PREDICTING BIRD SPECIES

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

SINDHU KALEESWARAN(RA2011026010082)
APARNA SURESH(RA2011026010099)
CHEREDDY SOWMYA SRI(RA2011026010113)

Under the guidance of
Dr.M.S.Abirami

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"PREDICTING BIRD SPECIES"** is the bona fide work of **SINDHU KALEESWARAN(RA2011026010082),APARNA SURESH (RA2011026010099),CHEREDDY SOWMYA SRI(RA2011026010113)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
DR.M.S.ABIRAMI	Dr. R.Annie Uthra
GUIDE	**HEAD OF THE DEPARTMENT**
Assistant Professor	Professor & Head
Department of Computing Technologies	Department of Computational Intelligence

ABSTRACT

Predicting bird species is a challenging task that requires the integration of ecological knowledge, advanced data analysis techniques, and machine learning algorithms. This paper explores the current state of bird species prediction and highlights potential future enhancements in the field.

The prediction of bird species involves the analysis of various features, including plumage, behavior, habitat, and geographic location. Researchers have leveraged advancements in technology, such as high-resolution cameras and bioacoustic recording devices, to gather large volumes of data for analysis. Machine learning algorithms are then employed to detect patterns and characteristics that

Future enhancements for bird species prediction include improving data collection methods by utilizing advanced sensors and technologies like remote sensing and bioacoustics. Integrating multimodal data sources, such as visual, acoustic, and environmental variables, can enhance prediction accuracy. The application of deep learning techniques, transfer learning, and active learning can further improve model performance.

Other enhancements involve considering contextual information, developing models for fine-grained classification, and enabling transferability across different geographic regions. Additionally, the use of explainable AI techniques can provide insights into the decision-making process of prediction models.

Predicting bird species has broad applications in conservation efforts, biodiversity monitoring, and ecological research. Accurate identification and monitoring of bird species contribute to understanding population trends, habitat suitability, and the impacts of environmental changes.

In conclusion, while predicting bird species is a complex task, ongoing advancements in technology, data collection, and machine learning techniques hold promise for improving accuracy and expanding the scope of bird species prediction. Continued research and collaboration will facilitate the development of robust models that enhance our understanding and conservation of avian biodiversity.

TABLE OF CONTENTS

INTRODUCTION

Predicting bird species involves using data and machine learning algorithms to identify the type of bird based on various features such as physical characteristics, behavior, and habitat. This can be done using techniques such as image recognition, natural language processing, and statistical analysis.
Accurately identifying bird species can be important for a variety of reasons, such as tracking population trends, studying migration patterns, and understanding the impact of environmental changes on bird populations. Additionally, birdwatchers and enthusiasts may also use bird identification tools to help them identify the birds they observe in the wild.
To predict bird species, various data sources can be used, including photographs, audio recordings, and field observations. Machine learning models can then be trained on this data to recognize patterns and make predictions about the species of the bird.
One of the most common techniques for predicting bird species is through image recognition. This involves using computer vision algorithms to analyze features of a bird's physical appearance, such as its coloration, markings, and shape, and matching those features to known species in a database. Researchers and bird enthusiasts can take photographs of birds in the wild or in captivity and upload them to an image recognition tool or app that will analyze the image and provide a predicted species identification.
Another method for predicting bird species is through the use of audio recordings. This technique is particularly useful for identifying birds by their songs or calls. Researchers can record bird songs and calls using specialized equipment and then use machine learning algorithms to analyze the recordings and identify the species of the bird.
In addition to predicting bird species, machine learning techniques can also be used to study bird behavior and migration patterns. Researchers can track bird movements using GPS tags and other tracking devices and then use machine learning algorithms to analyze the data and identify patterns in the birds' behavior.

Overall, predicting bird species using machine learning techniques has the potential to revolutionize the field of ornithology and provide valuable insights into bird behavior, migration patterns, and conservation efforts.

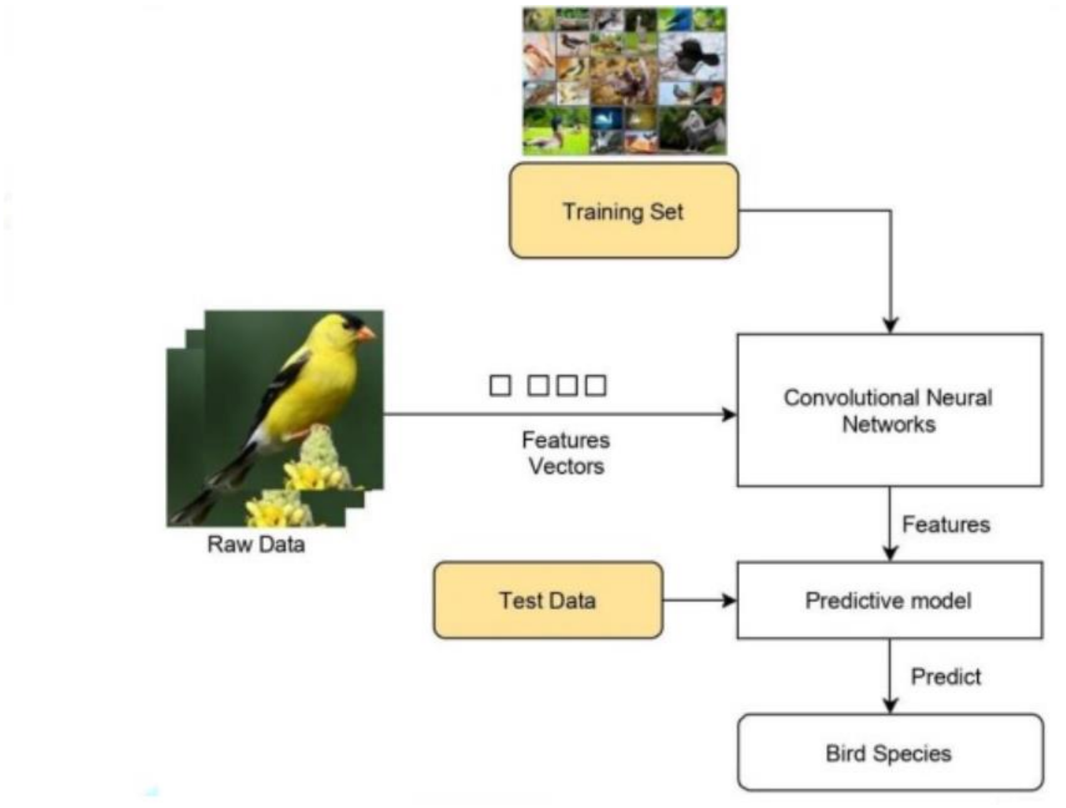
LITERATURE SURVEY

There are some of the recent studies and research papers on bird species prediction:

1. Bird species identification has been the subject of extensive research in the field of ornithology.
2. Image Processing: Bird species can also be identified based on their visual characteristics. Image processing techniques have been used to identify bird species from photographs or videos. This can involve analyzing the shape, color, and markings of the bird to identify the species.
3. "Bird Species Recognition from Short Duration Recordings using Deep Convolutional Neural Networks presents a deep learning approach for bird species recognition from short duration audio recordings, including feature extraction and training of the network.
4. Bird Species Identification with Convolutional Neural Networks advancements in bird species identification using CNNs, including data augmentation, feature extraction, and transfer learning.
5. A Comparative Study of Bird Species Identification using Machine Learning Techniques , compares various machine learning techniques for bird species identification, including SVM, random forests, and decision trees.
6. "Visual Recognition of Bird Species provides a comprehensive survey of visual recognition techniques for bird species identification, including traditional computer vision techniques and deep learning approaches.
7. One limitation of citing sources for bird species identification using computer vision is the availability and quality of the data. There may be limited or incomplete datasets available, which can limit the effectiveness of machine learning algorithms used for bird species identification. Additionally, images of birds may be taken under different lighting conditions, angles, or distances, which can make it challenging to develop accurate models that can identify bird species under different conditions.
8. Overall, bird species identification is a complex and challenging task that requires a combination of different approaches. Advances in machine learning and computer vision are likely to continue to improve bird species identification in the future.

SYSTEM ARCHITECTURE AND DESIGN

Architecture diagram for proposed model:



Description for modules and components:

1.CNN Configuration: The CNN model conguration for bird species identication utilized a stack of convolutional layers comprising an input layer, two Fully connected(FC) layers, and one final output softmax layer. Convolutional layers apply a convolution operation to the input and this passes the resulting information on to the next layer. Each convolutional layer comprised 5*5 convolution, Batch normalization(BN): It is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the impact of stabilizing the learning process and significantly reducing the number of training epochs required to train deep networks, Rectified linear unit(ReLU) activation: It is a piecewise linear function. If it is positive, then produces the input directly, otherwise, it will output zero, and pooling layers: By combining the outputs of neuron clusters at one layer into a single neuron in the next layer the pooling layers reduces the dimensions of the data.

2.Image processing module: This module is responsible for receiving and processing the bird image, which may be captured using a camera or extracted from a video. The module typically applies various techniques, such as image filtering, segmentation, and feature extraction, to extract relevant information from the image.

3.Feature extraction module: This module is responsible for extracting relevant features from the bird image, such as the shape, color, texture, and patterns of the bird's feathers. These features can then be used to differentiate between different bird species.

4.Machine learning module: This module is responsible for using machine learning algorithms to train a model on a dataset of labeled bird images. The model can then be used to predict the species of a new bird image based on its extracted features.

5.Classification module: This module is responsible for classifying the bird species based on the predictions made by the machine learning model. It may use various techniques, such as rule-based systems or probabilistic models, to assign a species label to the input image.

6.Database module: This module is responsible for storing a database of known bird species and their corresponding features. It may also store information about the bird's habitat, behavior, and other relevant characteristics.

7.User interface module: This module provides a user-friendly interface for interacting with the system. It may allow users to input bird images, view identification results, and access additional information about the identified bird species.

Bird species identification typically involves several components:

1. Data collection: This component involves collecting data on bird species, such as images, audio recordings, and text descriptions. The data can be collected through various means, such as field surveys, citizen science projects, and online databases.

2. Preprocessing: This component involves preparing the data for analysis by cleaning, normalizing, and transforming it as needed. For example, images may need to be resized, audio recordings may need to be filtered or denoised, and text descriptions may need to be tokenized and lemmatized.

3. Feature extraction: This component involves identifying relevant features in the data that can be used to distinguish between different bird species. For example, features extracted from images may include color, texture, and shape, while features extracted from audio recordings may include frequency, amplitude, and duration.

4. Model development: This component involves developing a machine learning model that can predict the bird species based on the extracted features. The model can be developed using various techniques, such as neural networks, support vector machines, decision trees, or random forests.

5. Evaluation: This component involves testing the accuracy of the model by using it to predict the species of birds in a test dataset. Evaluation metrics, such as accuracy, precision, recall, and F1-score, can be used to assess the performance of the model.

In addition to these components, bird species identification may also involve other components, such as data augmentation, transfer learning, and ensemble methods, depending on the specific techniques used. Overall, the goal of bird species identification is to develop an accurate and reliable method for identifying bird species based on their physical characteristics, behavior, or vocalizations.

METHODOLOGY

ResNet-34 is a popular deep convolutional neural network architecture that has shown to be effective in a wide range of image classification tasks, including bird species classification.
The ResNet-34 architecture consists of 34 layers, including 32 convolutional layers and 2 fully connected layers. The key innovation of ResNet is the use of residual connections, which enable the network to learn deeper and more complex representations of the input images.

In the context of bird species classification, ResNet-34 can be used as follows:

1. Data preparation: The first step is to prepare the dataset of bird images. The dataset should be split into training, validation, and testing sets, with each set containing a balanced number of images from each bird species. The images should be resized to a standard size, normalized, and augmented with random transformations such as rotation and flip.

2. Feature extraction: The next step is to use the ResNet-34 architecture to extract features from the pre-processed bird images. This involves using the pre-trained ResNet-34 model as a feature extractor and passing each bird image through the model to obtain a vector of features. The features can be extracted from the last convolutional layer or the last fully connected layer, depending on the specific implementation.

3. Model training: The extracted features are used to train a fully connected neural network classifier. The classifier typically consists of one or more dense layers, with a softmax activation function in the final layer to produce the probability distribution over the bird species. The classifier is trained using the extracted features as input and the bird species labels as output, using a loss function such as cross-entropy.

4. Model evaluation: Once the classifier is trained, it needs to be evaluated on the testing set to determine its accuracy. The accuracy is typically measured using metrics such as accuracy, precision, recall, and F1-score.

5. Model fine-tuning: If the accuracy of the model is not satisfactory, the model can be fine-tuned by adjusting the hyperparameters, such as learning rate, number of layers, and batch size. This process is repeated until the desired accuracy is achieved.

6. Model deployment: Once the model is trained and evaluated, it can be deployed for bird species classification. This involves using the model to predict the bird species in new images. The model can be integrated into a mobile app or a web service for easy access by users.

In summary, ResNet-34 can be used for bird species classification by using its pre-trained model as a feature extractor, and training a fully connected neural network classifier on the extracted features. With appropriate data preparation, training, and evaluation, ResNet-34 can achieve high accuracy in bird species classification tasks.

```
ResNet34(
  (conv1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(1, 1), padding=(4, 4))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (3): ReLU(inplace=True)
  )
  (res1): Sequential(
    (0): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res2): Sequential(
    (0): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res3): Sequential(
    (0): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (downsample1): Sequential(
    (0): Sequential(
      (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    )
  )
  (res4): Sequential(
    (0): Sequential(
      (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    )
    (1): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res5): Sequential(
    (0): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res6): Sequential(
    (0): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res7): Sequential(
    (0): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res8): Sequential(
    (0): Sequential(
      (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    )
    (1): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (downsample2): Sequential(
    (0): Sequential(
      (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    )
  )
  (res9): Sequential(
    (0): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res10): Sequential(
    (0): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res11): Sequential(
    (0): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res12): Sequential(
    (0): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res13): Sequential(
    (0): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (res14): Sequential(
    (0): Sequential(
      (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      (2): ReLU(inplace=True)
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (1): Sequential(
      (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (downsample3): Sequential(
    (0): Sequential(
      (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
  )
  (res15): Sequential(
    (0): Sequential(
      (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (res16): Sequential(
    (0): Sequential(
      (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
    )
    (1): Sequential(
      (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
    )
  )
  (classifier): Sequential(
    (0): AdaptiveMaxPool2d(output_size=(1, 1))
    (1): Flatten(start_dim=1, end_dim=-1)
    (2): Dropout(p=0.17, inplace=False)
    (3): Linear(in_features=512, out_features=450, bias=True)
  )
)
```

CODING AND TESTING

Implementation:

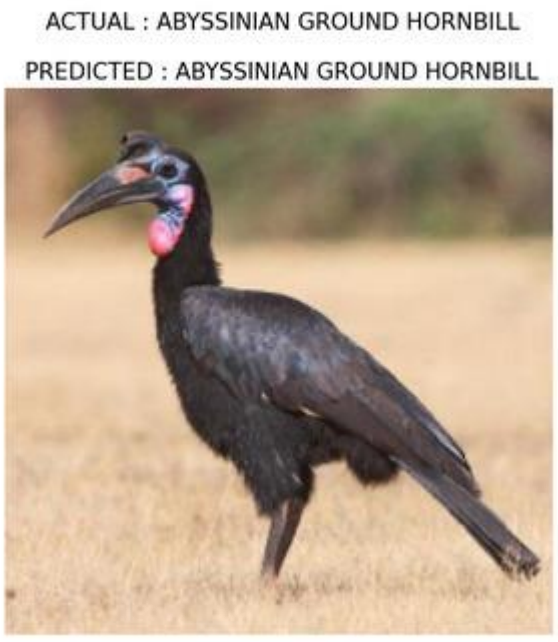
```
import numpy as np
import cv2
import tensorflow as tf
from tensorflow.keras.applications.resnet50 import ResNet50,
preprocess_input, decode_predictions
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

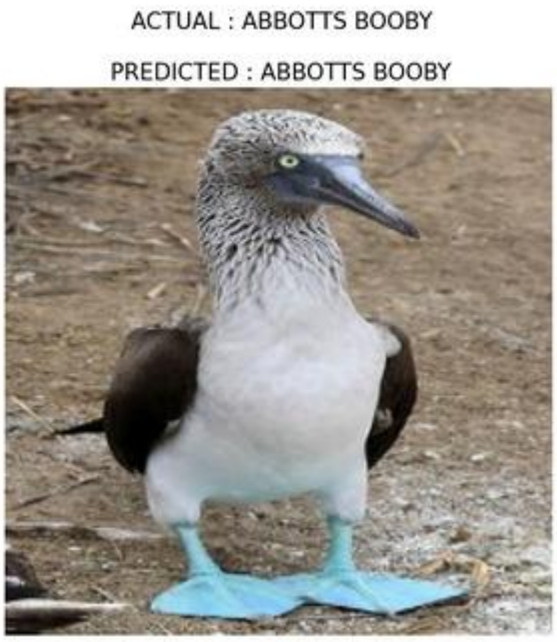
Define the parameters

```
input_shape = (224,224,3)
resnet_weights_path = 'imagenet'
num_classes = 525
image_size = 224
batch_size = 64

data_generator = ImageDataGenerator(preprocessing_function =
preprocess_input)
train_generator = data_generator.flow_from_directory('/kaggle/input/100-bird-
species/train',
                                    target_size = (image_size,image_size),
                                    batch_size = batch_size,
                                    shuffle = True,
                                    class_mode = 'categorical')


val_generator = data_generator.flow_from_directory('/kaggle/input/100-bird-
species/valid',
                                    target_size = (image_size,image_size),
                                    batch_size = batch_size,
                                    shuffle = False,
                                          class_mode = 'categorical')

test_generator = data_generator.flow_from_directory('/kaggle/input/100-bird-
species/test',
                                          target_size = (image_size,image_size),
                                          batch_size = batch_size,
                                          shuffle = False,
                                          class_mode = 'categorical')

import random

train_files = train_generator.filepaths
train_labels = train_generator.classes
class_name = list(train_generator.class_indices.keys())

length_images = len(train_labels)
sample_size = min(length_images, 25)
sample_images = random.sample(range(length_images), sample_size)

for i in range(sample_size):
   plt.rcParams['figure.figsize'] = (20, 30)
   img = plt.imread(train_files[sample_images[i]])
   plt.subplot(5, 5, i+1)
   plt.title(class_name[train_labels[sample_images[i]]])
   plt.imshow(img)
   plt.axis('off')

plt.show()

my_new_model = Sequential()
resnet_weights_path =
'/kaggle/input/resnet50/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5'
RESNET50_POOLING_AVERAGE = 'avg'
my_new_model.add(ResNet50(include_top = False, pooling =
RESNET50_POOLING_AVERAGE, weights = resnet_weights_path))

my_new_model.add(Dense(num_classes, activation = 'softmax'))
my_new_model.layers[0].trainable = True

my_new_model.compile(loss = 'categorical_crossentropy',
                     optimizer='sgd',
                     metrics=['accuracy'])

my_new_model.fit(
    train_generator,steps_per_epoch=20,batch_size= batch_size,
validation_data=val_generator, epochs=3)

import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
# Plot training loss
plt.plot(my_new_model.history.history['loss'])
plt.plot(my_new_model.history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()

test_predictions = my_new_model.predict(test_generator)
test_labels = test_generator.classes
test_files = test_generator.filepaths
class_labels = list(test_generator.class_indices.keys())

from sklearn.metrics import accuracy_score
test_acc = accuracy_score(test_labels, np.argmax(test_predictions, axis=1))

print(f'Test accuracy: {test_acc:.2%}')

from sklearn.metrics import confusion_matrix
import seaborn as sns

# Get predicted labels
test_pred_labels = np.argmax(test_predictions, axis=1)
```

```
# Create confusion matrix
confusion_matrix(test_labels, test_pred_labels)

from sklearn.metrics import confusion_matrix

# Create confusion matrix
cm = confusion_matrix(test_labels, test_pred_labels)

# Select the first 10 classes
num_classes = 10
cm_first_10_classes = cm[:num_classes, :num_classes]

# Print the confusion matrix for the first 10 classes
print(cm_first_10_classes)

from sklearn.metrics import classification_report

# Generate classification report
report = classification_report(test_labels, test_pred_labels,
target_names=class_labels)

# Print classification report
print(report)

length = len(test_labels)
sample = min(length, 20)

for i in range(sample):
    plt.rcParams['figure.figsize'] = (30, 50)
    img = plt.imread(test_files[i])
    pred_label = np.argmax(test_predictions[i])
    actual_label = test_labels[i]

    plt.subplot(5,4,i+1)
    plt.imshow(img)
    plt.title(f" ACTUAL : {class_labels[actual_label]}\n\n PREDICTED :
{class_labels[pred_label]}",fontdict={'fontsize': 15})
    plt.axis('off')

plt.show()
```

SCREENSHOTS AND RESULTS


ACTUAL : ABYSSINIAN GROUND HORNBILL
PREDICTED : ABYSSINIAN GROUND HORNBILL



ACTUAL : AFRICAN CROWNED CRANE

PREDICTED : AFRICAN CROWNED CRANE

ACTUAL : ABBOTTS BABBLER

PREDICTED : YELLOW BELLIED FLOWERPECKER

CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, predicting bird species can be a complex task that requires a combination of data analysis, ecological knowledge, and advanced machine learning techniques. Through the use of various features such as plumage, behavior, habitat, and geographic location, it is possible to build models that can accurately classify and predict bird species.

Over the years, advancements in technology, such as high-resolution cameras and bioacoustic recording devices, have provided researchers with a wealth of data to analyze and extract valuable information about bird species. These data, combined with powerful machine learning algorithms, enable the development of models that can detect subtle patterns and characteristics in bird species.

However, it is important to note that bird species prediction is still an ongoing research area, and there are challenges that need to be addressed. Factors such as variations within species, hybridization, and limited availability of comprehensive datasets can affect the accuracy of predictions. Additionally, the field of ornithology continues to discover new species, which adds to the complexity of bird species prediction.

Despite these challenges, predicting bird species holds great potential for various applications, including conservation efforts, biodiversity monitoring, and ecological research. By accurately identifying and monitoring bird species, we can gain insights into population trends, habitat suitability, and the impact of environmental changes.

In summary, bird species prediction is a fascinating and valuable area of research that combines ecological knowledge with advanced machine learning techniques. Continued advancements in technology, data collection, and algorithm development will contribute to more accurate predictions, leading to better understanding and conservation of avian biodiversity.

There are several potential future enhancements that can further improve bird species prediction:

1. Improved data collection: Collecting high-quality, comprehensive, and diverse datasets is crucial for accurate bird species prediction. Efforts should continue to expand data collection methods, such as using advanced sensors and technologies like remote sensing, bioacoustics, and genetic analysis. This would provide a more comprehensive understanding of bird species characteristics and enable better training of prediction models.

2. Integration of multimodal data: Combining different types of data, such as visual, acoustic, and environmental variables, can enhance the accuracy of bird species prediction. Integrating these multimodal data sources and developing models that can effectively utilize them will provide a more holistic view of bird species identification and behavior.

3. Incorporating deep learning techniques: Deep learning algorithms, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have shown promise in various image and audio processing tasks. Applying these techniques to bird species prediction can help capture complex patterns and improve the accuracy of predictions.

4. Transfer learning and pre-training: Pre-training models on large-scale datasets, such as general image or audio datasets, and then fine-tuning them on bird-specific data can be beneficial. Transfer learning approaches enable models to leverage knowledge gained from related tasks and datasets, even when specific bird species data is limited. This approach can improve prediction performance, especially for rare or understudied species.

5. Incorporating citizen science data: Citizen science initiatives have been successful in engaging the public in scientific research, including bird observations. Integrating data collected by birdwatchers and citizen scientists into prediction models can increase the volume and diversity of data available, allowing for more accurate predictions across larger geographical areas.

6. Real-time monitoring: Developing real-time bird species prediction systems would enable on-the-fly identification and monitoring of bird species. This could be achieved by deploying portable devices or mobile applications that can instantly process and classify bird observations, aiding in conservation efforts and ecological research.

7. Collaboration and open data sharing: Collaboration among researchers, institutions, and organizations is crucial for advancing bird species prediction. Sharing datasets, models, and methodologies openly can foster innovation, allow for benchmarking, and promote reproducibility in the field.

8.Active learning and data annotation: Active learning techniques can optimize the process of data annotation by intelligently selecting the most informative samples for manual labeling. By actively involving human experts in the annotation process, prediction models can iteratively improve their performance with fewer labeled samples, reducing the overall annotation effort and increasing efficiency.

9.Fine-grained classification: While bird species prediction focuses on identifying the species of a bird, fine-grained classification goes a step further by distinguishing between different subspecies or individuals within a species. Developing models that can accurately classify fine-grained variations can provide valuable insights into population dynamics, genetic diversity, and individual behavior.

By focusing on these future enhancements, we can continue to improve the accuracy, efficiency, and applicability of bird species prediction models, leading to better conservation strategies, understanding of bird populations, and overall avian biodiversity management.

REFERENCES

1. https://snyk.io/advisor/python/torchvision/functions/torchvision.datasets.CIFAR10
2. https://www.kaggle.com/datasets/pytorch/resnet34
3. https://www.kaggle.com/code/shamiulislamshifat/playing-with-resnet50-full-tutorial
4. https://discuss.tensorflow.org/
5. https://stackoverflow.com/
6. https://huggingface.co/
7. https://www.tensorflow.org/versions/r2.1/api_docs/python/tf/keras/Model#fit
8. https://medium.com/towards-data-science/deep-learning-using-transfer-learning-python-code-for-resnet50-8acdfb3a2d38
9. https://datagen.tech/guides/computer-vision/resnet-50/
10. https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/ResNet50
11. https://viso.ai/deep-learning/resnet-residual-neural-network/
12. https://keras.io/api/applications/resnet/