VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

An Internship Report on

BIRD SPECIES CLASSIFICATION

Submitted in partial fulfillment of the requirements as a part of the internship for the award of degree of Bachelor of Engineering in Information Science and Engineering

Submitted by

RAGHAVAN V

1RN20IS114

Under the Guidance of

Ms. Kavitha B

Assistant Professor

Department of ISE

ESTD:2001

An Institute with a Difference

Department of Information Science and Engineering

RNS Institute of Technology

Channasandra, Dr. Vishnuvardhan Road, RR Nagar Post, Bengaluru – 560 098

2023 -2024

RNS Institute of Technology

Channasandra, Dr. Vishnuvardhan Road, RR Nagar Post,

Bengaluru – 560 098

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the internship project report entitled ***BIRD IMAGE CLASSIFICATION*** has been successfully completed by **RAGHAVAN V** bearing USN **1RN20IS114**, presently VIII semester students of **RNS Institute of Technology** in partial fulfillment of the requirements as a part of the ***AI/ML Internship*** for the award of the degree of ***Bachelor of Engineering in Information Science and Engineering*** under **Visvesvaraya Technological University, Belagavi** during academic year **2023 – 2024**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report and deposited in the departmental library. The internship project report has been approved as it satisfies the academic requirements as a part of Internship work.

Ms. Kavitha B	**Dr. Nirmal Kumar S Benni/ Ms. Aruna U**	**Dr. Suresh L**
Internship Guide	Internship Coordinators	Professor & HOD
Assistant Professor	Associate Professor/ Assistant Professor	Department of ISE
Department of ISE	Department of ISE	RNSIT

External Viva

Name of the Examiners	**Signature with date**
1. _____	_____
2. _____	_____

NASTECH

Microsoft Partner Network

CERTIPORT
A PEARSON VUE BUSINESS
AUTHORIZED TESTING CENTER

CERTIFICATE

This is to certify that **Raghavan V** bearing USN: **1RN20IS114** from RNSIT has taken part in internship training on the **Artificial Intelligence - Machine Learning & Data Science using Python & Azure** and successfully completed project work **"Birds Species Image Classification Using CNN"** in New Age Solutions Technologies (NASTECH) from August 2023 to September 2023.

Raghavan V has shown great enthusiasm and interest in the project. The incumbents' conduct and performance were found satisfactory during all phases of the project.

Thank you very much.

Sincerely Yours,

Deepak Garg

Founder

Address: Block No.10, Kailashnagar, Highland Residency, Kolshet Road, Thane (W)- 400608, Mumbai, Maharashtra.
Mobile Number: 8591132820| E-mail: deeprgarg@nastech.in

DECLARATION

I, **Raghavan V [USN: 1RN20IS114]**, student of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship project work entitled ***BIRD IMAGE CLASSIFIER*** has been carried out and submitted in partial fulfillment of the requirements for the *VIII Semester degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi* during academic year 2023-2024.

Place : Bengaluru
Date :

RAGHAVAN V
1RN20IS114

ABSTRACT

Bird species classification is a challenging task in the field of computer vision due to the vast diversity of bird species and the subtle variations in their appearances. Convolutional Neural Networks (CNNs) have proven to be highly effective in image classification tasks, and this paper presents an in-depth exploration of building a bird image classifier using CNNs. The primary objective is to develop a robust and accurate model capable of identifying various bird species from images.

This project provides a comprehensive guide to building a bird image classifier using CNNs, from data collection and preprocessing to model architecture design, training, validation, evaluation, and deployment. The results demonstrate the feasibility of using deep learning techniques for bird species identification, with potential applications in wildlife conservation, ornithology, and birdwatching.

ACKNOWLEDGMENT

The fulfillment and rapture that go with the fruitful finishing of any assignment would be inadequate without the specifying the people who made it conceivable, whose steady direction and support delegated the endeavors with success.

We would like to profoundly thank **Management** of **RNS Institute of Technology** for providing such a healthy environment to carry out this AI/ML Internship Project.

We would like to express our thanks to our Director **Dr. M K Venkatesha** for his support and for inspiring us towards the attainment of knowledge.

We would like to express our thanks to our Principal **Dr. Ramesh Babu H S** for his support and for inspiring us towards the attainment of knowledge.

We wish to place on record our words of gratitude to **Dr. Suresh L,** Professor and Head of the Department, Information Science and Engineering, for having accepted to patronize us in the right direction with all his wisdom.

We like to express our profound and cordial gratitude to our Guide**, Ms. Kavitha B,** Assistant Professor, Department of Information Science and Engineering for their valuable guidance, constructive comments, continuous encouragement throughout the Internship.

We like to express our profound and cordial gratitude to our Internship Project Coordinators**, Dr. Nirmalkumar S Benni,** Associate Professor and **Ms. Aruna U,** Assistant Professor Department of Information Science and Engineering for their valuable guidance, constructive comments, continuous encouragement throughout the Internship Work and guidance in preparing report.

We would like to thank all other teaching and non-teaching staff of Information Science & Engineering who have directly or indirectly helped us to carry out the Internship Project Work.

Also, we would like to acknowledge and thank our parents who are source of inspiration and instrumental in carrying out this Internship Project Work.

RAGHAVAN V

ACKNOWLEDGMENT 1RN20IS114

TABLE OF CONTENT

LIST OF FIGURES

Chapter 1

INTRODUCTION

Bird species classification is a challenging task in the field of computer vision due to the vast diversity of bird species and the subtle variations in their appearances. Convolutional Neural Networks (CNNs) have proven to be highly effective in image classification tasks, and this paper presents an in-depth exploration of building a bird image classifier using CNNs. The primary objective is to develop a robust and accurate model capable of identifying various bird species from images.

1.1 COMPANY PROFILE

NASTECH is formed with the purpose of bridging the gap between Academia and Industry Nastech is one of the leading Global Certification and Training service providers for technical and management programs for educational institutions. We collaborate with educational institutes to understand their requirements and form a strategy in consultation with all stakeholders to fulfill those by skilling, reskilling and upskilling the students and faculties on new age skills and technologies.

1.2 DOMAIN/TECHNOLOGY

The domain chosen for our project is AI/ML. Machine learning, the fundamental driver of AI, is possible through algorithms that can learn themselves from data and identify patterns to make predictions and achieve your predefined goals, rather than blindly following detailed programmed instructions, like in traditional computer programming. This technology allows the machine to perceive, learn, reason and communicate through observation of data, like a child that grows up and acquires knowledge from examples. Machines also have the advantage of not being limited by our inherent biological limitations. With machine learning, manufacturing companies have increased production capacity up to 20%, while lowering material consumption rates by 4%.

Nowadays, the revolutionary AI technology evolved from rule-based expert systems to machine learning and more advanced subcomponents such as deep learning (learning representations instead of tasks), artificial neural networks (inspired by animal brains) and reinforcement learning (virtual agents rewarded if they made good decisions).

1.3 DEPARTMENT

With the blessings of our beloved Chairman Dr. R N Shetty and the able guidance from Principal Dr. Ramesh Babu H S, the department of Information Science and Engineering has completed 20 years of academic excellence with a current intake of 180. The department is accredited by the NBA in the year 2011 and 2018. The department follows a quality procedure to prepare students to be industry ready and encourages them to pursue higher education. Department maintains higher academic standards with outcome based education, witnessing higher university results and ranks. More than 95% of students are placed in top companies with paid internships. The department is blessed with expertized teachers with an average experience of 14+ years. Industry interaction and Alumni connect is a hallmark of the department with the Centre of Excellence in Data Science and Cyber Security.

1.4 Problem Statement

The "Indian-Birds-Species-Image-Classification" consists of 8 bird species found in India, including Asian Green Bee-eater, Brown-Headed Barbet, Common Kingfisher, Coppersmith Barbet, Hoopoe, Indian Peacock, Indian Roller and Rufous Treepie .

- **Biodiversity Conservation**: Understanding and classifying bird species is crucial for conservation efforts. It helps researchers and conservationists identify endangered, threatened, or invasive species.
- **Ecological Research**: Bird species classification provides valuable insights into ecosystems. Different species play distinct roles in maintaining ecological balance.
- **Environmental Monitoring:** Birds can be sensitive indicators of environmental health. Changes in bird populations can signal environmental changes such as pollution, habitat loss, or climate change.

Chapter 2

REQUIREMENT ANALYSIS, TOOLS &TECHNOLOGIES

2.1 HARDWARE REQUIREMENT

Processor	Intel Core i7 processor
Processor Speed	2.30 GHz
RAM	16 GB
Storage Space	20 GB
Monitor Resolution	1024*768 or 1336*768 or 1280*1024

Table 2.1: Hardware Requirements

2.2 SOFTWARE REQUIREMENT

The software requirements are description of features and functionalities of the system.

Table 2.2 gives details of software requirements.

Operating System	Windows 11
IDE	Anaconda
Tools	Jupyter Notebook
Libraries	pandas, numpy , os, matplotlib, tensorflow, seaborn, keras, cv2, random, skLearn

Table 2.2: Software Requirements

2.2.1 Anaconda

Anaconda is the birthplace of Python data science. It is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package managementand deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded byPeter Wang and Travis Oliphant in 2012. Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open- source packages can be.

installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface.

2.2.2 Jupyter Notebook

Jupyter Notebook (formerly IPython Notebook) is a web-based interactive computational environment for creating notebook documents. Jupyter Notebook is built using several open- source libraries, including IPython, ZeroMQ, Tornado, jQuery, Bootstrap, and MathJax. A Jupyter Notebook document is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media. Underneath the interface, a notebook is a JSON document, following a versioned schema, usuallyending with the ".ipynb" extension. Jupyter Notebook is similar to the notebook interface of otherprograms such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

Chapter 3
DESIGN AND IMPLEMENTATION

3.1 PROBLEM STATEMENT

The following procedure shows the steps in data manipulation:

1.Data Collection and Preprocessing:

Data Collection: Gather a labeled dataset suitable for your task. The dataset should include images and their corresponding labels. The labels could be image categories, object classes, or any other classification target.

Data Preprocessing: Prepare the data by resizing all images to a consistent size (e.g., 224x224 pixels), normalizing pixel values (usually between 0 and 1 or -1 and 1), and performing any data augmentation if needed (e.g., random rotations, flips, and zooms). Split the dataset into training, validation, and test sets.

2. Model Architecture:

Define the architecture of your CNN. A CNN typically consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

Configure the number of layers, filter sizes, the number of filters in each layer, activation functions, and any regularization techniques (e.g., dropout or batch normalization).

3. Building the Model:

Use a deep learning framework like TensorFlow or PyTorch to build your CNN model. Create a neural network model object and add layers sequentially or using a functional API.

4. Compiling the Model:

Compile the model by specifying the loss function (categorical cross-entropy for classification tasks), the optimizer (e.g., Adam or SGD), and evaluation metrics (e.g., accuracy).

5. Training:

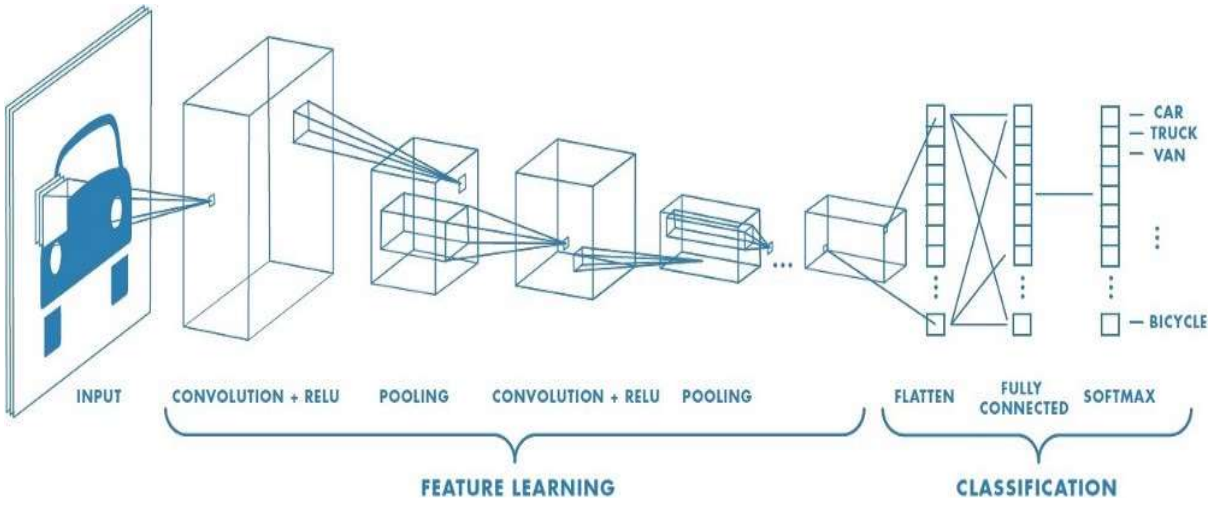
Train the CNN model using the training data. Use the training set for forward and backward passes, compute gradients, and update the model's weights using the chosen optimizer.

6. Evaluation:

After training, evaluate the model's performance on the test dataset, which contains data the model has never seen. Calculate and analyze performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices to assess the model's effectiveness.

3.2 Architecture :

Convolution Neural Network (CNN) :



CNNs are a specific type of ANN (Artificial Neural Network) designed for processing grid-like data, such as images and videos. They are characterized by their use of convolutional layers, pooling layers, and fully connected layers.

Steps Involved :

1. Data Collection:

- Assemble a diverse dataset of bird images, including various species and environmental conditions.

2. Image Preprocessing:

- Resize images to a consistent resolution suitable for the model.
- Normalize pixel values to a common scale (e.g., [0, 1]).

Augment the dataset with transformations like rotation, flipping, and zooming to increase model.

3. Image Splitting:

- Divide the dataset into training, validation, and test sets. A common split is 80% for training, 10% for validation, and 10% for testing.

4. Model Architecture:

- Design a CNN architecture suitable for image classification. Typical components include convolutional layers for feature extraction, pooling layers for spatial reduction, and fully connected layers for classification.
- Choose an appropriate activation function, loss function, and optimization algorithm.

5. Model Training:

a. Initialize the CNN model with random weights.
b. Feed batches of training images through the network.
c. Compute the loss between predicted and actual labels.
d. Use backpropagation to update model weights and minimize the loss.
e. Repeat this process for multiple epochs until the model converges.

6. Validation:

- Periodically evaluate the model on the validation set to monitor generalization performance.
- Adjust hyperparameters or modify the model architecture if necessary.

7. Hyperparameter Tuning:

- Fine-tune hyperparameters, including learning rate, batch size, and regularization, to optimize model performance.

8. Testing:

- Assess the final model on the test set to estimate its performance on unseen data.

9. Performance Evaluation:

- Analyze metrics such as accuracy, precision, recall, and F1 score to gauge the model's classification performance.

10. Model Deployment:

- Integrate the trained model into applications or systems for real-world use.
- Optimize the model for inference on target platforms, considering factors like computational resources and response time requirements.

11. Monitoring and Maintenance:

- Continuously monitor the model's performance in real-world scenarios.
- Periodically retrain the model with new data to adapt to changing conditions and ensure ongoing accuracy.

These steps provide a general framework for building a bird image classification system using CNNs. Adjustments may be needed based on the specifics of the dataset and the desired application.

3.3 Libraries

Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with" relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.

- Easy handling of missing data (represented as NaN, NA, or NaT) in floating point as well as non-floating point data

- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for user in computations.
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data.
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into Data Frame objects
- Intelligent label-based slicing, fancy indexing, and sub setting of large data sets.
- Intuitive merging and joining data sets.
- Flexible reshaping and pivoting of data sets.
- Hierarchical labeling of axes (possible to have multiple labels per tick).

Numpy

NumPy, which stands for Numerical Python, is a powerful library in Python for numerical and mathematical operations. Here's a brief overview of its features:

- **Multidimensional Arrays:** NumPy provides a powerful **array** object that represents a multidimensional, homogeneous array. This array is the fundamental building block for various numerical computations.
- **Mathematical Functions:** NumPy includes a wide range of mathematical functions that operate on entire arrays without the need for explicit looping
- **Efficiency:** NumPy operations are implemented in C and Fortran, making them highly efficient. It also provides tools for integrating with code written in other languages.
- **Memory Management:** NumPy provides tools for efficiently storing and accessing large arrays of data. It includes features for memory-mapped files, which allow efficient storage and retrieval of array data on disk.

SeaBorn

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables.

- Built in themes for styling matplotlib graphics.
- Visualizing univariate and bivariate data.
- Fitting in and visualizing linear regression models.
- Seaborn works well with NumPy and Pandas data structures.

It comes with built in themes for styling Matplotlib graphics

Matplot_lib

Matplotlib is a low-level graph plotting library in python that serves as a visualization utility. It was created by John D. Hunter. It is open source, and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and JavaScript for Platform compatibility.

- Creates publication quality plots.
- Makes interactive figures that can zoom, pan, update.
- Customizes visual style and lausert.
- Export to many file formats.
- Can be embedded in JupyterLab and Graphical User Interfaces.
- Uses a rich array of third-partypackages built on Matplotlib.

TensorFlow

TensorFlow, developed by Google, is a versatile open-source machine learning framework that

serves as a comprehensive tool for a wide array of tasks, including deep learning, neural networks, and numerical computations. The framework operates on the principle of symbolic computational graphs, where models.are represented as such graphs, allowing for efficient execution and optimization. TensorFlow provides ease of use through its support for high-level APIs like Keras, catering to beginners, while also offering lower-level APIs for users seeking more flexibility and customization in their machine learning models.

Cross-platform compatibility is a notable feature of TensorFlow, as it is designed to run seamlessly on various hardware configurations, including CPUs, GPUs, and TPUs (Tensor Processing Units), ensuring scalability and performance optimization. Within its ecosystem, TensorFlow offers a rich set of tools and libraries for tasks such as data preprocessing, model deployment, and visualization. TensorBoard, one of its integrated tools, stands out as a powerful visualization platform for monitoring and debugging models throughout the development process.

TensorFlow's prominence in deep learning is underscored by its robust support for neural networks. It facilitates the creation and training of diverse neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers. Moreover, TensorFlow extends its utility to resource-constrained platforms through TensorFlow Lite, a lightweight version tailored for mobile and edge devices, enabling the deployment of machine learning models in these environments.

For end-to-end machine learning workflows, TensorFlow Extended (TFX) provides a comprehensive platform that covers the entire machine learning lifecycle. From data validation to training, serving, and monitoring, TFX ensures a streamlined process for deploying production-ready machine learning models. The framework's active community contributes to its continuous

improvement, and extensive documentation, tutorials, and resources are available, fostering an environment conducive to learning and development

3.4 Code Snippet

Import libraries

```
import numpy as np
import os
import pandas as pd
from matplotlib import pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
import cv2
import random
from tensorflow.keras.optimizers import Adam,SGD
from sklearn.metrics import confusion_matrix, classification_report
```

- **Load the image**

```
data_path = r"C:\Users\ragha\AIML Python\reduced bird image data\train\Common-Kingfisher\
Common-Kingfisher_765.jpg"
image = cv2.imread(data_path)
cv2.imread(data_path)
```

- **Training and Validation**

```
train = ImageDataGenerator(rescale=1/255)
valid = ImageDataGenerator(rescale=1/255)
train_dataset = train.flow_from_directory(r'C:\Users\ragha\AIML Python\reduced bird image data\
train',
                                          target_size=(256,256),
                                          color_mode='rgb',
                                          class_mode='categorical',
                                          batch_size=BATCH_SIZE,
                                          shuffle=True)
```

```
valid_dataset = valid.flow_from_directory(r'C:\Users\ragha\AIML)
                    target_size=(256,256),
                    color_mode='rgb',
                    class_mode='categorical',
                     batch_size=BATCH_SIZE,
                    shuffle=True)
```

➢ Label and Mapping

```
labels = {value: key for key, value in train_dataset.class_indices.items()}

    print("Label Mappings for classes present in the training and     validation datasets\n")
for key, value in labels.items():
    print(f"{key} : {value}")
```


Classification

Label Mappings for classes present in the training and validation datasets

0 : Asian-Green-Bee-Eater
1 : Brown-Headed-Barbet
2 : Common-Kingfisher
3 : Coppersmith-Barbet
4 : Hoopoe
5 : Indian-Peacock
6 : Indian-Roller
7 : Rufous-Treepie

➢ Mini Batch Gradient descent model

```
model2 = tf.keras.models.Sequential
([
tf.keras.layers.Conv2D(128,(3,3), activation = 'relu',input_shape =(256,256,3)),
tf.keras.layers.MaxPool2D(2,2),
 tf.keras.layers.BatchNormalization(),
##
tf.keras.layers.Conv2D(64,(3, 3), activation='relu'),
 tf.keras.layers.MaxPool2D(2,2),
tf.keras.layers.BatchNormalization(),
##
tf.keras.layers.Conv2D(32,(3, 3), activation='relu'),
tf.keras.layers.MaxPool2D(2,2),
tf.keras.layers.BatchNormalization(),
##
##
tf.keras.layers.Flatten(),
##
tf.keras.layers.Dense(256, activation='relu'),
tf.keras.layers.Dropout(0.5),
tf.keras.layers.Dense(8, activation='softmax'
])
```


optimizer = Adam(learning_rate=0.001)

```
model2.compile(loss = 'categorical_crossentropy',
optimizer = optimizer,
metrics = ['accuracy'])
```

- **Model Fitting**

```
model2_fit = model2.fit(
train_dataset,
epochs = 10,
batch_size = 34 ,
validation_data=valid_dataset)
```



```
Epoch 1/10
150/150 [==============================] - 370s 2s/step - loss: 7.5438 - accuracy: 0.3704 -
val_loss: 6.1856 - val_accuracy: 0.1053
Epoch 2/10
150/150 [==============================] - 369s 2s/step - loss: 2.6936 - accuracy: 0.5280 -
val_loss: 9.7825 - val_accuracy: 0.1973
Epoch 3/10
150/150 [==============================] - 368s 2s/step - loss: 1.2989 - accuracy: 0.6267 -
val_loss: 1.8677 - val_accuracy: 0.3433
Epoch 4/10
150/150 [==============================] - 369s 2s/step - loss: 0.8165 - accuracy: 0.7359 -
val_loss: 1.6311 - val_accuracy: 0.5320
Epoch 5/10
150/150 [==============================] - 369s 2s/step - loss: 0.6691 - accuracy: 0.7802 -
val_loss: 1.7864 - val_accuracy: 0.5900
Epoch 6/10
150/150 [==============================] - 367s 2s/step - loss: 0.4908 - accuracy: 0.8341 -
val_loss: 1.7801 - val_accuracy: 0.6213
Epoch 7/10
150/150 [==============================] - 367s 2s/step - loss: 0.4846 - accuracy: 0.8412 -
val_loss: 2.2733 - val_accuracy: 0.5367
Epoch 8/10
150/150 [==============================] - 371s 2s/step - loss: 0.3757 - accuracy: 0.8749 -
val_loss: 1.9912 - val_accuracy: 0.6400
Epoch 9/10
150/150 [==============================] - 368s 2s/step - loss: 0.3403 - accuracy: 0.8890 -
val_loss: 1.7403 - val_accuracy: 0.6513
Epoch 10/10
150/150 [==============================] - 367s 2s/step - loss: 0.3025 - accuracy: 0.9004 -
val_loss: 2.0036 - val_accuracy: 0.6287
```

- **Model Prediction**

```
pred = model2.predict(test_dataset)
```

```
fig, ax = plt.subplots(nrows=2, ncols=5, figsize=(12, 10))
idx = 0

for i in range(2):
    for j in range(5):
        predicted_label = labels[np.argmax(predictions[idx])]
        ax[i, j].set_title(f"{predicted_label}")
        ax[i, j].imshow(test_dataset[0][0][idx])
        ax[i, j].axis("off")
        idx += 1

plt.tight_layout()
plt.suptitle("Test Dataset Predictions", fontsize=20)
plt.show()

test_loss, test_accuracy = model2.evaluate(test_dataset , batch_size =200)
```



```
print(f"Test Loss:     {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

➢ Plotting the Classification Matrix

```
y_pred = np.argmax(predictions, axis=1)
y_true = test_dataset.classes

cf_mtx = confusion_matrix(y_true, y_pred)

group_counts = ["{0:0.0f}".format(value) for value in cf_mtx.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in cf_mtx.flatten()/np.sum(cf_mtx)]
box_labels = [f"{v1}\n({v2})" for v1, v2 in zip(group_counts, group_percentages)]
box_labels = np.asarray(box_labels).reshape(8, 8)

plt.figure(figsize = (12, 10))
sns.heatmap(cf_mtx, xticklabels=labels.values(), yticklabels=labels.values(),
   cmap="YlGnBu", fmt="", annot=box_labels)
plt.xlabel('Predicted Classes')
plt.ylabel('True Classes')
plt.show()
```

Chapter 4

OBSERVATIONS AND RESULTS

4.1 Results and snapshots

The below images shows the visuals of the bird image classification process step by step

4.1.1 Image Plotting

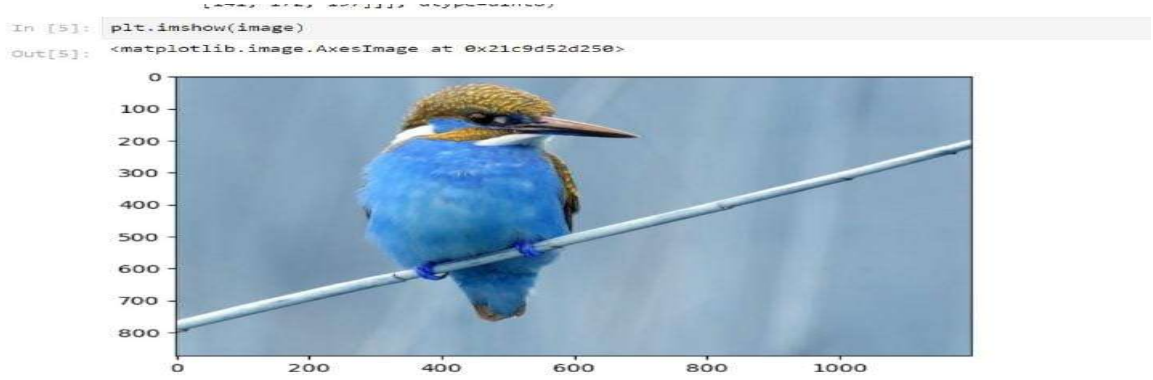


Fig 4.1 Sample image plotting

4.1.2 Epoch Running

```
In [40]: model_fit = model.fit(
             train_dataset,
             epochs = 10 ,
             batch_size = 1,
             validation_data=valid_dataset)

Epoch 1/10
5100/5100 [==============================] - 852s 167ms/step - loss: 2.0569 - accuracy: 0.1708 - val_loss: 18.8793 - val_accuracy: 0.2420
Epoch 2/10
5100/5100 [==============================] - 845s 166ms/step - loss: 2.0044 - accuracy: 0.2098 - val_loss: 16.5226 - val_accuracy: 0.2780
Epoch 3/10
5100/5100 [==============================] - 851s 167ms/step - loss: 1.9263 - accuracy: 0.2231 - val_loss: 12.7601 - val_accuracy: 0.2773
Epoch 4/10
5100/5100 [==============================] - 864s 169ms/step - loss: 1.8893 - accuracy: 0.2275 - val_loss: 6.5716 - val_accuracy: 0.2847
Epoch 5/10
5100/5100 [==============================] - 869s 170ms/step - loss: 1.8814 - accuracy: 0.2286 - val_loss: 12.3329 - val_accuracy: 0.2753
Epoch 6/10
5100/5100 [==============================] - 877s 172ms/step - loss: 1.8793 - accuracy: 0.2394 - val_loss: 20.5774 - val_accuracy: 0.2793
Epoch 7/10
5100/5100 [==============================] - 866s 170ms/step - loss: 1.8396 - accuracy: 0.2412 - val_loss: 24.0551 - val_accuracy: 0.2680
Epoch 8/10
5100/5100 [==============================] - 862s 169ms/step - loss: 1.8505 - accuracy: 0.2402 - val_loss: 9.6119 - val_accuracy: 0.2847
Epoch 9/10
5100/5100 [==============================] - 863s 169ms/step - loss: 1.8082 - accuracy: 0.2455 - val_loss: 13.7061 - val_accuracy: 0.2827
Epoch 10/10
5100/5100 [==============================] - 862s 169ms/step - loss: 1.8866 - accuracy: 0.2427 - val_loss: 20.3183 - val_accuracy: 0.2807
```

Fig 4.2 Epochs

4.1.3 Predictions

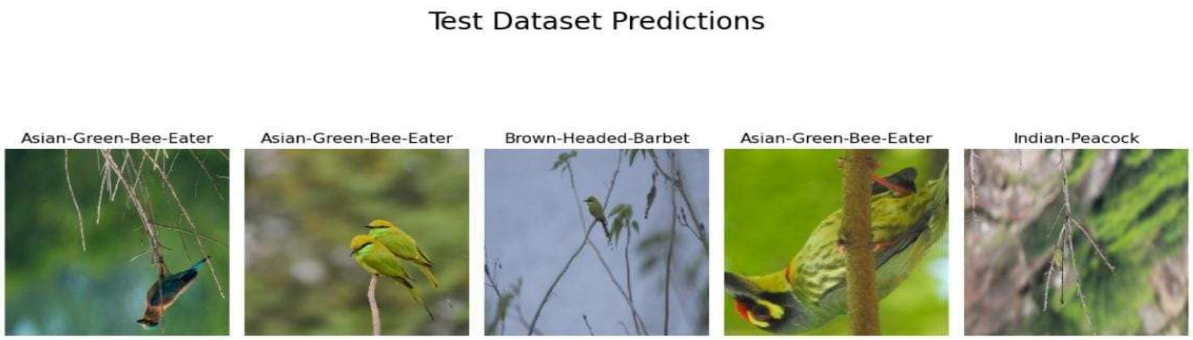


Fig 4.3 Predictions

4.1.4 Sample training images

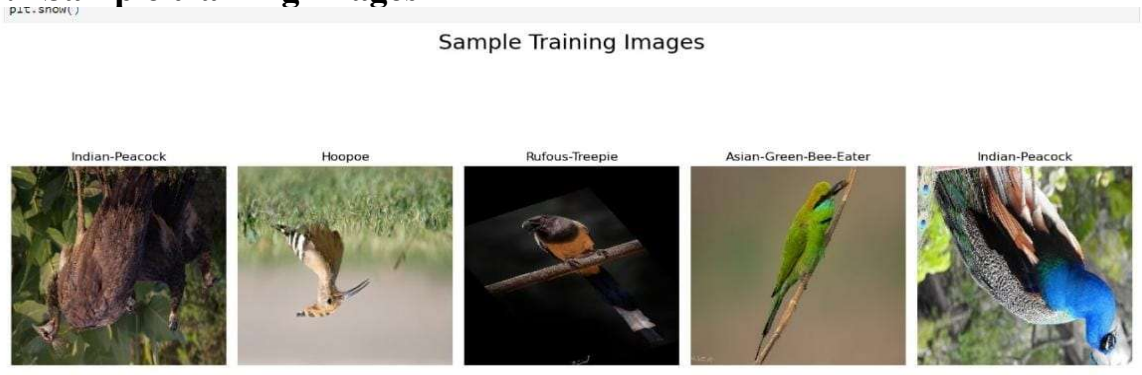


Fig 4.4 Sample training images

4.1.5 Some Wrong Prediction

Fig 4.5 Wrong Predictions

4.1.6 Stochastic Gradient Descendent –(SGD)

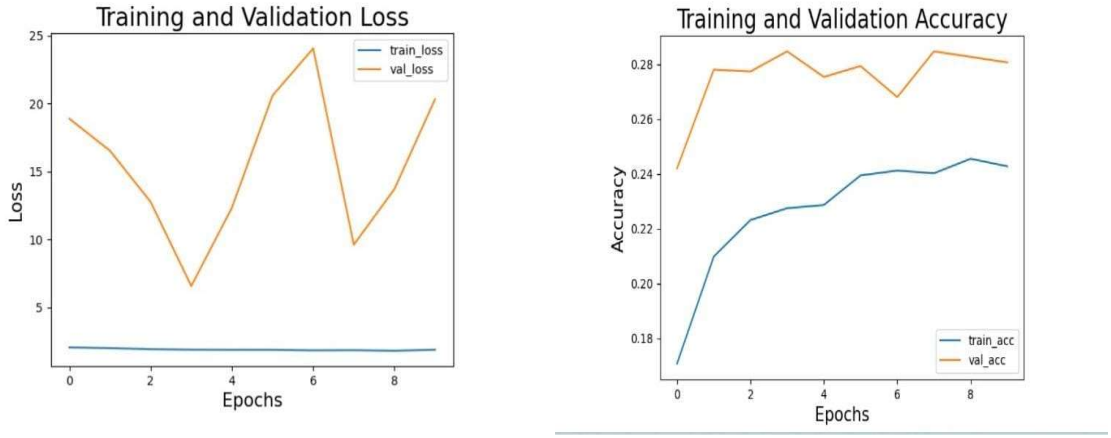


Fig 4.6 Graphs for Stochastic Gradient Descendent

4.1.7 Mini-Batch Gradient Descendent

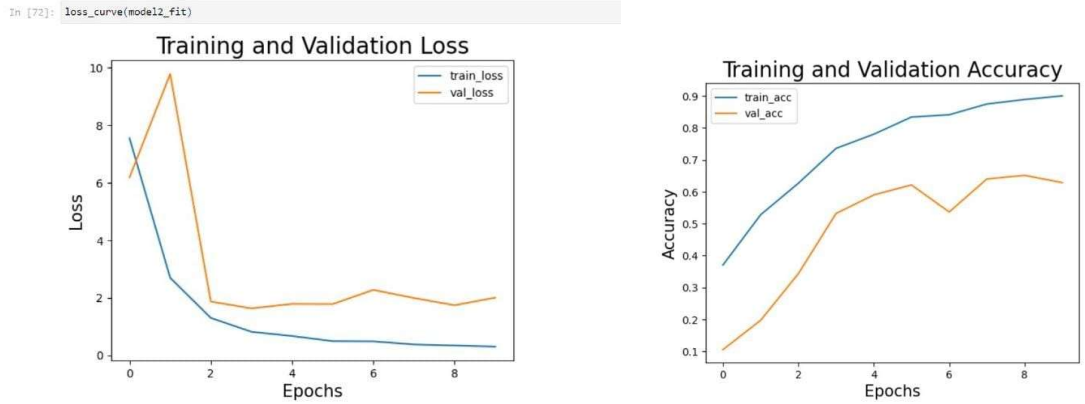


Fig 4.7 Graphs for Mini-Batch Gradient Descendent

4.1.8 Heatmap

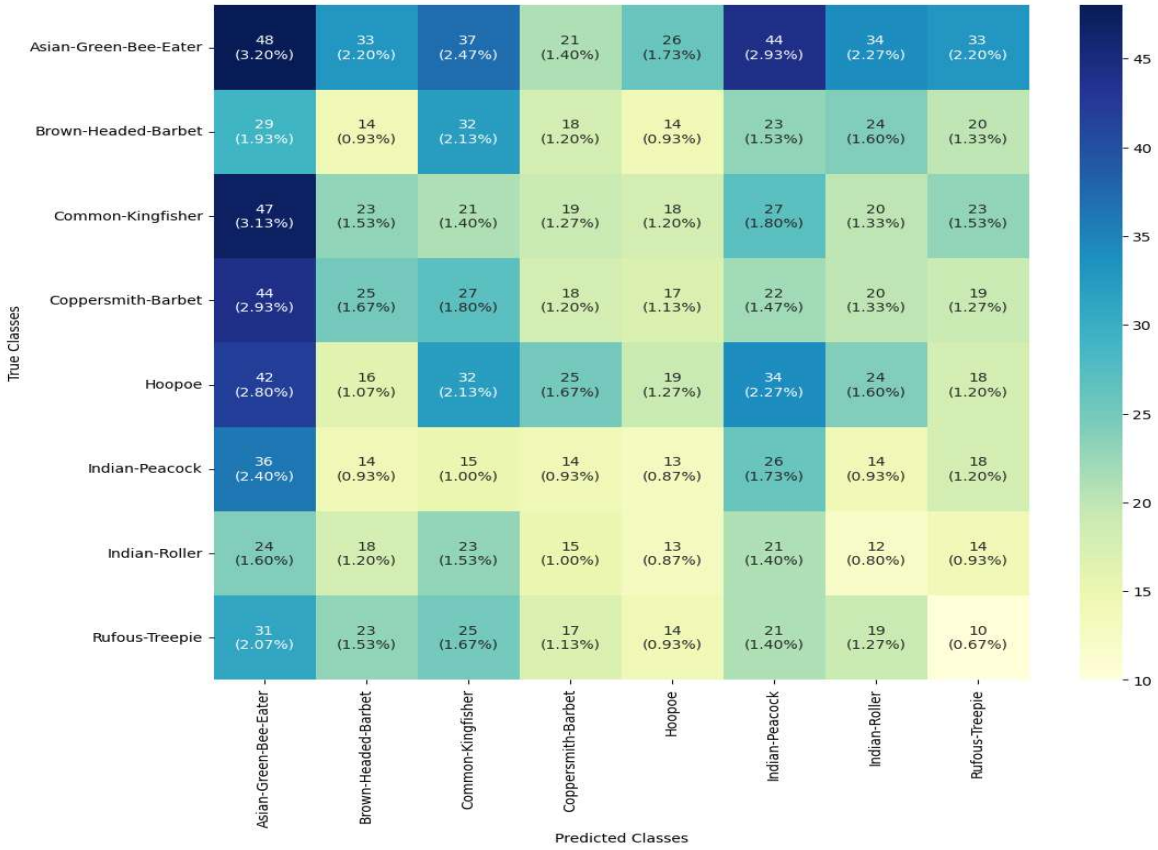


Fig 4.8 Heatmap for Predictions

Chapter 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1 Conclusion

- In conclusion, the developed Convolutional Neural Network (CNN) for bird image classification demonstrates promising accuracy and generalization capabilities.
- The model's performance suggests its potential application in real-world scenarios, such as wildlife monitoring and ornithological studies. While the current results are

encouraging, further enhancements could be explored, including fine-tuning the model, leveraging transfer learning, and expanding the dataset for improved robustness in diverse environments.

- In summary, the bird image classification using Mini Batch Gradient Descent (MGD) yields notable results. The iterative optimization process of MGD effectively refines model parameters, enhancing accuracy.
- The approach showcases potential for broader applications, with room for future refinement through parameter tuning and exploration of advanced optimization techniques. Future work could delve into further optimization strategies and explore the synergy between MGD and other advanced optimization techniques to maximize classification performance.
- Discuss ethical considerations related to the use of AI in wildlife monitoring, emphasizing the importance of avoiding biases in the dataset and being mindful of the potential impact on the studied bird populations.

5.2 Future Enhancements

- **User-Friendly Interfaces**: Build user-friendly interfaces or applications that allow non-experts, such as field researchers, to easily deploy and use the bird image classification system, fostering broader adoption and practical application.
- **Robustness to Environmental Changes**: Train the model to be robust to variations in environmental conditions, such as different lighting, weather, and backgrounds, to enhance its applicability across diverse ecosystems.
- **Fine-tuning Hyperparameters**: Conduct an extensive hyperparameter search to fine-tune the model's architecture, learning rates, and regularization parameters for optimal performance on the specific bird image dataset.
- **Active Learning**: Develop strategies for actively selecting and labeling informative samples from unlabeled data, optimizing the model's performance with less manual annotation effort.
- **Data Augmentation:** Implement data augmentation techniques to artificially increase the size of your dataset. This can help improve model generalization by exposing it to a wider variety of bird image variations, such as rotations, flips, and zooms

REFERENCES

[1].International Journal of Engineering Research & Technology (IJERT) http://www.ijert.org ISSN: 2278-0181 IJERTV9IS060279 (This work is licensed under a Creative Commons Attribution 4.0 International License.) Published by : www.ijert.org Vol. 9 Issue 06, June-2020

[2]. 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution(CC BY) license (http://creativecommons.org/licenses/by/4.0/).

[3]. Article submitted 2022-01-20. Resubmitted 2022-03-08. Final acceptance 2022-03-08. Final version published as submitted by the authors.- https://doi.org/10.3991/ijoe.v18i07.29639

[4]. Coding Lane - https://youtu.be/E5Z7FQp7AQQ?feature=shared