BIRD SPECIES CLASSIFICATION USING DEEP LEARNING

A Project Report submitted in the partial fulfillment of the requirements for the award of the degree.

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

N.Bhargava Raju (19471A05N3)

Ch.Shanmukha Chakravarthy (19471A05K6)

D.Balaji (19471A05K9)

Under the esteemed guidance of

SK.Rafi M.Tech.,(Ph.D)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

NARASARAOPET

(Affiliated to J.N.T.U, Kakinada , Approved by AICTE & Accredited by NBA)

(2022-2023)

NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPETA

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "BIRDS SPECIES CLASSIFICATION USING DEEP LEARNING" is a bonafide work done by "N.Bhargava Raju (19471A05N3), Ch.Shanmukha Chakravarthy (19471A05K6), D.Balaji (19471A05K9) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2022-2023.

PROJECT GUIDE	PROJECT CO-ORDINATOR
SK.Rafi M.Tech.,(Ph.D)	**M.Sireesha M.Tech.,(Ph.D)**
Asst. Prof.	**Assoc. Prof.**

HEAD OF THE DEPARTMENT	EXTERNAL EXAMINER
Dr. S. N. TirumalaRao, M.Tech.,Ph.D.	
Professor	

ACKNOWLEDGEMENT

We wish to express my thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sir **M.V.Koteswara Rao,B.sc** who took keen interest in me in every effort throughout this course.We owe out gratitude to our honoured principal **Dr.M.Sreenivasa Kumar,*M.Tech.,Ph.D(UK).,MISTE.,FIE(I)*** for his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude to **Dr.S.N.Tirumala Rao,M.Tech, Ph.D** H.O.D. CSE department and our guide **SK.Rafi M.Tech.,(Ph.D).** Assoc.Professor of CSE department whose valuable guidance and unstinting encouragement enable me to accomplish our project successfully in time.

We extend our sincere thanks to **Mrs.M.Sireesha,M.Tech.,(Ph.D).**Assoc. Professor & coordinator of the project for extending her encouragement. Her profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff of our department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we receive from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions and clarifying out doubts which had really helped us in successfully completing our project.

By

N.Bhargava Raju (19471A05N3)
Ch.Shanmukha Chakravarthy (19471A05K6)
D.Balaji (19471A05K9)

NARASARAOPETA
NEC ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research.

M2: Build a passionate and a determined team of faculty with student centric teaching,imbibing experiential, innovative skills.

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems.

NARASARAOPETA
NEC ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a centre of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertize in modern software tools and technologies to cater to the real time requirements of the Industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO's)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project Course Outcomes (CO'S):

CO425.1: Analyse the System of Examinations and identify the problem.

CO425.2: Identify and classify the requirements.

CO425.3: Review the Related Literature**.**

CO425.4: Design and Modularize the project.

CO425.5: Construct, Integrate, Test and Implement the Project.

CO425.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C425.1		✓											✓		
C425.2	✓		✓		✓								✓		
C425.3				✓		✓	✓	✓					✓		
C425.4			✓			✓	✓	✓					✓	✓	
C425.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C425.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C425.1	2	3											2		
C425.2			2		3								2		
C425.3				2		2	3	3					2		
C425.4			2			1	1	2					3	2	
C425.5					3	3	3	2	3	2	2	1	3	2	1
C425.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO's and PO's:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO's:

Name of the course from which principles are applied in this project	Description of the device	Attained PO
C3.2.4, C3.2.5	Gathering the requirements and defining the problem, plan to develop a smart bottle for health care using sensors.	PO1, PO3
CC4.2.5	Each and every requirement is critically analyzed, the process model is identified and divided into five modules	PO2, PO3
CC4.2.5	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC4.2.5	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC4.2.5	Documentation is done by all our four members in the form of a group	PO10
CC4.2.5	Each and every phase of the work in group is presented periodically	PO10, PO11
CC4.2.5	Implementation is done and the project will be handled by the hospital management and in future updates in our project can be done based on air bubbles occurring in liquid in saline.	PO4, PO7
CC4.2.8 CC4.2.	The physical design includes hardware components like sensors, gsm module, software and Arduino.	PO5, PO6

ABSTRACT

Nowadays, Birdwatching is a common hobby but to identify their species requires the assistance of bird books. To provide birdwatchers a handy tool to admire the beauty of birds, we developed a deep learning platform to assist users in recognizing 27 species of birds endemic to Taiwan using a mobile app named the Internet of Birds (IoB). Bird images were learned by a convolutional neural network (CNN) to localize prominent features in the images. First, we established and generated a bounded region of interest to refine the shapes and colors of the object granularities and subsequently balanced the distribution of bird species. Then, a skip connection method was used to linearly combine the outputs of the previous and current layers to improve feature extraction.

Finally, we applied the softmax function to obtain a probability distribution of bird features. The learned parameters of bird features were used to identify pictures uploaded by mobile users. The proposed CNN model with skip connections achieved higher accuracy of 99.00 % compared with the 93.98% from a CNN and 89.00% from the SVM for the training images. As for the test dataset, the average sensitivity, specificity, and accuracy were 93.79%, 96.11%, and 95.37%, respectively

***Key Words*: Convolution neural network, Artificial Intelligence, Machine Learning, Image Classification.**

INDEX

1.Introduction

Introduction

Numerous people visit bird sanctuaries to see various birds and to enjoy the beautiful variants of colors and characteristics of the birds, People barely have the knowledge about the various species and hence cannot easily distinguish the characteristics and the species name without the expertise in the field of ornithology. Bird watching is usually considered to be a good recreational activity that most of the people do practice besides their usual life style. The automatic identification and classification of birds by making use of the modern artificial intelligence and machine learning motivates the development of the proposed model. Deep Learning is a subset of machine learning comprising of various algorithms and was inspired by the human neural networks, the algorithms imitates the working of human brains in processing of the data and produces a pattern of data for decision making. This paper presents an approach of convolution neural network model for identifying the species of birds.

CNN is commonly applied for analyzing the visual imagery and image Classification usually refers to taking an input of an image and classifying it to some particular class, here in we have presented a CNN based classification model which classifies the bird species given a bird image as input. The convolution neural network model is capable of extracting the variant features based on size, shape and color from the images and are hence capable of successful classification.

Existing System

Evaluating image captions is very challenging partially due to the fact that there are multiple correct captions for every single image. Most of the existing one-to-one metrics operate by penalizing mismatches between reference and generative caption without considering the intrinsic variance between ground truth captions. It usually leads to over-penalization and thus a bad correlation to human judgment. Recently, the latest one-to-one metric BERTScore can achieve high human correlation in system-level tasks while some issues can be fixed for better performance. we propose a novel metric based on BERTScore that could handle such a challenge and extend BERTScore with a few new features appropriately for image captioning evaluation. The experimental results show that our metric achieves state-of-the-art human judgment correlation

Disadvantages:

1. Computation time is very high.
2. Generates not effective results.
3. Manual work is more.

Proposed System

We proposed to develop a system which will help to generate captions. In Artificial Intelligence (AI), the contents of an image are generated automatically which involves computer vision and NLP (Natural Language Processing). The neural node which is regenerative,is created .Generally human beings can caption the image by visualizing the image.As same as the machine can also can caption the image by usong some deeplearning and NLP techniques. It depends on computer vision and machine translation.This model consists of Convolution Neural Network(CNN) as well as Recurrent Neural Network(RNN) &Long ShortTerm Memory(LSTM).The Flickr8k dataset has become a standard benchmark for sentence based image description associated with manually annotated bounding boxes.

Advantages:

1. Generates accurate and efficient results.
2. Computation time is greatly reduced.
3. Automatically caption is generated.

System Requirements

Hardware Requirements:

- CPU: 8th Generation IntelÂ® Core i7 Processor
- RAM: 8 GB or above
- Hard Disk: 500 GB or above
- GPU: NVIDIA GTX 960

Software Requirements:

- Operating System: Windows 10
- IDE: Atom Editor
- Language: Python v3.8.2
- Backend Libraries: Keras, TensorFlow, NumPy, Scikit-Learn, OpenCV
- Frontend: HTML5, CSS3, JavaScript

2. LITERATURE SURVEY

Deep Learning

Deep learning is an artificial intelligence(AI) function that imitates the working of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. **Deep Learning** is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**.

Deep Learning Methods

Deep learning uses artificial neural networks to perform sophisticated computations on large amounts of data. It is a type of machine learning that works based on the structure and function of the human brain. Deep learning algorithms train machines by learning from examples. Industries such as health care, ecommerce, entertainment, and advertising commonly use deep learning. There are many models available in Deep learning ,model selection plays a vital role. Few examples are: LSTM,RNN,CNN etc. Deep learning has evolved over the past five years, and deep learning algorithms have become widely popular in many industries. Deep learning algorithms train machines by learning from examples. Industries such as health care, eCommerce, entertainment, and advertising commonly use deep learning.

While deep learning algorithms feature self-learning representations, they depend upon ANNs that mirror the way the brain computes information. During the training process, algorithms use unknown elements in the input distribution to extract features, group objects, and discover useful data patterns. Much like training machines for self-learning, this occurs at multiple levels, using the algorithms to build the models.

Deep learning models make use of several algorithms. While no one network is considered perfect, some algorithms are better suited to perform specific tasks. To choose the right ones, it’s good to gain a solid understanding of all primary algorithms.

Defining Neural Networks

A neural network is structured like the human brain and consists of artificial neurons, also known as nodes. These nodes are stacked next to each other in three layers:

- The input layer
- The hidden layer(s)
- The output layer

Data provides each node with information in the form of inputs. The node multiplies the inputs with random weights, calculates them, and adds a bias. Finally, nonlinear functions, also known as activation functions, are applied to determine which neuron to fire.

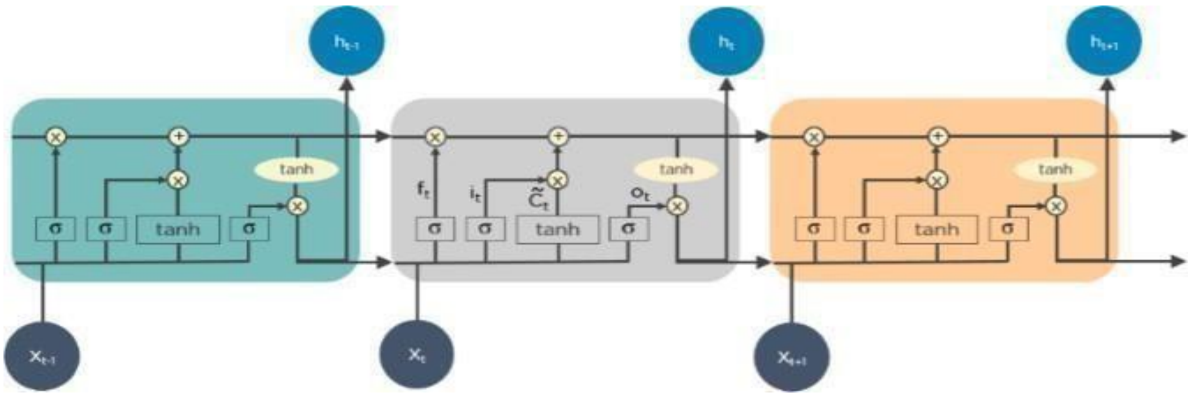
LSTM:

Long Short Term Memory Networks are a type of Recurrent Neural Network (RNN) that can learn and memorize long-term dependencies. Recalling past information for long periods is the default behavior.

LSTMs retain information over time. They are useful in time-series prediction because they remember previous inputs. LSTMs have a chain-like structure where four interacting layers communicate in a unique way. Besides time-series predictions, LSTMs are typically used for speech recognition, music composition, and pharmaceutical development.

How Do LSTMs Work?

- First, they forget irrelevant parts of the previous state
- Next, they selectively update the cell-state values
- Finally, the output of certain parts of the cell state



LSTM

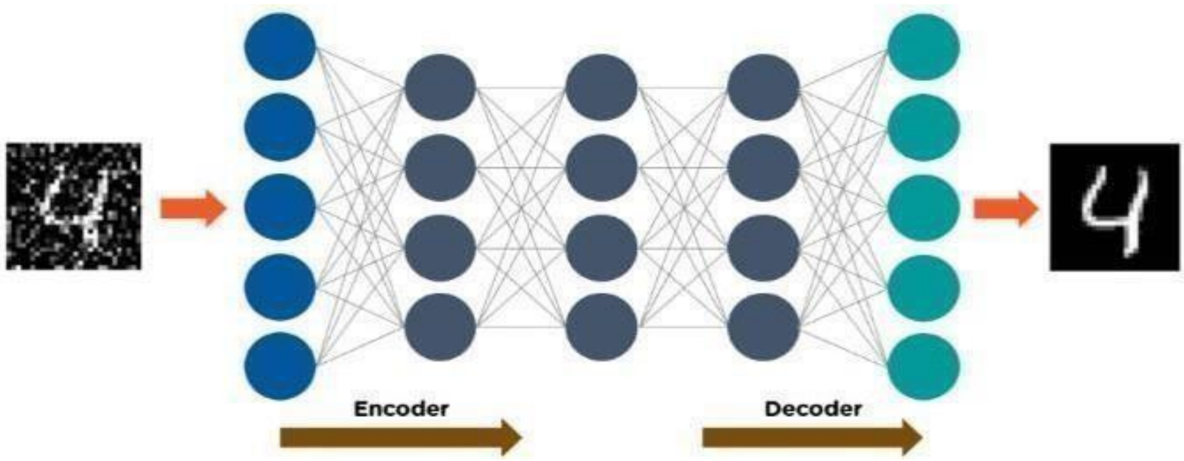
Autoencoders:

Autoencoders are a specific type of feed forward neural network in which the input and output are identical. Geoffrey Hinton designed autoencoders in the 1980s to solve unsupervised learning problems. They are trained neural networks that replicate the data from the input layer to the output layer. Autoencoders are used for purposes such as pharmaceutical discovery, popularity prediction, and image processing.

How Do Autoencoders Work?

An autoencoder consists of three main components: the encoder, the code, and the decoder.

- Autoencoders are structured to receive an input and transform it into a different representation. They then attempt to reconstruct the original input as accurately as possible.
- When an image of a digit is not clearly visible, it feeds to an autoencoder neural network.
- Autoencoders first encode the image, then reduce the size of the input into a smaller representation. Finally, the autoencoder decodes the image to generate the reconstructed image.



Autoencoders

Restricted Boltzmann Machines (RBMs):

RBMs are stochastic neural networks that can learn from a probability distribution over a set of inputs.

RBMs consist of two layers:

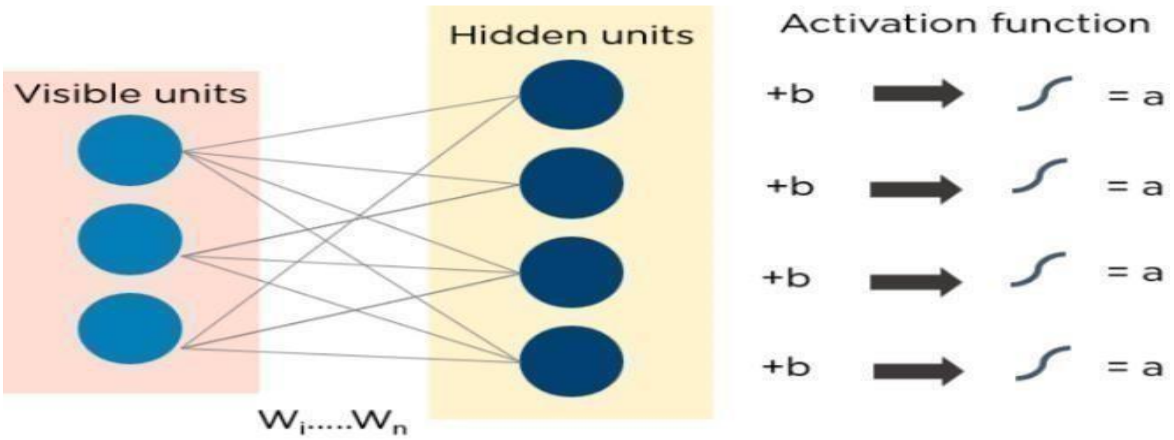
- Visible units
- Hidden units

Each visible unit is connected to all hidden units. RBMs have a bias unit that is connected to all the visible units and the hidden units, and they have no output nodes.

How Do RBMs Work?

RBMs have two phases: forward pass and backward pass.

- RBMs accept the inputs and translate them into a set of numbers that encodes the inputs in the forward pass.
- RBMs combine every input with individual weight and one overall bias. The algorithm passes the output to the hidden layer.
- In the backward pass, RBMs take that set of numbers and translate them to form the reconstructed inputs.
- RBMs combine each activation with individual weight and overall bias and pass the output to the visible layer for reconstruction.
- At the visible layer, the RBM compares the reconstruction with the original input to analyze the quality of the result.



RBM

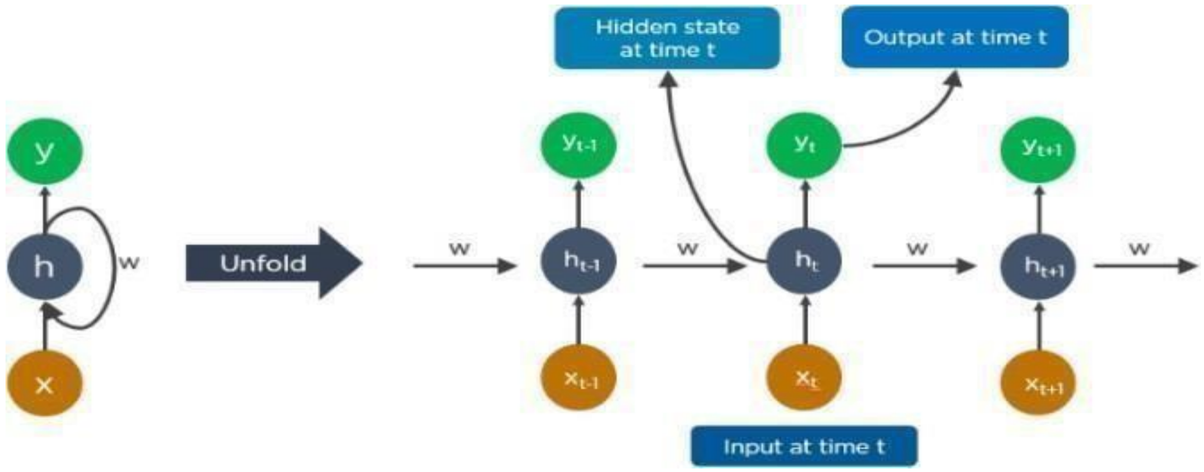
Recurrent Neural Networks (RNNs):

RNNs have connections that form directed cycles, which allow the outputs from the LSTM to be fed as inputs to the current phase.

The output from the LSTM becomes an input to the current phase and can memorize previous inputs due to its internal memory. RNNs are commonly used for image captioning, time-series analysis, natural-language processing, handwriting recognition, and machine translation.

How Do RNNs work?

- The output at time t-1 feeds into the input at time t.
- Similarly, the output at time t feeds into the input at time t+1.
- RNNs can process inputs of any length.
- The computation accounts for historical information, and the model size does not increase with the input size.



RNN

CNN:

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one from the other. Sequential model in CNN allows to build a model layer by layer. There are three types of layers in a convolutional neural network: convolutional layer, pooling layer, and fully connected layer.

Convolutional layer:

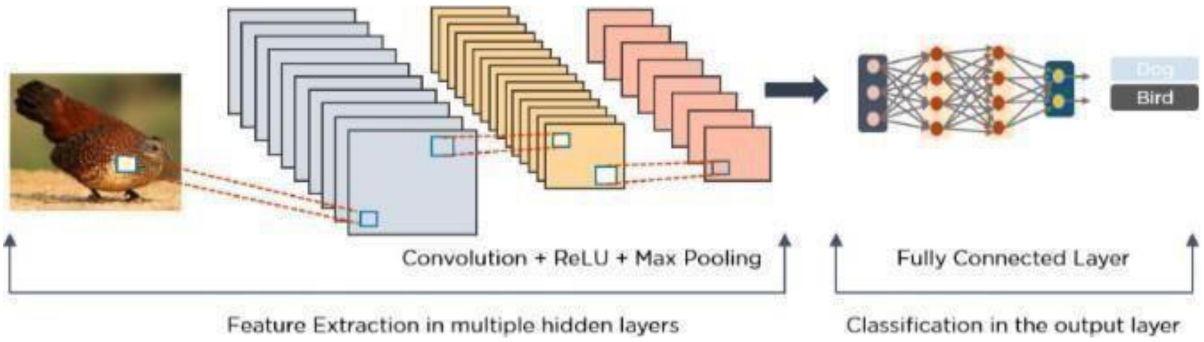
Convolutional layer is the first layer that is used to extract the various features from the input images. This layer separates and identifies the various features of the image for analysis in a process called as **Feature Extraction**. Various parameters such as filter,kernel size,activation,padding and input shape are used. The Activation function used in this layer Rectified Linear Unit(ReLU) and it returns 0 if it receives zero input but for any positive value x it returns the x value.

Rectified Linear Unit (ReLU):

- CNN's have a ReLU layer to perform operations on elements. The output is a rectified feature map. **Pooling Layer**
- The rectified feature map next feeds into a pooling layer. Pooling is a down-sampling operation that reduces the dimensions of the feature map.
- The pooling layer then converts the resulting two-dimensional arrays from the pooled feature map into a single, long, continuous, linear vector by flattening it.

Fully Connected Layer

- A fully connected layer forms when the flattened matrix from the pooling layer is fed as an input, which classifies and identifies the images.



CNN

Applications of Deep Learning:

1.Self-Driving Cars

2.Visual Recognition

3.Fraud Detection

4.Healthcare

5.Personalisations

6.Detecting Developmental Delay in Children

7.Colorization of Black and White Images

8.Adding Sounds to Silent Movies

9.Facial Expression Recognition

10.Breast cancer prediction

Importance of Deep Learning:

Deep Learning is a branch of artificial intelligence that uses data to enable machines **to** learn to perform tasks on their own. This technology is already live and used in automatic email reply predictions, virtual assistants, facial recognition systems[17], and self- driving cars. Breakthroughs in this technology are also making an impact in the health care sector. Using these types of advanced analytics, we can provide better information to health care.

The ability to process large numbers of features makes deep learning very powerful when dealing with unstructured data. However, deep learning algorithms can be overkill for less complex problems because they require access to a vast amount of data to be effective. If the data is too simple or incomplete, it is very easy for a deep learning model to become overfitted and fail to generalize well to new data. As a result, deep learning models are not as effective as other techniques (such as boosted decision trees or linear models) for most practical business problems such as understanding customer churn, detecting fraudulent transactions and other cases with smaller datasets and fewer features. In certain cases like multiclass classification, deep learning can work for smaller, structured datasets.

Implementation of Deep Learning using Python

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- Web development (server-side),
- software development,
- mathematics and System scripting.

The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.

It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse, Anaconda which are particularly useful when managing larger collections of Python files.

Python was designed for its readability. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

In the older days, people used to perform Deep Learning tasks manually by coding all the algorithms and mathematical and statistical formula. This made the process time consuming, tedious and inefficient. But in the modern days, it is become very much easy and efficient compared to the olden days by various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reason is its vast collection of libraries. Python libraries that used in Deep Learning are:

1. Pandas

2.Numpy

3.Matplotlib

4.Tensor flow

5.Keras

Pandas: It is a popular Python library for data analysis. It is not directly related to Deep Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

NumPy: It is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Deep Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

Matpoltlib: It is a very popular Python library for data visualization. Like Pandas, it is not directly related to Deep Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots[9]. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, histogram, error charts, bar chats, etc.

TensorFlow: It is an open-source library developed by Google primarily for deep learning applications[12]. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind.

Keras : It is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models[13]. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows to define and train neural network models in just a few lines of code.

3.SYSTEM ANALYSIS

Scope of the project

The main aim of the proposed work is to develop an automated model which has capability of identifying the species of the bird where bird image is given as a test image from the dataset. The main objectives are to develop an automated model by making use of train an

Data Pre-processing:

a) Data Visualization:

The total number of images in the dataset is visualized in both categories – 'images' and 'corresponding captions'.

b) Data Augmentation:

ImageDataGenerator is a class from Keras,it provides a quick and easy way to augment your images. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change.This augmentation makes the model robust and predicts the accurate output.The flow_from_directory() method allows you to read the images directly from the directory and augment them while the neural network model is learning on the training data.

Need of Data Pre-processing

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format.

Another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set, and best out of them is chosen.

Preparing the dataset :

The collected images were divided into train and test datasets i.e. 80 percent of the total images of the species were used as training data and the remaining 20 percent for the purpose of testing the trained model. The total number of train and test images with the species name is as follows

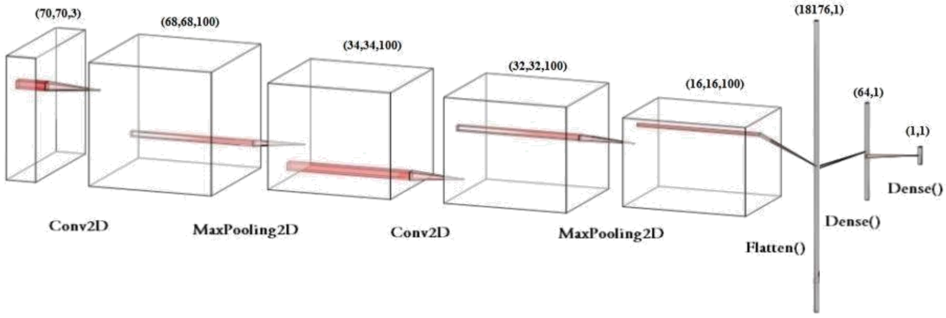
Build the Neural Network:

This convolution network consists of two pairs of Conv and MaxPool layers to extract features from the dataset. Which is then followed by a Flatten and Dense layer to convert the data in 1D and ensure overfitting[8].

- Conv2D Layer
    - **The filter** parameter means the number of this layer's output filters which is less in the early layers and more when we are closer to the prediction, [recommended to start up with 32,64,128 and the number varies according to the depth of the model]
    - **The kernal_size** specifies the width and the height of the 2D convolution window [odd integer and depend on the image size if image size > 128x128 then use 5*5 if less use 3x3 or 1x1]
    - **The activation** parameter refers to the type of activation function ○ **The padding** parameter is enabled to zero-padding to preserve the spatial dimensions of the volume so the output volume size matches the input volume size
    - **The input_shape** parameter has pixel high and pixel wide and have the 3 color channels: RGB
- MaxPool2D Layer

  To pool and reduce the dimensionlaity of the data[10]

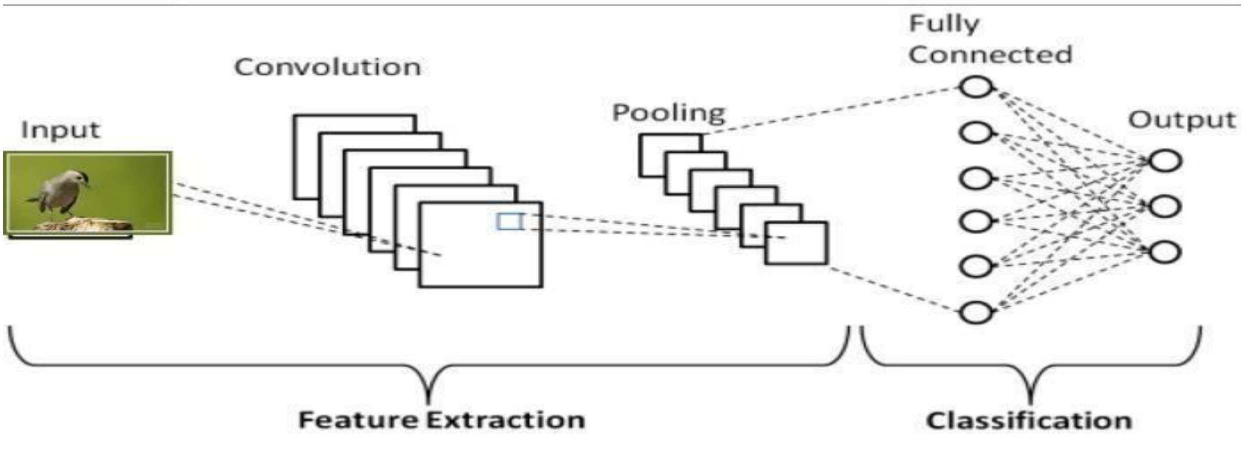
    - pool_size: max value over a 2x2 pooling window
    - strides: how far the pooling window moves for each pooling step
- Flatten Layer
    - flatten is used to flatten the input to a 1D vector then passed to dense
- Dense Layer (The output layer) ○ **The units** parameter means that it has 2 nodes one for with and one for without because we want a binary output
    - **The activation** parameter we use the softmax activation function on our output so that the output for each sample is a probability distribution over the outputs of with and without mask.



Convolutional Neural Network

Convolution Neural Network For Birds Species Classification :

A convolution neural network is a Deep learning algorithm which takes input images assign importance to various contents of the images and is self-capable of differentiating one from another. The input image is nothing but the pixel values in the computer vision, The RGB image consists of three planes RED, GREEN, BLUE. The main role of convolution neural network is to reduce the image into the form which is easy to process and to effectively mine all the features that could help in best classification.



The Convolution neural network model for bird's species classification.

4.IMPLEMENTATON

```
import os import pickle import
numpy as np from tqdm.notebook
import tqdm

from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input from
tensorflow.keras.preprocessing.image import load_img, img_to_array from
tensorflow.keras.preprocessing.text import Tokenizer from
tensorflow.keras.preprocessing.sequence import pad_sequences from
tensorflow.keras.models import Model from tensorflow.keras.utils import
to_categorical, plot_model from tensorflow.keras.layers import Input, Dense,
LSTM, Embedding, Dropout, add

# Import dataset
data_dir = 'gpiosenka/100-bird-species'
print('Folders : ', os.listdir(data_dir))
classes = os.listdir(data_dir + "/train")
print(len(classes))
print('200 classes : ', classes)
# load vgg16 model
model = VGG16()
# restructure the model
model = Model(inputs=model.inputs,
outputs=model.layers[-2].output)
# summarize
print(model.summary())
#training dataset
dataset = ImageFolder(data_dir + '/train', transform=ToTensor())
print('Size of training dataset :', len(dataset))
#testing dataset
test = ImageFolder(data_dir + '/test', transform=ToTensor())
print('Size of test dataset :', len(test))
img, label = dataset[0]
print(img.shape)
def show_example(img, label):
    print('Label: ', dataset.classes[label], "("+str(label)+")")
```

```
#examples showing

show_example(*dataset[0])

show_example(*dataset[1099])

show_example(*dataset[20199])

show_example(*dataset[6099])

torch.manual_seed(45)
val_size = 8000
train_size = len(dataset) - val_size

train_ds, val_ds = random_split(dataset, [train_size, val_size])
len(train_ds), len(val_ds)

batch_size = 256
train_loader = DataLoader(train_ds, batch_size, shuffle=True, num_workers=4, pin_memory=True)
val_loader = DataLoader(val_ds, batch_size*2, num_workers=4, pin_memory=True)
test_loader = DataLoader(test, batch_size*2, num_workers=4, pin_memory=True)

for images, labels in train_loader:
    #images,labels = images.to(device),labels.to(device)
    fig, ax = plt.subplots(figsize=(12, 6))
    ax.set_xticks([]); ax.set_yticks([])
    ax.imshow(make_grid(images, nrow=16).permute(1, 2, 0))
    break

#defining model

def apply_kernel(image, kernel):
    ri, ci = image.shape       # image dimensions
    rk, ck = kernel.shape      # kernel dimensions
    ro, co = ri-rk+1, ci-ck+1  # output dimensions
    output = torch.zeros([ro, co])
    for i in range(ro):
        for j in range(co):
            output[i,j] = torch.sum(image[i:i+rk,j:j+ck] * kernel)
    return output

simple_model = nn.Sequential(
    nn.Conv2d(3, 8, kernel_size=3, stride=1,
    padding=1), nn.MaxPool2d(2, 2)
)
simple_model.cuda()

def get_default_device():
    """Pick GPU if available, else CPU"""
    if torch.cuda.is_available():
        return torch.device('cuda')
    else:
        return torch.device('cpu')

def to_device(data, device):
    """Move tensor(s) to chosen device"""
    if isinstance(data, (list,tuple)):
```

```
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

class DeviceDataLoader():
    """Wrap a dataloader to move data to a device"""
    def __init_(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter_(self):
        """Yield a batch of data after moving it to device"""
        for b in self.dl:
            yield to_device(b, self.device)

    def __len_(self):
        """Number of batches"""
        return len(self.dl)

device = get_default_device()
device

for images, labels in train_loader:
    #images = images.cuda
    images,labels = images.to(device),labels.to(device)
    print('images.shape:', images.shape)
    out = simple_model(images)
    #out = out.cuda
    print('out.shape:', out.shape)
    break

def accuracy(outputs, labels):
    _, preds = torch.max(outputs, dim=1)
    return torch.tensor(torch.sum(preds == labels).item() / len(preds))

class ImageClassificationBase(nn.Module):
    def training_step(self, batch):
        images, labels = batch
        images,labels = images.to(device),labels.to(device)
        out = self(images) # Generate predictions loss =
        F.cross_entropy(out, labels) # Calculate loss return
        loss


    def validation_step(self, batch):
        images, labels = batch
        images,labels = images.to(device),labels.to(device)
        out = self(images) # Generate predictions loss =
        F.cross_entropy(out, labels) # Calculate loss
        acc = accuracy(out, labels) # Calculate accuracy
        return {'val_loss': loss.detach(), 'val_acc': acc}

    def validation_epoch_end(self, outputs):
        batch_losses = [x['val_loss'] for x in outputs]
```

```
batch_accs = [x['val_acc'] for x in outputs]
        epoch_acc = torch.stack(batch_accs).mean()      # Combine accuracies
        return {'val_loss': epoch_loss.item(), 'val_acc': epoch_acc.item()}

    def epoch_end(self, epoch, result):
        print("Epoch [{}], train_loss: {:.4f}, val_loss: {:.4f}, val_acc: {:.4f}".format( epoch,
            result['train_loss'], result['val_loss'], result['val_acc']))

class CnnModel(ImageClassificationBase):
    def __init_(self):
        super().__init__()
        self.network = nn.Sequential(
            nn.Conv2d(3, 224, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(224, 256, kernel_size=3, stride=1,
            padding=1), nn.ReLU(),
            nn.MaxPool2d(2, 2), # output: 150 x 16 x 16

            nn.Conv2d(256, 512, kernel_size=3, stride=1,
            padding=1), nn.ReLU(),
            nn.Conv2d(512, 512, kernel_size=3, stride=1,
            padding=1), nn.ReLU(),
            nn.MaxPool2d(2, 2), # output: 200 x 8 x 8

            nn.Conv2d(512, 1024, kernel_size=3, stride=1,
            padding=1), nn.ReLU(),
            nn.Conv2d(1024, 1024, kernel_size=3, stride=1,
            padding=1), nn.ReLU(),
            nn.MaxPool2d(2, 2), # output: 250 x 4 x 4

            nn.Flatten(),
            nn.Linear(36000, 1500),
            nn.ReLU(),
            nn.Linear(1500, 1000),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(1000, 200))

    def forward(self, xb):
        return self.network(xb)

class CnnModel2(ImageClassificationBase):
    def __init_(self):
        super().__init__()
        # Use a pretrained model
        self.network = models.resnet18(pretrained=True)
        # Replace last layer
        num_ftrs = self.network.fc.in_features
        self.network.fc = nn.Linear(num_ftrs, 200)

    def forward(self, xb):
        return torch.sigmoid(self.network(xb))
```

```
model = CnnModel2()
model.cuda()

for images, labels in train_loader:
    images,labels = images.to(device),labels.to(device)
    print('images.shape:', images.shape)
    out = model(images)
    print('out.shape:', out.shape)
    print('out[0]:', out[0])
    break

train_dl = DeviceDataLoader(train_loader, device)
val_dl = DeviceDataLoader(val_loader, device)
#to_device(model, device);

#training the second model

@torch.no_grad()
def evaluate(model, val_loader):

    model.eval()
    outputs = [model.validation_step(batch) for batch in
    val_loader] return model.validation_epoch_end(outputs)


def fit(epochs, lr, model, train_loader, val_loader, opt_func=torch.optim.SGD):
    history = []
    optimizer = opt_func(model.parameters(), lr)
    for epoch in range(epochs):
        # Training Phase
        model.train()
        train_losses = []
        for batch in train_loader: #batches
            = batches.to(device) loss =
            model.training_step(batch)
            train_losses.append(loss)
            loss.backward() optimizer.step()
            optimizer.zero_grad()

        # Validation phase
        result = evaluate(model, val_loader)
        result['train_loss'] = torch.stack(train_losses).mean().item()
        model.epoch_end(epoch, result)
        history.append(result)
    return history

model = to_device(CnnModel2(), device)

evaluate(model, val_loader)

num_epochs = 20
opt_func = torch.optim.Adam
lr = 0.001

history = fit(num_epochs, lr, model, train_dl, val_dl, opt_func)

def plot_accuracies(history):
```

```
    accuracies = [x['val_acc'] for x in history]
    plt.plot(accuracies, '-x')
    plt.xlabel('epoch')
    plt.ylabel('accuracy')
    plt.title('Accuracy vs. No. of epochs');

plot_accuracies(history)

def plot_losses(history):
    train_losses = [x.get('train_loss') for x in history]
    val_losses = [x['val_loss'] for x in history]
    plt.plot(train_losses, '-bx')
    plt.plot(val_losses, '-rx')
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.legend(['Training', 'Validation'])
    plt.title('Loss vs. No. of epochs');

plot_losses(history)

evaluate(model, test_loader)
```

```
    image = img_to_array(image)
# reshape data for model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
    #    preprocess image for vgg
image = preprocess_input(image)
    # extract features
feature = model.predict(image, verbose=0)
    # get image ID
image_id = img_name.split('.')[0]
    # store feature
    features[image_id] = feature
```

```
            # delete additional spaces
        caption = caption.replace('\s+', ' ')
# add start and end tags to the caption
            caption = 'startseq ' + " ".join([word for word in caption.split() if len(word)>1]) + ' endseq'
captions[i] = caption
```

```
# generate caption for an image def
predict_caption(model, image, tokenizer, max_length):
   # add start tag for generation process
in_text = 'startseq'
   # iterate over the max length of sequence       for i in
range(max_length):            # encode input sequence
sequence = tokenizer.texts_to_sequences([in_text])[0]
      # pad the sequence              sequence =
pad_sequences([sequence], max_length)
      # predict next word   yhat = model.predict([image,
sequence], verbose=0)
      # get index with high probability
yhat = np.argmax(yhat)        # convert
index to word              word =
idx_to_word(yhat, tokenizer)
      # stop if word not found
if word is None:
         break
      # append word as input for generating next word
in_text += " " + word     # stop if we reach end tag
if word == 'endseq':
         break
     from nltk.translate.bleu_score import corpus_bleu
# validate with test data
actual, predicted = list(), list()

for key in tqdm(test):
# get actual caption
captions =
mapping[key] #
predict the caption for
image    y_pred =
predict_caption(model,
```

```
features[key],
tokenizer, max_length)
   # split into words    actual_captions = [caption.split()
for caption in captions]  y_pred = y_pred.split() #
append to the list actual.append(actual_captions)
predicted.append(y_pred)

# calcuate   BLEU   score   print("BLEU-1:   %f"   %   corpus_bleu(actual,
predicted,    weights=(1.0,    0,    0,    0)))   print("BLEU-2:    %f"    %
corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))

from   PIL   import   Image   import
matplotlib.pyplot    as    plt    def
generate_caption(image_name):
   # load the image
   # image_name = "1001773457_577c3a7d70.jpg"
  image_id = image_name.split('.')[0]
img_path  =  os.path.join(BASE_DIR,  "Images",  image_name)
image = Image.open(img_path) captions = mapping[image_id]
   print('---------------------Actual---------------------')    for caption in captions:
print(caption) # predict the caption y_pred = predict_caption(model,
features[image_id], tokenizer, max_length)
  print('..............................Predicted..............................')
  print(y_pred)
 plt.imshow(image)
 generate_caption("1001773457_577c3a7d70.jpg")

def data_augmentor(h_flip=True, v_flip=False, rotate=True,):
    augmentor = Sequential() # Create returned Sequential class.
    augmentor.add(layers.Rescaling(1./255)) # Rescale image values from 0 - 255 to 0 - 1.
    # Based on function arguments: perform random flipping
    if h_flip and v_flip:
        augmentor.add(layers.RandomFlip('horizontal_and_vertical'))
    elif h_flip:
        augmentor.add(layers.RandomFlip('horizontal'))
    elif v_flip:
        augmentor.add(layers.RandomFlip('vertical'))
    # Based on function arguments: perform random rotation
    if rotate:
```

```
    augmentor.add(layers.RandomRotation(0.2))
  return augmentor

  FILEPATH = '../input/100-bird-species/' # filepath for data
  BATCH_SIZE = 128 # keras training batch size
  IMAGE_SIZE = (224, 224) # size of images in dataset (keras will resize)



  ROUND_1_TRAINABLE_LAYERS = 0 # layers at end of the mobile_net base model to have
  trainable parameters
  ROUND_1_EPOCHS = 10

  ROUND_2_TRAINABLE_LAYERS = 38
  ROUND_2_LEARNING_SCALER = 0.2
  ROUND_2_EPOCHS = 10

  ROUND_3_TRAINABLE_LAYERS = 74
  ROUND_3_LEARNING_SCALER = 0.2
  ROUND_3_EPOCHS = 10

  LEARNING_RATE = 0.001

  MODEL_CHECKPOINT_FILEPATH = './Bird-Classifier-Model-Checkpoint.ckpt'

  SEED = 6278


train = image_dataset_from_directory(directory = FILEPATH + 'train/',
                                     shuffle = True,
                                     labels='inferred',
                                     label_mode='categorical',
                                     batch_size = BATCH_SIZE,
                                     image_size = IMAGE_SIZE,
                                     seed = SEED)

# Validation Dataset
valid = image_dataset_from_directory(directory = FILEPATH + 'valid/',
                                     shuffle = True,
                                     labels='inferred',
                                     label_mode='categorical',
```

```
                           batch_size = BATCH_SIZE,
                           image_size = IMAGE_SIZE,
                           seed = SEED)

# Test Dataset
test = image_dataset_from_directory(directory = FILEPATH + 'test/',
                           shuffle = True,
                           labels='inferred',
                           label_mode='categorical',
                           batch_size = BATCH_SIZE,
                           image_size = IMAGE_SIZE,
                           seed = SEED

class_names = train.class_names

plt.figure(figsize=(18, 10))

for images, labels in train.take(1):
    for i in range(15):
        ax = plt.subplot(3, 5, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[np.argmax(labels[i])])
        plt.axis("off")
```

```
def data_augmentor(h_flip=True, v_flip=False, rotate=True,):

    augmentor = Sequential() # Create returned Sequential class.
    augmentor.add(layers.Rescaling(1./255)) # Rescale image values from 0 - 255 to 0 - 1.

    # Based on function arguments: perform random flipping
    if h_flip and v_flip:
        augmentor.add(layers.RandomFlip('horizontal_and_vertical'))
    elif h_flip:
        augmentor.add(layers.RandomFlip('horizontal'))
    elif v_flip:
        augmentor.add(layers.RandomFlip('vertical'))

    # Based on function arguments: perform random rotation
    if rotate:
        augmentor.add(layers.RandomRotation(0.2))
```

```
augmentor1 = data_augmentor(h_flip = True, v_flip = True, rotate=False)
augmentor2 = data_augmentor(h_flip = False, v_flip = False, rotate=True)
augmentor3 = data_augmentor(h_flip = True, v_flip = False, rotate=True)

augs = [augmentor1, augmentor2, augmentor3]

details = ['H and V Flip',
           'Rotate',
           'H Flip and Rotate']

plt.figure(figsize=(20, 8))
for images, labels in train.take(1):
    for i, aug in enumerate(augs):
        img = images[i]
        for j in range(8):
            ax = plt.subplot(3, 8, (i*8 + j + 1))
            img_augmented = aug(tf.expand_dims(img, 0))
            plt.imshow(img_augmented[0])
            plt.axis('off')
            ax.set_title(details[i])

plt.suptitle('Different Augmentations Demonstration', fontsize=20);

IMAGE_SHAPE = IMAGE_SIZE + (3,)

mobile_base = MobileNetV2(input_shape = IMAGE_SHAPE,
                          include_top = False,
                          weights = 'imagenet')

mn_layers = len(mobile_base.layers)



mobile_base.trainable = True

print('Setting the last {} layers in the Mobile Net Base Model to
trainable!\n'.format(ROUND_1_TRAINABLE_LAYERS))

for mn_layer in mobile_base.layers[:-ROUND_1_TRAINABLE_LAYERS]:
    mn_layer.trainable = False

trainableParams = np.sum([np.prod(v.get_shape()) for v in mobile_base.trainable_weights])
nonTrainableParams = np.sum([np.prod(v.get_shape()) for v in
mobile_base.non_trainable_weights])
totalParams = trainableParams + nonTrainableParams

print('The number of trainable parameters in the Mobile Net V2 Base Model is
{}.'.format(trainableParams))
print('The number of non-trainable parameters in the Mobile Net V2 Base Model is
{}.'.format(nonTrainableParams))
print('The total number of parameters is {}.'.format(totalParams))
```

5.METHODOLOGY

The principal aim of this project is identification of images of birds and their classification into the individual species. This project has been developed on the mainstreamalgorithm of Deep Learning, which is Convolutional Neural Network (CNN).

The execution of the entire project comprises four main steps:

1. Gathering and Localizing the Bird Dataset.

   In order to build our Deep Learning image dataset, we utilized Microsofts Bing Image Search API v7. Bing Image Search API is a comprehensive Cognitive Service family of Microsoft. The dataset mostly includes the birds found in Asian sub-continent. The entire dataset houses 60 species of birds and consists of 8218 images.

2. Implementing the CNN architecture.

   The CNN architecture to be developed is a smaller and more portable version of the VGGNet network [9].

   Characteristics of VGGNet architectures are:

   - 3X3 Convolutional Layers stacked on each other at increasing depth.
   - Max Pooling to minimize the size of the image and the number of parameters.
   - Fully Connected Layers at the end of the network prior to a softmax classifier.

   TensorFlow Backend is used for implementing this architecture. Multiple layers of Convolution and ReLU are stacked together in order to learn an affluent set of attributes.

   The Convolution Layer has 32 filters with 3X3 feature detector for the first convolution block. This operation is followed by application of ReLU function. The Pooling layer incorporates a 3X3 pool to reduce the spatial dimensions from 96X96 to 32X32 (96X96X3 dimensional images were used to train the network).

   Another Convolution Layer is stacked onto this, where filter size is increased from 32 to 64 but the feature detector still being 3X3 dimensional. This is again followed by ReLU function. Further, Max Pooling is applied where the pool window size is decreased from 3X3 to 2X2 at strides of 2.

   A final Convolution Layer is used where filter size is increased to 124 followed by implementation of ReLU function. Max Pooling is applied again with the pool window size of 2X2 at strides of 2. A Dropout Layer is used to prevent overfitting with the dropout value of 0.25.

Finally, the Fully Connected Layers are added using the Dense Layer of size 1024. A Dropout Layer is implemented again with the value of 0.50.

At the end, Softmax classifier is used for predicting a single class out of various mutually exclusive classes.

3. Training the CNN Model.

After successful deployment the CNN Model, the network was all set to be trained with the bird images using Keras using Adam Optimizer. All the necessary packages were imported in the training script. Matplotlib backend was used for saving figures in the background.

For data augmentation, the ImageDataGenerator class has been used to increase the diversity of the information available for training models significantly without actually collecting new information. This technique also helps in preventing overfitting.

A common practice is to divide the dataset into training and testing sets when implementing deep learning. An 80:20 random split of the dataset was created with the help of train_test_split function, rendering 80% of the data for training and the remaining 20% for testing.

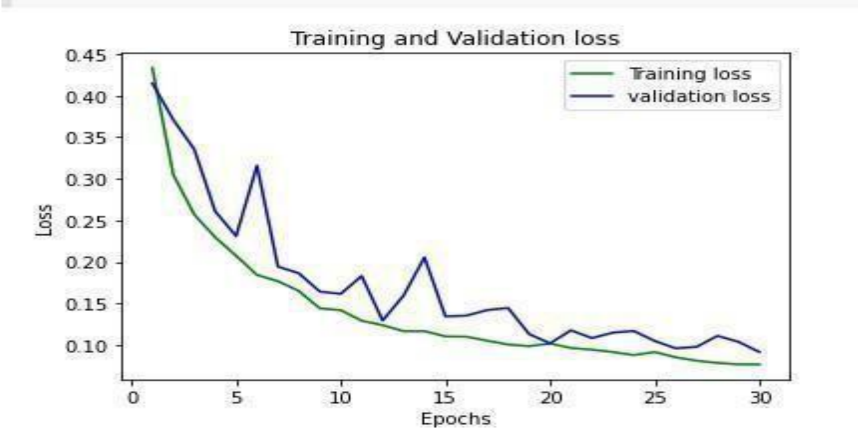
After the CNN finished training, both the Model and Label Binarizer files were saved to the local disk as it is required to load them in the framework, whenever the network is tested on images extrinsic to the training and testing dataset.

4. Testing the Efficacy of the Trained Model.

Now that the CNN model was trained, a classification script was implemented to identify images of birds.

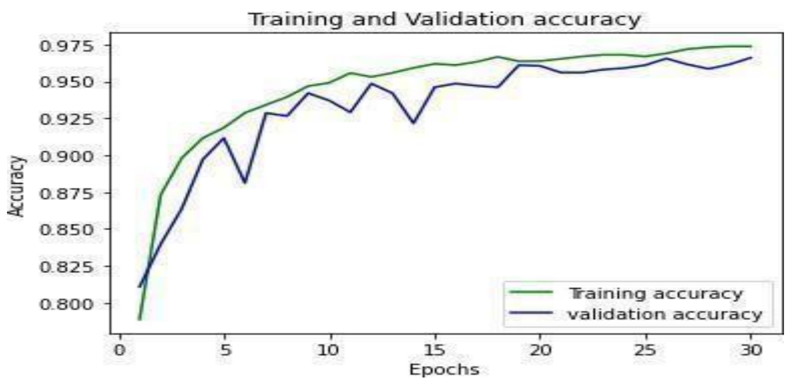
The user browses and uploads a sample image through the web portal. Ultimately, the client-server architecture navigates the submitted sample bird image to the testing script. The script retrieves information from the trained model and label binarizer file stored on the disk thereby successfully predicting the bird species.

6.RESULT ANALYSIS



Loss of Training and Validation

Here, the above figure shows the Training loss and Validation loss.Training and Validation loss were reducing when increase in Epochs.

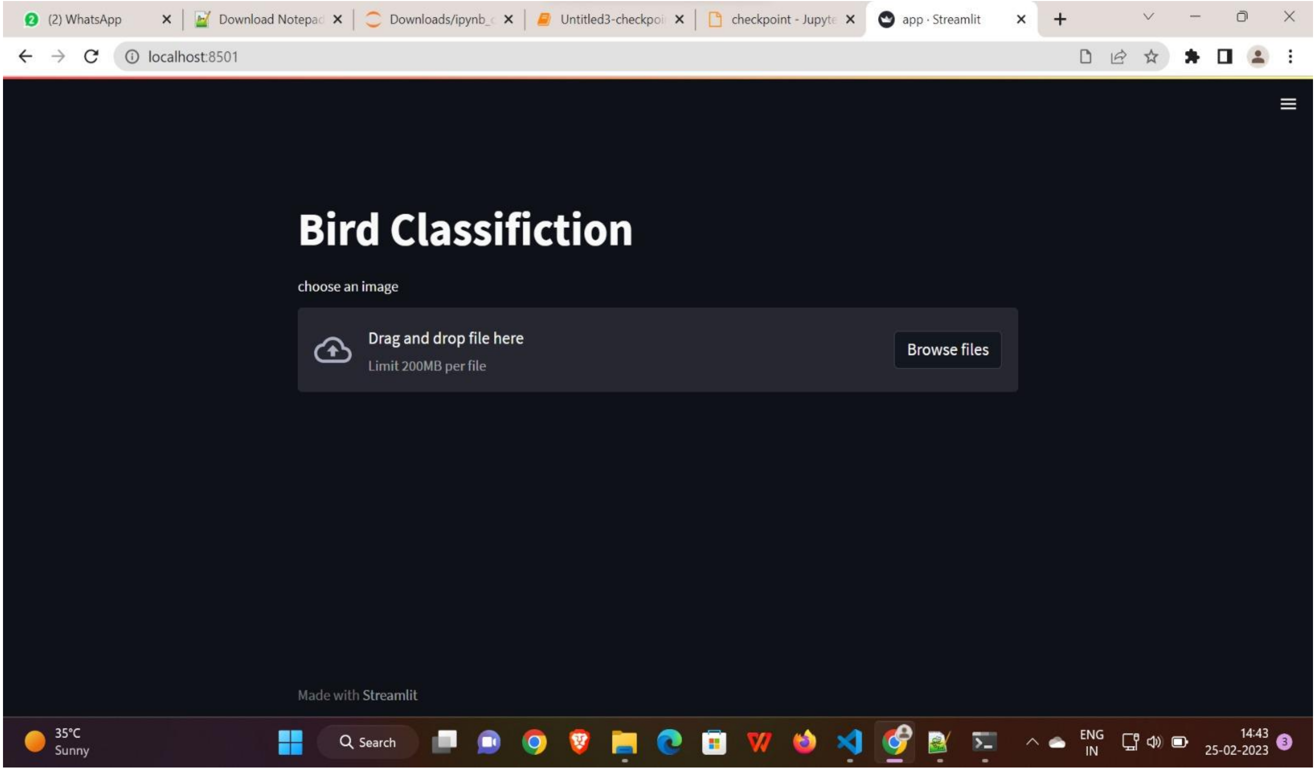


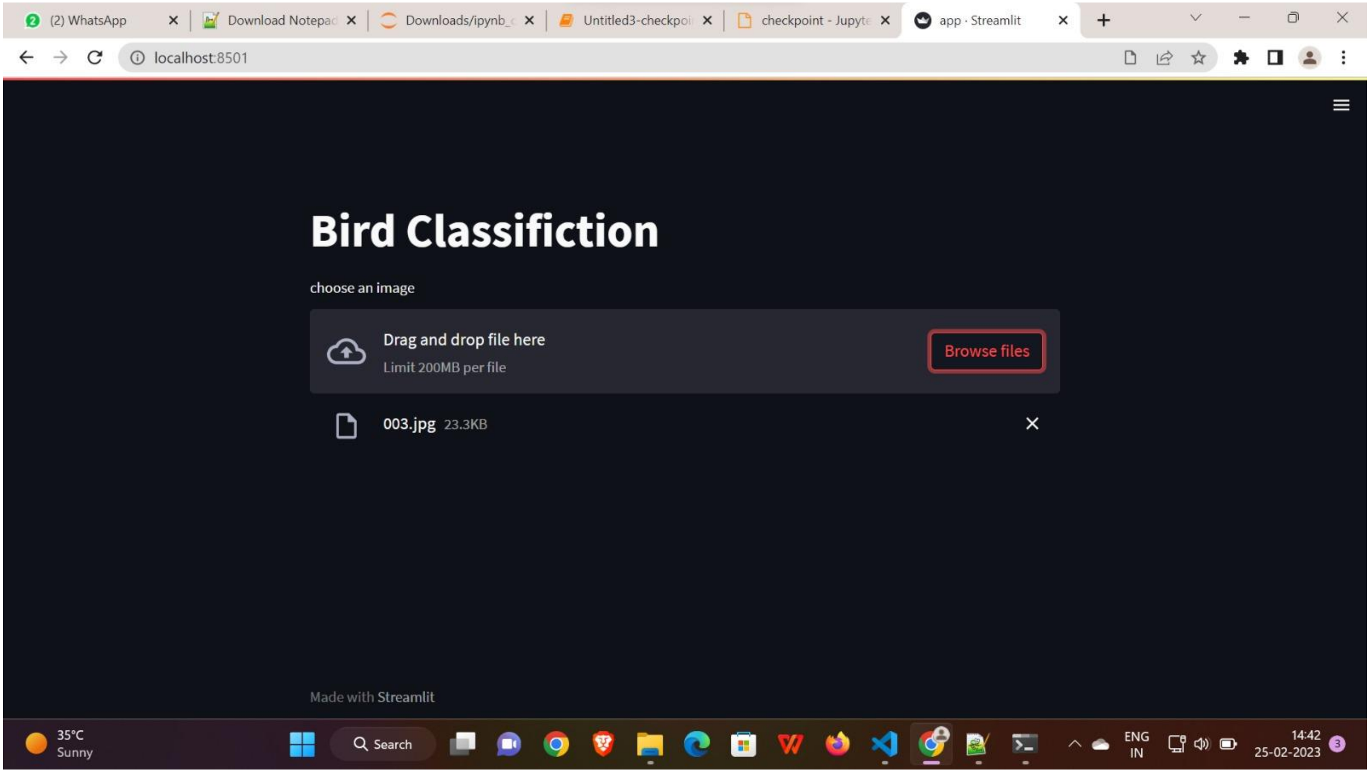
Accuracy of Training and Validation

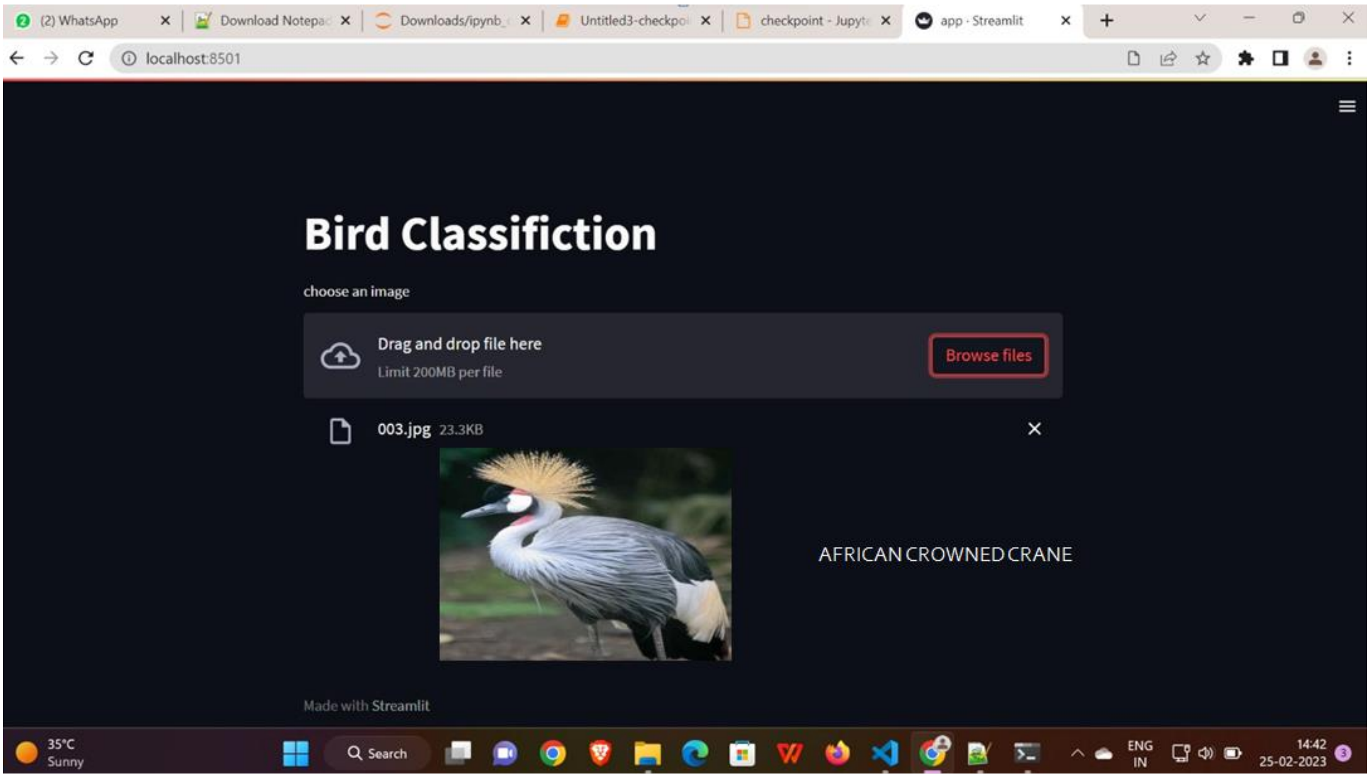
Here, the above figure shows the accuracy of Training and Validation.Accuracy of Training and Validation were increasing while increase in Epochs.

This study developed a software platform that uses deep learning for image processing to identify bird species from digital images uploaded or captured by an end-user on a smartphone in real time. [13] To develop such system a trained dataset is required to classify an image. Trained dataset consists of two parts trained result and test result. The dataset has to be retrained to achieve higher accuracy in identification. The trained dataset is created using 50000 steps, higher the number of steps higher its accuracy. The accuracy of trained dataset is 93%. The testing dataset has nearly 1000 images with an accuracy of 80%. Whenever a user will upload an input file, the image is temporarily stored in database. This input file is then passed to the system and is given to CNN where CNN is coupled with trained dataset. Various features such as head, body, color, beak, shape, entire image of bird is considered for classification to have maximum accuracy. Each feature is given through deep convocational network to extract features out. These features are then collected and forwarded to classifier. The input will be compared with the trained dataset to generate results. Image is compared with the pre trained dataset images and the score sheet is generated. The score sheet is an output of top 5 match results by which the highest matching value of score sheet is the resultof bird species.

7.SCRCEENSHOTS






(2) WhatsApp
Download Notepad
Downloads/ipynb
Untitled3-checkpoi
checkpoint - Jupyte
app · Streamlit
localhost:8501
Bird Classifiction
choose an image
Drag and drop file here
Limit 200MB per file
Browse files
003.jpg 23.3KB
AFRICAN CROWNED CRANE
Made with Streamlit
35°C
Sunny
Search
ENG
IN
14:42
25-02-2023

8.Conclusion

This model helps building applications that helps tourist who go onto bird sanctuaries identify the bird species by just capturing a picture of a bird and uploading it as input to the model. As many species of birds have become endangered and are near to extinction many people have no knowledge about the species which are few in number, Thus application built using this model may be helpful in identifying the endangered species and help society in spreading awareness about the need of all the species for balance in the nature. As the model implies the knowledge of Deep Convolution neural networks, we can infer that the CNN is the best algorithm for analysing the visual imagery and image Classification.

The CNN model gives the BLEU score with 0.6. The categories in results are due to neighbourhood of some particular words, i.e., for word like car it's neighbourhood words like vehicle, van, cab etc. are also generated which might be incorrect. After so much of experiments, it is conclusive that use of larger datasets increases performance of the model. The larger dataset will increase accuracy as well as reduce losses. Also, it will be interesting that how unsupervised data for both images as well as text can be used for improving the image caption generating approaches.

9.Future Scope

The future of image processing involves new intelligent, digital automated robots made by research scientists in various parts of the world. [15] It includes development in various image processing applications. Due to changes in image processing and other related technologies, there will be millions of robots in the world in a few, transforming the way of living. Researches in image processing and artificial intelligence will involve voice commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing operation & surgery, reprogramming defects in human DNA, and automatic driving all formats of transportation. And for Image based species recognition of birds [6] we can further enhance the system with cloud feature which can store large amount of data for comparison and in case of neural network it can provide high computing power for processing.

10.References

1. M.Sireesha, S. N. TirumalaRao, Srikanth Vemuru, Frequent Itemset Mining Algorithms: A Survey Journal of Theoretical and Applied Information Technology Vol - 96, No .3, Feb - 2018 ISSN - 1992-8645, Pages – 744 – 755.

2. M. Sireesha, Srikanth Vemuru and S. N. TirumalaRao, ”Coalesce based binary table: an enhanced algorithm for mining frequent patterns", International Journal of Engineering and Technology, vol. 7, no. 1.5, pp. 51-55, 2018.

3. M.Sireesha, S. N. TirumalaRao, Srikanth Vemuru, Optimized Feature Extraction and Hybrid Classification Model for Heart Disease and Breast Cancer Prediction International Journal of Recent Technology and Engineering Vol - 7, No 6, Mar - 2019 ISSN - 2277-3878, Pages – 1754 – 1772

4. M.Sireesha, Srikanth Vemuru, S.N.Tirumala Rao “Classification Model for Prediction Of Heart Disease Using Correlation Coefficient Technique” International Journal of Advanced Trends in Computer Science and Engineering, Vol. 9, No. 2, March - April 2020, Pages- 2116 – 2123.

5. Sireesha Moturi , Dr. S. N. Tirumala Rao, Dr. Srikanth Vemuru,. (2020). Predictive Analysis of Imbalanced Cardiovascular Disease Using SMOTE. International Journal of Advanced Science and Technology, 29(05), 6301 - 6311.

6. Moturi S., Tirumala Rao S.N., Vemuru S. (2021) Risk Prediction-Based Breast Cancer Diagnosis Using Personal Health Records and Machine Learning Models. In: Bhattacharyya D., Thirupathi Rao N. (eds) Machine Intelligence and Soft Computing. Advances in Intelligent Systems and Computing, vol 1280. Springer, Singapore. https://doi.org/10.1007/978-981-15-9516-5_37

7. Moturi S., Srikanth Vamuru, Tirumala Rao S.N. (2021) ECG based Decision Support System for Clinical Management using Machine Learning Techniques. IOP Conference Series: Materials Science and Engineering. Volume 1085, Annual International Conference on Emerging Research Areas on "COMPUTING & COMMUNICATION SYSTEMS FOR A FOURTH INDUSTRIAL REVOLUTION" (AICERA 2020) 14th- 16th December 2020, Kanjirapally, India.

8. “Guide to the Sequential model - Keras Documentation”, Faroit.com, 2020. [Online]. Available:https://faroit.com/keras-docs/1.0.1/gettingstarted/sequential-model-guide/.

9. F. Hohman, M. Kahng, R. Pienta and D. H. Chau, “Visual Analytics in Deep Learning: An

Interrogative Survey for the Next Frontiers," in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 8, pp. 2674-2693, 1 Aug. 2019, doi: 10.1109/TVCG.2018.2843369.

10. K. Team, "Keras documentation: MaxPooling2D layer", Keras.io, 2020. [Online]. Available: https://keras.io/api/layers/pooling layers/ max pooling2d/. 2020.

11. "Bird Species Classificaion using Deep Learning" Kaggle.com, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9077750.

12. "TensorFlow White Papers", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/about/bib. 2020.

13. K. Team, "Keras documentation: About Keras", Keras.io, 2020. [Online].Available: https://keras.io/about. 2020.

14. "OpenCV", Opencv.org, 2020. [Online]. Available: https://opencv.org/.2020.

15. D. Meena and R. Sharan, "An approach to face detection and recognition," 2016 International Conference on Recent Advances andInnovations in Engineering (ICRAIE), Jaipur, 2016, pp. 1-6, doi:10.1109/ICRAIE.2016.7939462.

16. S. Ge, J. Li, Q. Ye and Z. Luo, "Detecting Masked Faces in the Wild with LLE-CNNs," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 426-434, doi:10.1109/CVPR.2017.53.

17. C. Kanan and G. Cottrell, "Color-to-Grayscale: Does the Method Matter in Image Recognition?",PLoSONE,vol.7,no.1,p.e29740,2012.Available:10.1371/journal.pone.0029 740.

18. W.H.O., "Coronavirus disease 2019 (covid-19): situation report, 205".2020

19. "Coronavirus Disease 2019 (COVID-19) – Symptoms", Centers for Disease Control and Prevention,2020.[Online].Available:https://www.cdc.gov/coronavirus/2019ncov/symptm s testing/symptoms.html. 2020.

20. T. Meenpal, A. Balakrishnan, and A. Verma, "Facial Mask Detection using Semantic Segmentation," 2019 4th Int. Conf. Comput. Commun. Secur. ICCCS 2019, no. October, pp. 1– 5, 2019, doi: 10.1109/CCCS.2019.8888092.

21. "Paris Tests Face-Mask Recognition Software on Metro Riders", May 2020, [online] Available: Bloomberg.com.

22. W.H.O., “Advice on the use of masks in the context of COVID-19:interim guidance”, 2020.

23. Opencv-python-tutroals.readthedocs.io. 2020. Changing Colorspaces — Opencv-Python Tutorials 1 Documentation. [online] Available at:https://opencv-python-tutroals.readthedocs.io/en/latest/pytutorials/pyimgproc/pycolorspaces/pycolorspaces.html
.
2020.