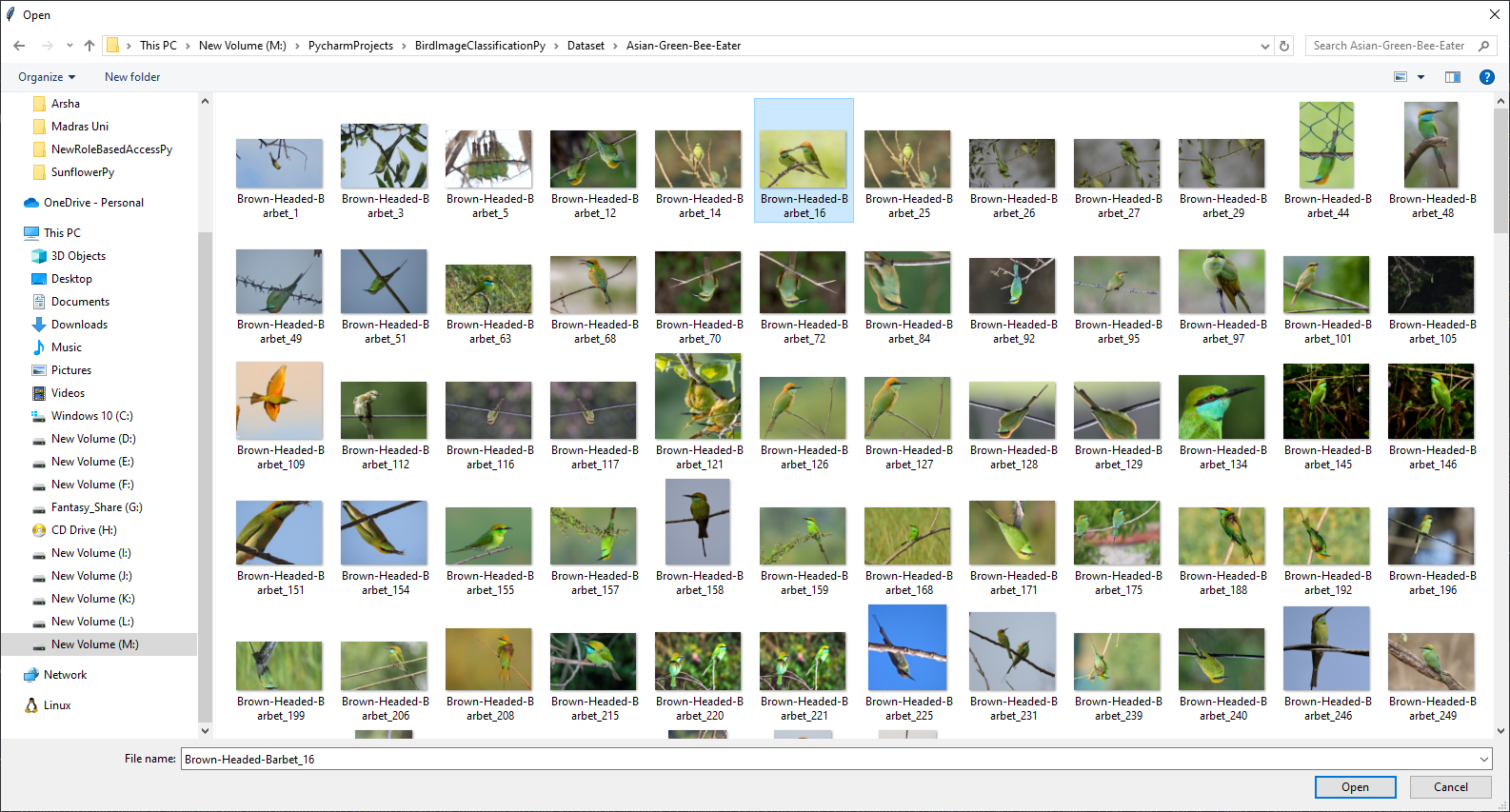
plt.show()

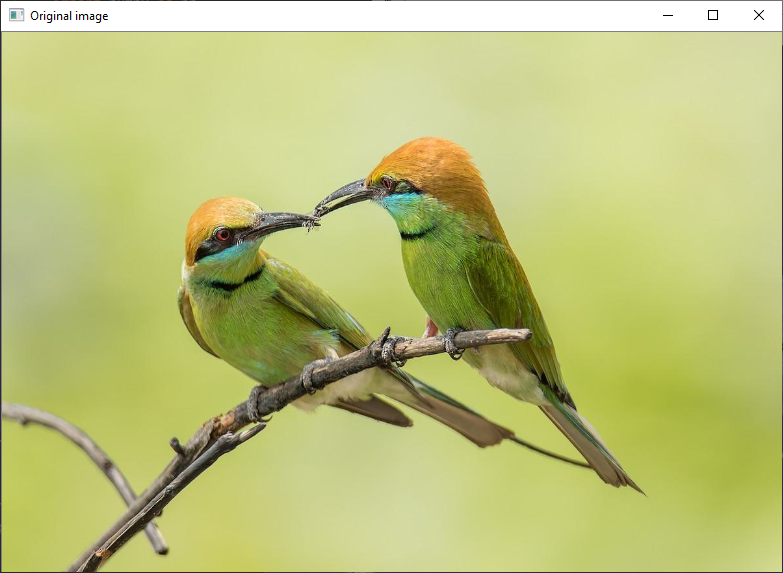
**4.2 SCREENSHOTS**

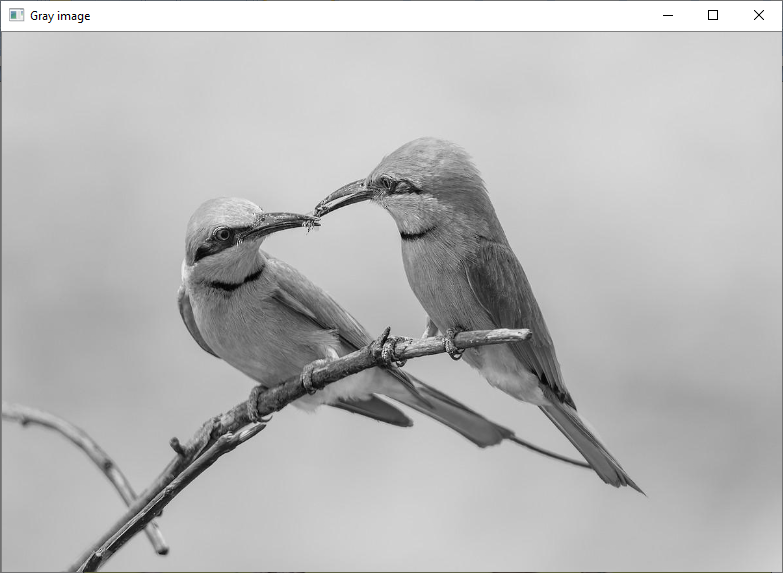


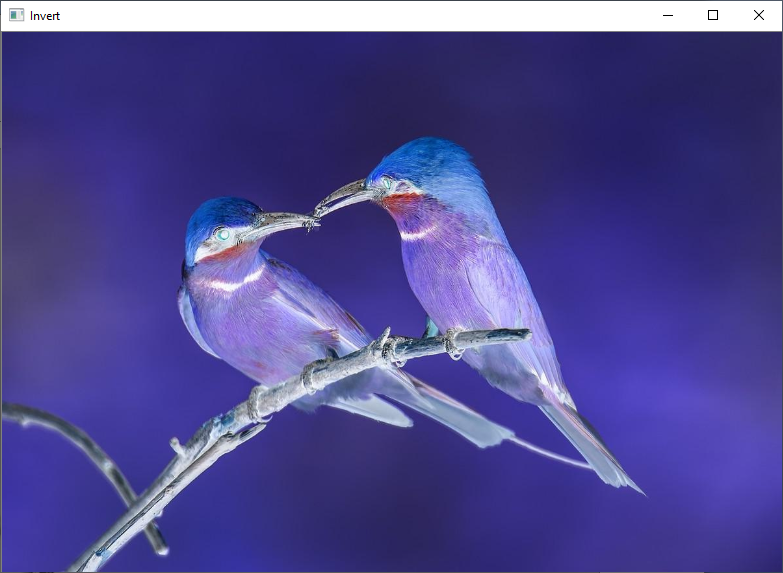


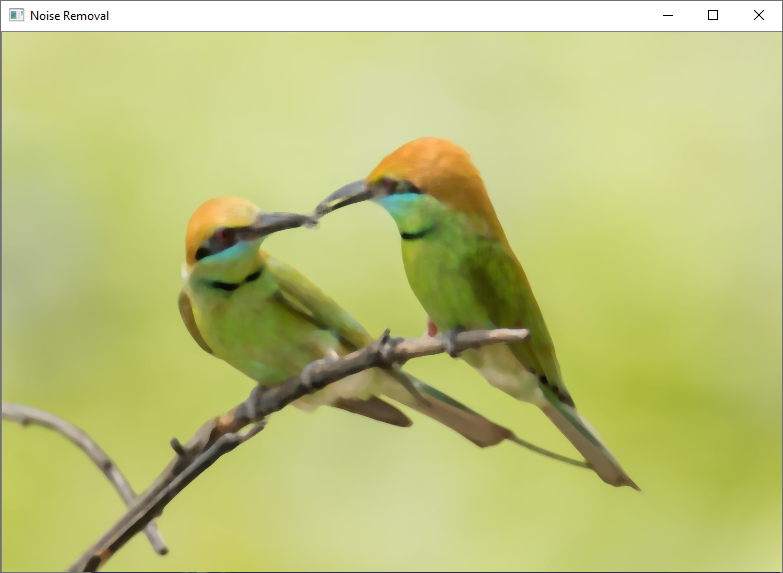












M:\PycharmProjects\BirdImageClassificationPy\venv\Scripts\python.exe M:/PycharmProjects/BirdImageClassificationPy/Model.py

2024-11-17 12:33:28.624085: W tensorflow/stream\_executor/platform/default/dso\_loader.cc:64] Could not load dynamic library 'cudart64\_110.dll'; dlerror: cudart64\_110.dll not found

2024-11-17 12:33:28.624787: I tensorflow/stream\_executor/cuda/cudart\_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.

Found 7500 images belonging to 25 classes.

Found 7500 images belonging to 25 classes.

2024-11-17 12:33:34.395236: W tensorflow/stream\_executor/platform/default/dso\_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found

2024-11-17 12:33:34.395468: W tensorflow/stream\_executor/cuda/cuda\_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)

2024-11-17 12:33:34.410879: I tensorflow/stream\_executor/cuda/cuda\_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-9BF8NUN

2024-11-17 12:33:34.411210: I tensorflow/stream\_executor/cuda/cuda\_diagnostics.cc:176] hostname: DESKTOP-9BF8NUN

2024-11-17 12:33:34.419011: I tensorflow/core/platform/cpu\_feature\_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 198, 198, 16) 448

max\_pooling2d (MaxPooling2D (None, 99, 99, 16) 0

)

conv2d\_1 (Conv2D) (None, 97, 97, 32) 4640

max\_pooling2d\_1 (MaxPooling (None, 48, 48, 32) 0

2D)

conv2d\_2 (Conv2D) (None, 46, 46, 64) 18496

max\_pooling2d\_2 (MaxPooling (None, 23, 23, 64) 0

2D)

conv2d\_3 (Conv2D) (None, 21, 21, 128) 73856

max\_pooling2d\_3 (MaxPooling (None, 10, 10, 128) 0

2D)

conv2d\_4 (Conv2D) (None, 8, 8, 128) 147584

max\_pooling2d\_4 (MaxPooling (None, 4, 4, 128) 0

2D)

flatten (Flatten) (None, 2048) 0

dense (Dense) (None, 256) 524544

dense\_1 (Dense) (None, 25) 6425

=================================================================

Total params: 775,993

Trainable params: 775,993

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch 1/10

WARNING:tensorflow:AutoGraph could not transform <function Model.make\_train\_function.<locals>.train\_function at 0x0000024875BFDB88> and will run it as-is.

Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH\_VERBOSITY=10`) and attach the full output.

Cause: 'arguments' object has no attribute 'posonlyargs'

To silence this warning, decorate the function with @tf.autograph.experimental.do\_not\_convert

2024-11-17 12:33:36.051379: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2024-11-17 12:33:36.697904: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

1/234 [..............................] - ETA: 7:18 - loss: 3.2115 - accuracy: 0.12502024-11-17 12:33:36.898347: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2024-11-17 12:33:37.522287: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2/234 [..............................] - ETA: 3:09 - loss: 3.8832 - accuracy: 0.09382024-11-17 12:33:37.712822: W tensorflow/core/framework/cpu\_allocator\_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

234/234 [==============================] - 194s 823ms/step - loss: 2.9430 - accuracy: 0.0992

Epoch 2/10

234/234 [==============================] - 176s 753ms/step - loss: 2.4101 - accuracy: 0.2430

Epoch 3/10

234/234 [==============================] - 172s 734ms/step - loss: 2.0574 - accuracy: 0.3578

Epoch 4/10

234/234 [==============================] - 195s 833ms/step - loss: 1.7750 - accuracy: 0.4501

Epoch 5/10

234/234 [==============================] - 192s 821ms/step - loss: 1.4904 - accuracy: 0.5383

Epoch 6/10

234/234 [==============================] - 192s 818ms/step - loss: 1.2249 - accuracy: 0.6216

Epoch 7/10

234/234 [==============================] - 194s 827ms/step - loss: 0.9597 - accuracy: 0.6990

Epoch 8/10

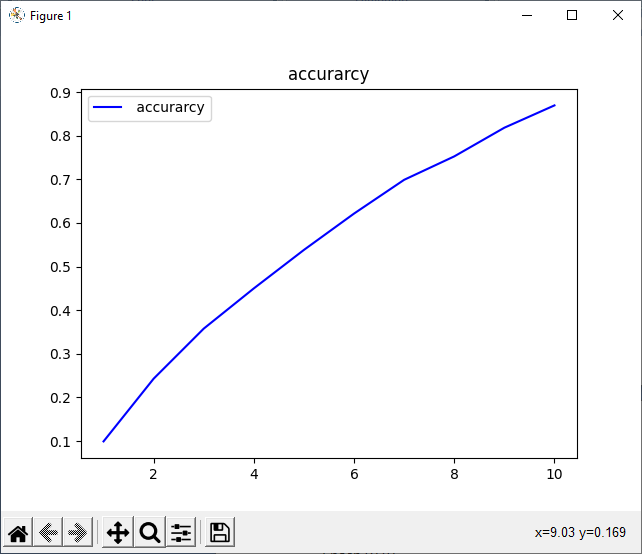
234/234 [==============================] - 194s 828ms/step - loss: 0.7511 - accuracy: 0.7527

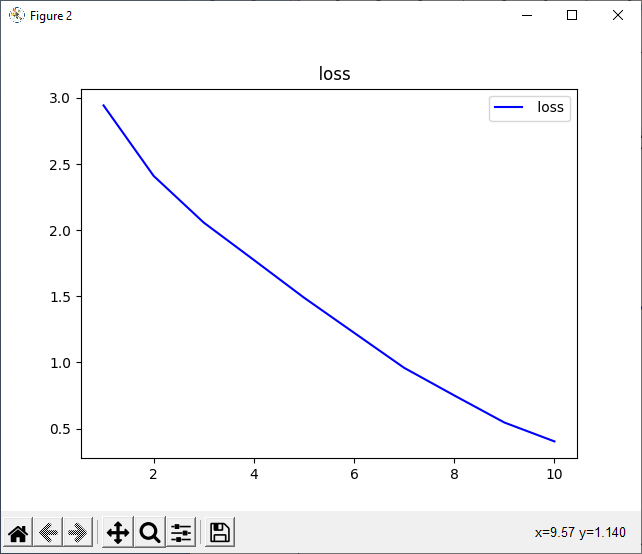
Epoch 9/10

234/234 [==============================] - 195s 831ms/step - loss: 0.5465 - accuracy: 0.8184

Epoch 10/10

234/234 [==============================] - 197s 841ms/step - loss: 0.4037 - accuracy: 0.8696





Classification Report

precision recall f1-score support

Asian-Green-Bee-Eater 0.98 0.95 0.96 300

Brown-Headed-Barbet 0.94 0.93 0.93 300

Cattle-Egret 0.93 0.96 0.94 300

Common-Kingfisher 0.98 0.94 0.96 300

Common-Myna 0.99 0.95 0.97 300

Common-Rosefinch 0.71 0.98 0.82 300

Common-Tailorbird 0.99 0.80 0.88 300

Coppersmith-Barbet 0.85 0.96 0.90 300

Forest-Wagtail 0.93 0.96 0.94 300

Gray-Wagtail 0.98 0.98 0.98 300

Hoopoe 0.97 0.93 0.95 300

House-Crow 0.99 0.88 0.93 300

Indian-Grey-Hornbill 0.83 0.92 0.87 300

Indian-Peacock 1.00 0.91 0.95 300

Indian-Pitta 0.97 0.98 0.98 300

Indian-Roller 0.96 0.88 0.92 300

Jungle-Babbler 0.94 0.94 0.94 300

Northern-Lapwing 0.93 0.90 0.91 300

Red-Wattled-Lapwing 0.99 0.89 0.94 300

Ruddy-Shelduck 0.91 0.96 0.94 300

Rufous-Treepie 0.92 0.95 0.93 300

Sarus-Crane 0.92 0.94 0.93 300

White-Breasted-Kingfisher 0.99 0.88 0.93 300

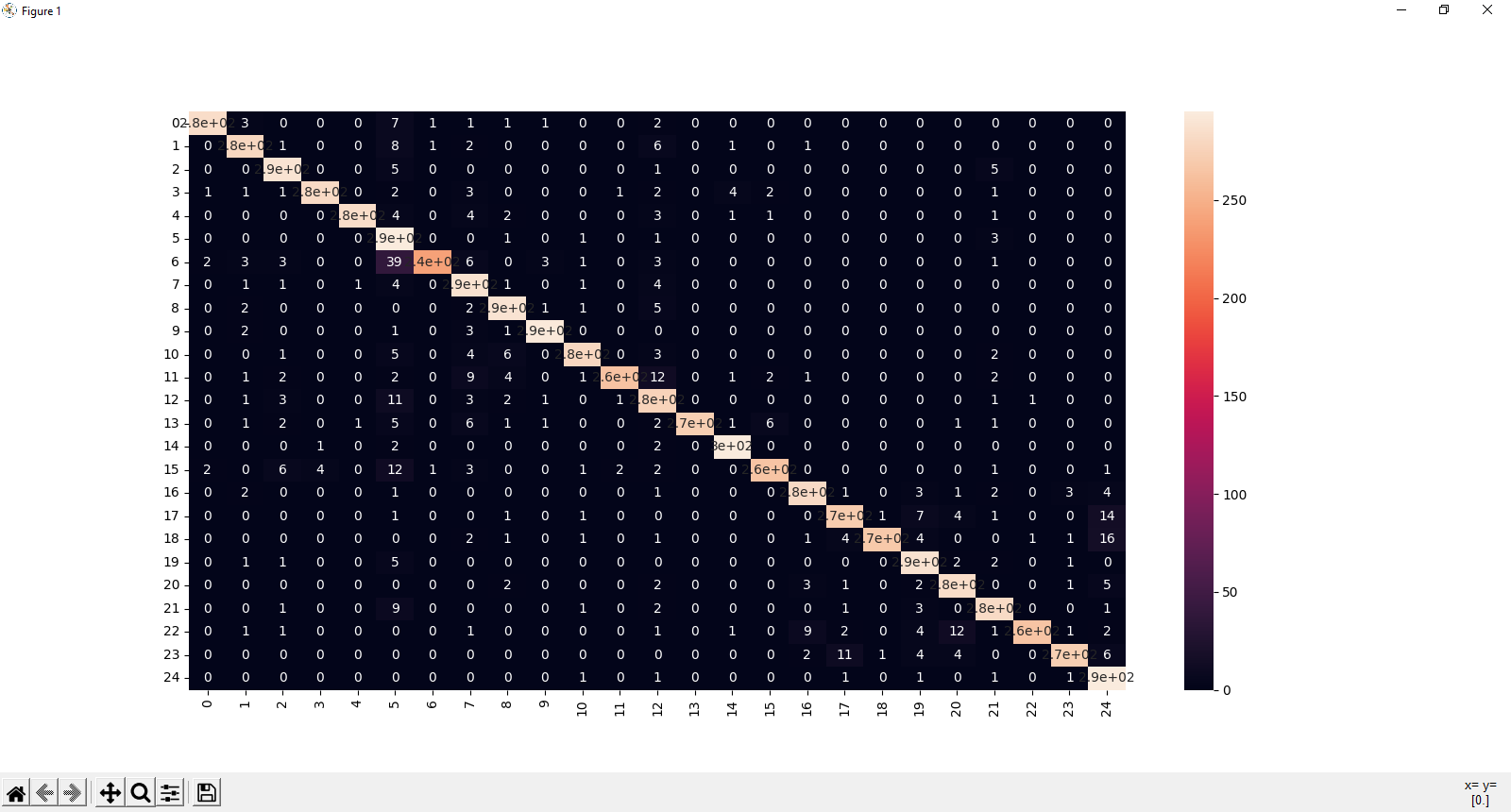
White-Breasted-Waterhen 0.97 0.91 0.94 300

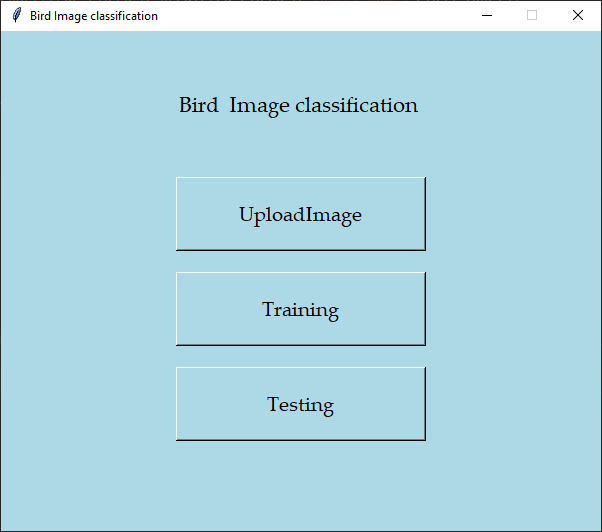
White-Wagtail 0.86 0.98 0.91 300

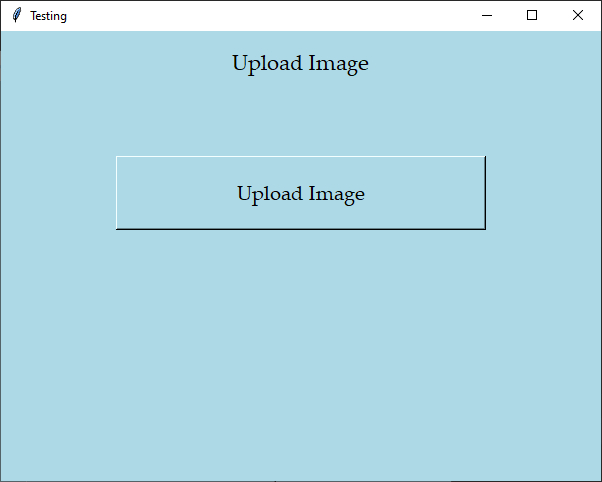
accuracy 0.93 7500

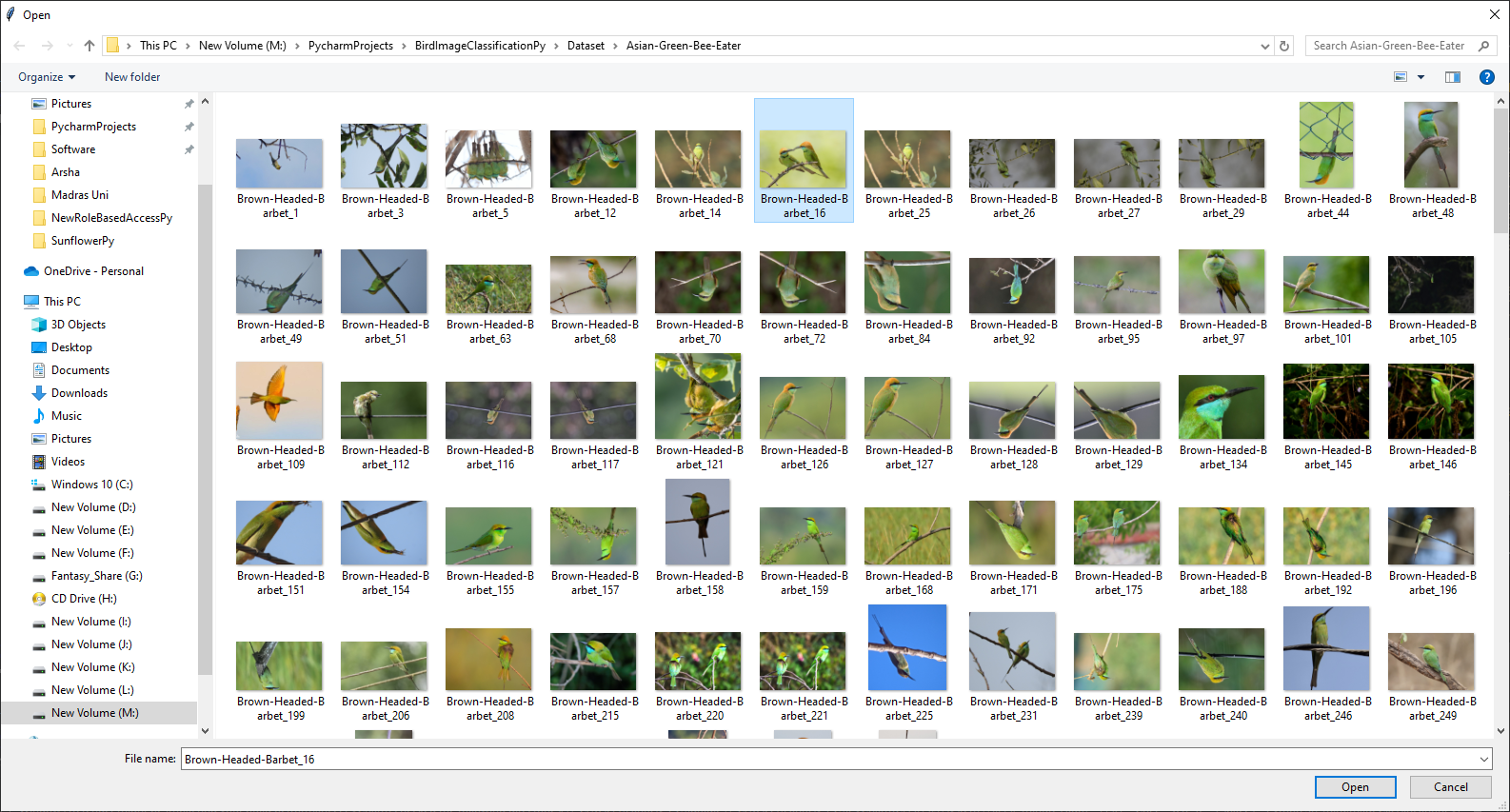
macro avg 0.94 0.93 0.93 7500

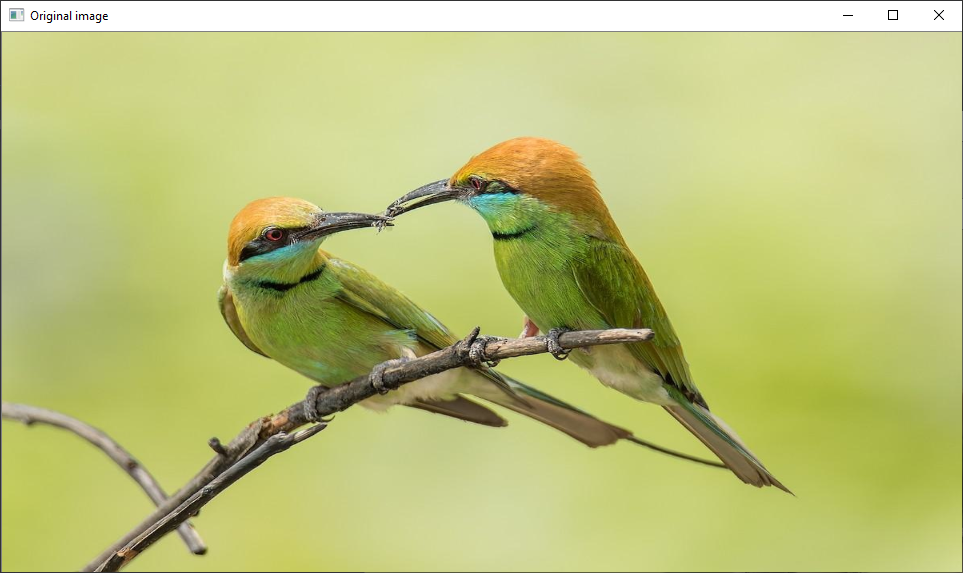
weighted avg 0.94 0.93 0.93 7500

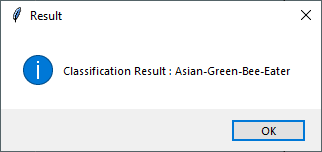


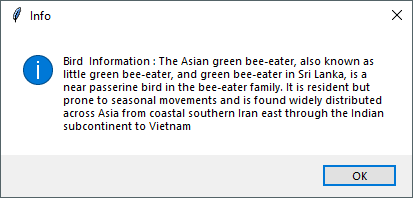












1. **CONCLUSION**

In conclusion, the bird species classification project using a Convolutional Neural Network (CNN) demonstrates the power of deep learning techniques in accurately identifying and predicting bird species based on images. By leveraging advanced image preprocessing, feature extraction, and classification processes, the model is able to differentiate between various bird species with a high degree of accuracy. The ability to classify bird species and provide detailed information about each species has significant implications for environmental monitoring, wildlife conservation, and educational purposes. Additionally, the system's potential for real-time applications in mobile apps and field devices enhances its practical value. Overall, this project highlights the effectiveness of CNNs in solving complex image recognition tasks and contributes to the growing field of automated species identification, offering a reliable tool for researchers, enthusiasts, and conservationists alike.

**REFERENCES**

[1] Mehra, Ritika, and R. Ragahvendra. "Using Classifier with Gated Recurrent Unit-Sigmoid Perceptron, Order to Get the Right Bird Species Detection." *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*. IEEE, 2023.

[2] Mahesh, T. R., et al. "Hybrid ant lion mutated ant colony optimizer technique with particle swarm optimization for leukemia prediction using microarray gene data." *IEEE Access* (2024).

[3] Ren, Guangxin, et al. "Rapid characterization of black tea taste quality using miniature NIR spectroscopy and electronic tongue sensors." *Biosensors* 13.1 (2023): 92.

[4] Edwards, Kiah, et al. "Raw Beef Patty Analysis Using Near-Infrared Hyperspectral Imaging: Identification of Four Patty Categories." *Sensors* 23.2 (2023): 697.

[5] Mohseni, Parvin, and Abozar Ghorbani. "Exploring the Synergy of Artificial Intelligence in Microbiology: Advancements, Challenges, and Future Prospects." *Computational and Structural Biotechnology Reports* (2024): 100005.

[6] Mohmad, G. H., and Radhakrishnan Delhibabu. "Speech Databases, speech features and classifiers in speech emotion recognition: A Review." *IEEE Access* (2024).

[7] Ropelewska, Ewa, et al. "Rapid detection of changes in image textures of carrots caused by freeze-drying using image processing techniques and machine learning algorithms." *Sustainability* 15.8 (2023): 7011.

[8] Liu, Kang, et al. "Analysis and Protection Research of Bird Droppings on the Electric Field Distribution near 330 kV Transmission Line Cathead Towers with Composite Insulators." *IEEE Access* (2023).

[9] Eniwaiye, A. A., et al. "Effect of spice-treated sundried bovine rumen digesta on performance, carcass characteristics and nutrient digestibility of finisher broiler chickens." (2023).

[10] Huded, Mr Dayananda, and Mallikarjun M. Maradi. "FINANCIAL INCLUSION AND WOMEN EMPOWERMENT THROUGH SELF HELP GROUPS: A BIRD EYEVIEW."