Digital Naturalist - AI Enabled Tool For Biodiversity Researchers Using IBM Watson

1. INTRODUCTION

1.1 Overview

A naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature. Being able to identify the flora and fauna around us often leads to an interest in protecting wild spaces, and collecting and sharing information about the species we see on our travels is very useful for conservation groups like NCC.

When venturing into the woods, field naturalists usually rely on common approaches like always carrying a guidebook around everywhere or seeking help from experienced ornithologists. There should be a handy tool for them to capture, identify and share the beauty to the outside world.Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals and other species they see on their hikes, canoe trips and other excursions.

In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals and get the prediction of the bird when an image is been given.

1.2 Purpose

Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals and other species they see on their hikes, canoe trips and other excursions.

2. LITERATURE SURVEY

2.1 Existing problem

When venturing into the woods, field naturalists usually rely on common approaches like always carrying a guidebook around everywhere or seeking help from experienced ornithologists. There should be a handy tool for them to capture, identify and share the beauty to the outside world

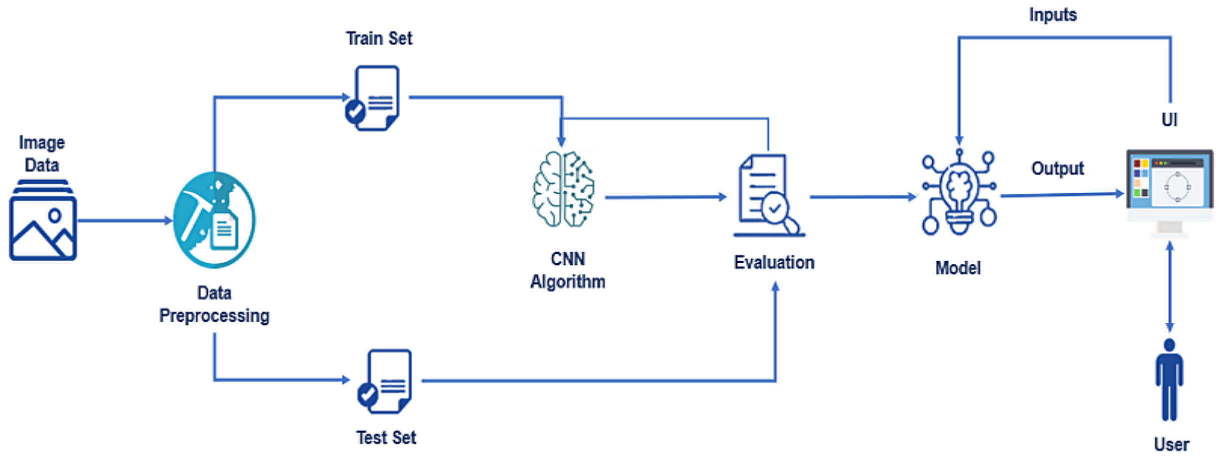
2.2 Proposed solution

A naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature. Being able to identify the flora and fauna around us often leads to an interest in protecting wild spaces, and collecting and sharing information about the species we see on our travels is very useful for conservation groups like NCC.Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals and other species they see on their hikes, canoe trips and other excursions.

In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals and get the prediction of the bird when an image is been given.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software designing

Software Requirements:

- Anaconda Navigator
- Tensor flow
- Keras
- Flask

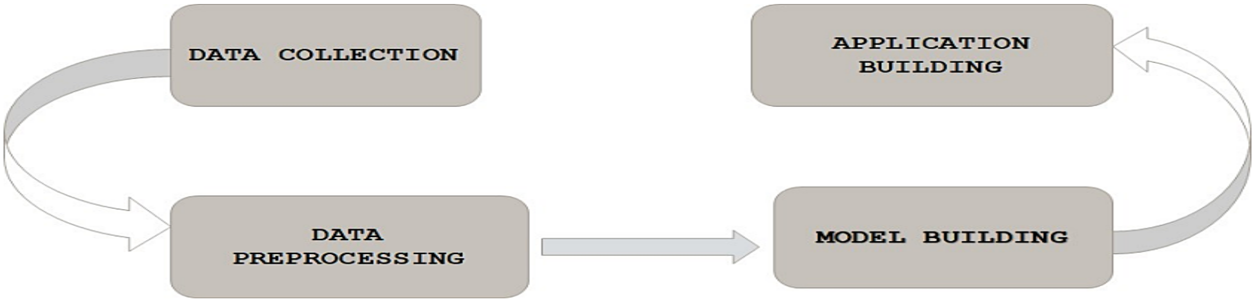
Hardware Requirements:

- Processor : Intel Core i3
- Hard Disk Space : Min 100 GB
- Ram : 4 GB
- Display : 14.1 “Color Monitor(LCD, CRT or LED)
- Clock Speed : 1.67 GHz

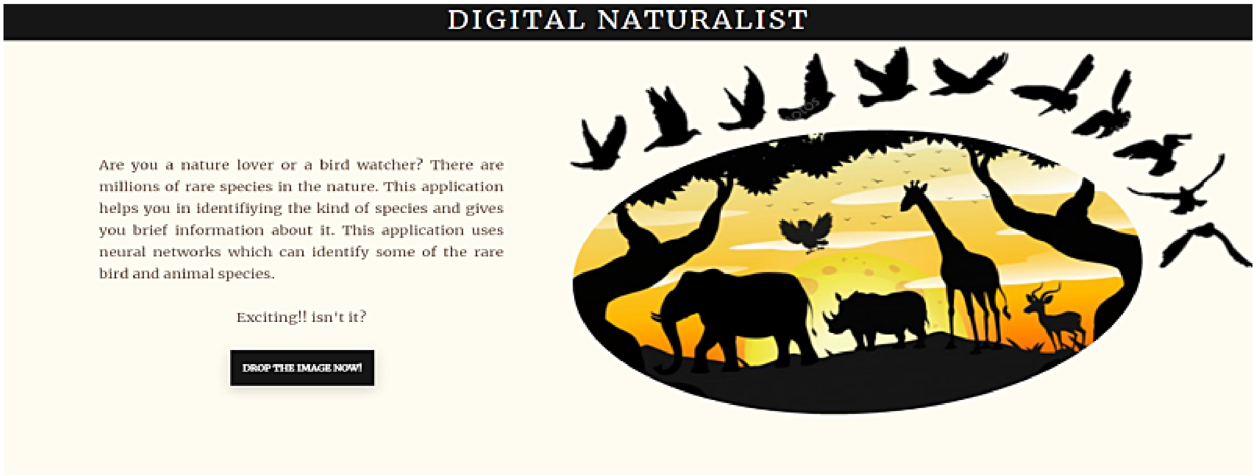
4. EXPERIMENTAL INVESTIGATIONS

Study shows that it provide with different test images of a species of birds, flowers and mammals, the model detects, upload image of the species. When we choose an image and click in it then it will shows the predicted output.

5. FLOWCHART

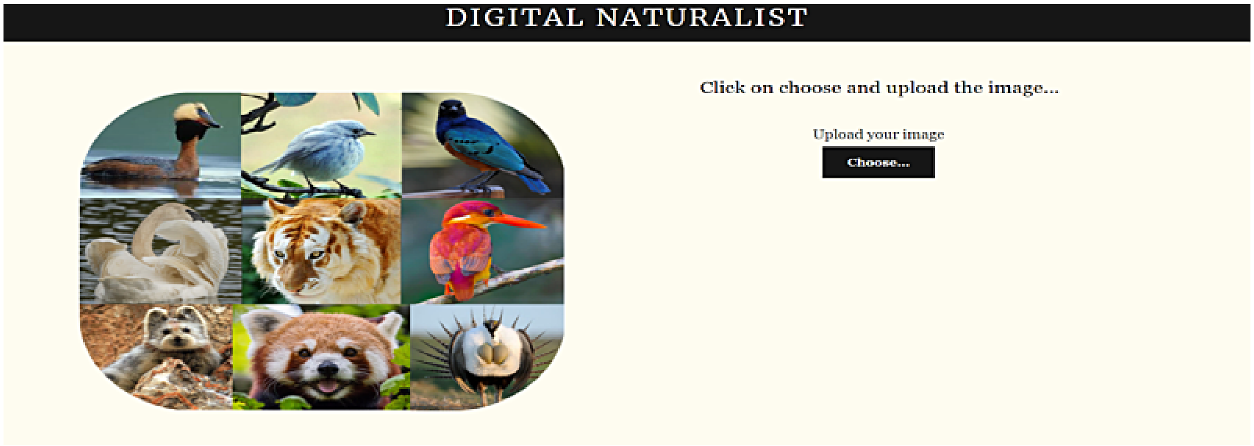


6. RESULT



Are you a nature lover or a bird watcher? There are millions of rare species in the nature. This application helps you in identifiying the kind of species and gives you brief information about it. This application uses neural networks which can identify some of the rare bird and animal species.

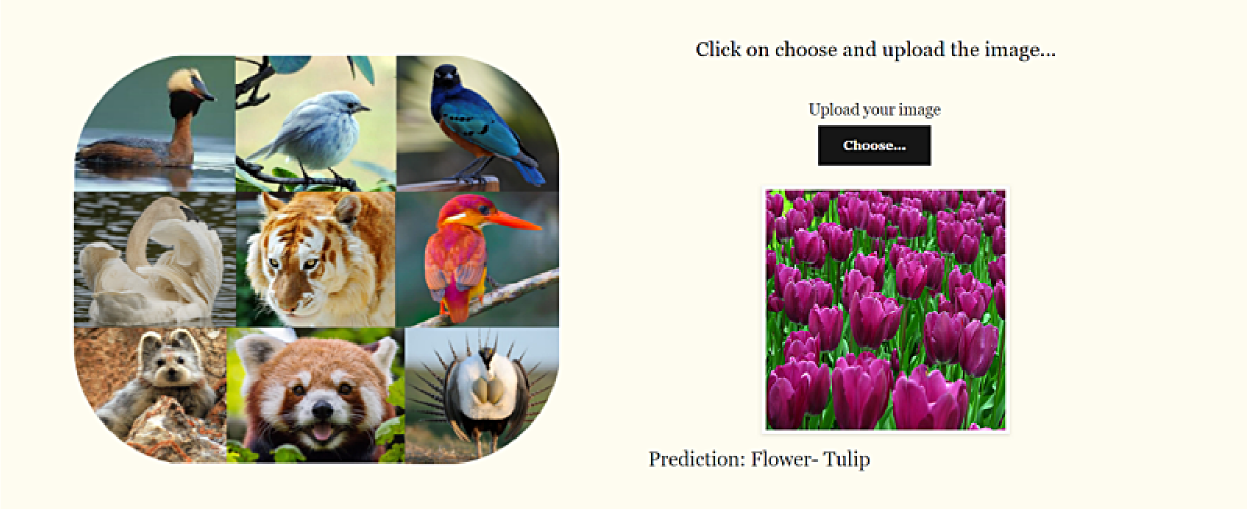
Exciting!! isn't it?



DIGITAL NATURALIST

Click on choose and upload the image...

Upload your image



Click on choose and upload the image...

Upload your image

Prediction: Flower- Tulip

7. ADVANTAGES & DISADVANTAGES

Advantages:

- Increased accuracy for effective species prediction
- Reduce time complexity of Naturalist.

Disadvantages:

- Data mining technique does not help to provide effective decision making.
- Can not handle enormous datasets for species records.

8. APPLICATIONS

- Deep learning technology is considered as one of the key technology used in AI enable technology in Biodiversity like Digital Naturalist.
- Deep learning based on machine learning and can be used to perform a prediction on uploading image.

9. CONCLUSION

In this project, we have established the application to predict different species based on the IBM cloud application.Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals and other species they see on their hikes, canoe trips and other excursions.

10. FUTURE SCOPE

The project can be further enhanced by deploying the deep learning model obtained using a web application and a larger dataset could be used for prediction to give higher accuracy and produce better result.

11. BIBILOGRAPHY

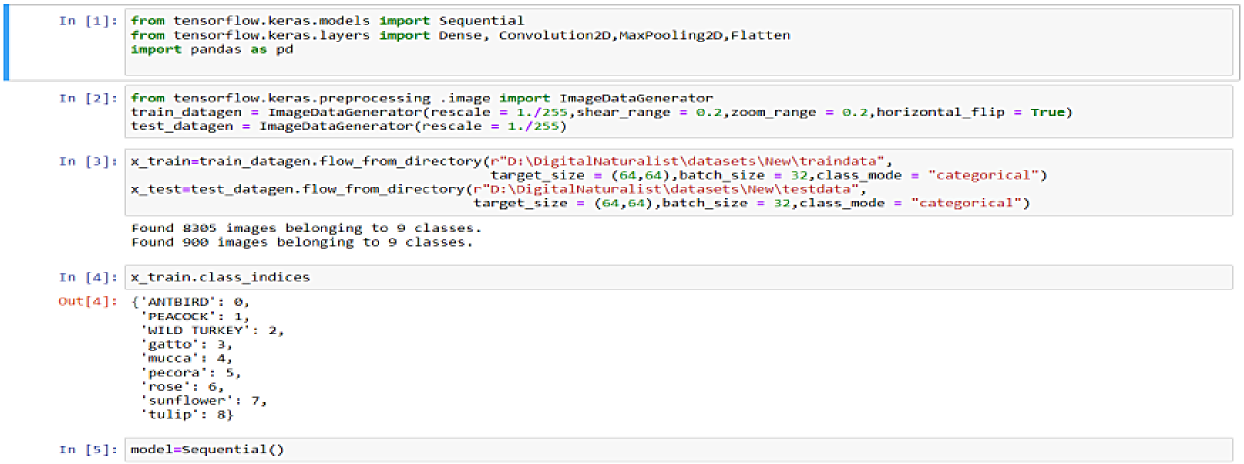
- Rémy Chauvin. 1977. Ethology: the biological study of animal behavior. International

Universities Press. Retrieved November 7, 2012 from http://books.google.com/books?id=h9EgAQAAIA AJ&pgis=1

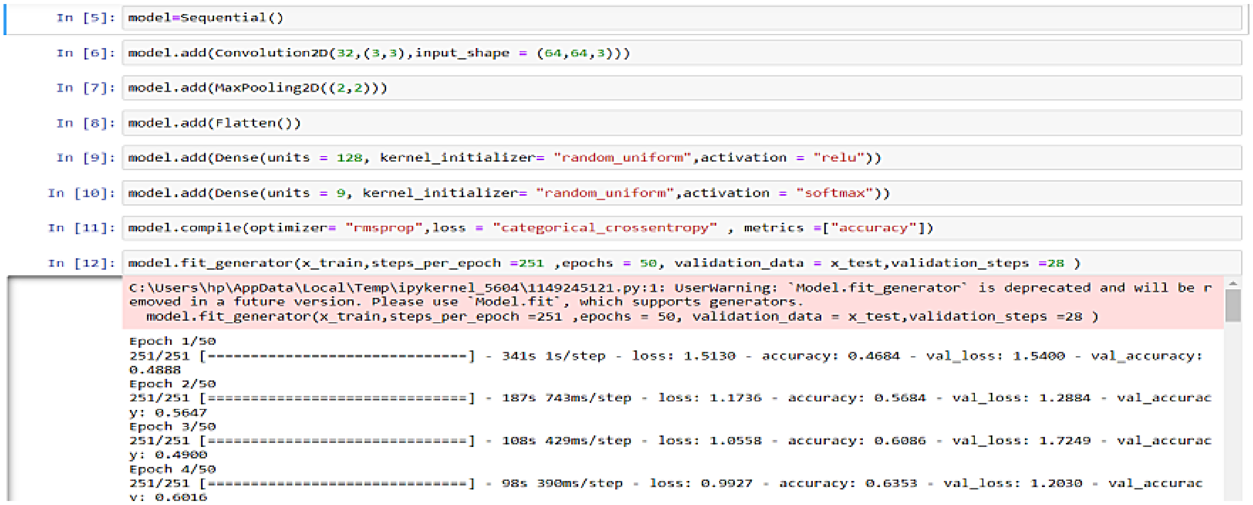
- Michael S Caldwell, J Gregory McDaniel, and Karen M Warkentin. 2009. Frequency information in the vibration-cued escape hatching of red-eyed treefrogs. The Journal of experimental biology 212, Pt 4: 566--75

APPENDIX

Source Code

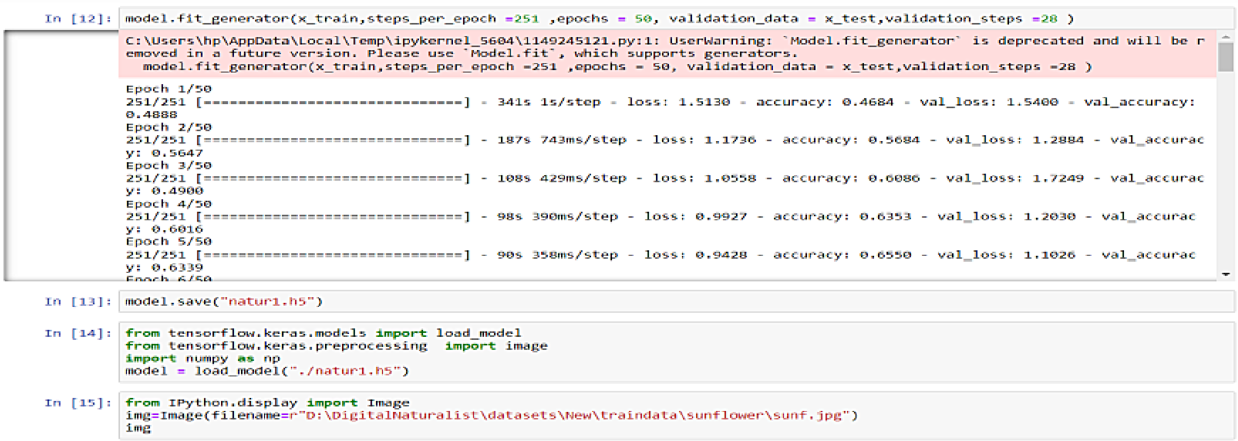


```
In [1]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Convolution2D,MaxPooling2D,Flatten
        import pandas as pd

In [2]: from tensorflow.keras.preprocessing .image import ImageDataGenerator
        train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,zoom_range = 0.2,horizontal_flip = True)
        test_datagen = ImageDataGenerator(rescale = 1./255)

In [3]: x_train=train_datagen.flow_from_directory(r"D:\DigitalNaturalist\datasets\New\traindata",
                                                  target_size = (64,64),batch_size = 32,class_mode = "categorical")
        x_test=test_datagen.flow_from_directory(r"D:\DigitalNaturalist\datasets\New\testdata",
                                                target_size = (64,64),batch_size = 32,class_mode = "categorical")

        Found 8305 images belonging to 9 classes.
        Found 900 images belonging to 9 classes.

In [4]: x_train.class_indices
Out[4]: {'ANTBIRD': 0,
         'PEACOCK': 1,
         'WILD TURKEY': 2,
         'gatto': 3,
         'mucca': 4,
         'pecora': 5,
         'rose': 6,
         'sunflower': 7,
         'tulip': 8}

In [5]: model=Sequential()
```



```
In [5]: model=Sequential()

In [6]: model.add(Convolution2D(32,(3,3),input_shape = (64,64,3)))

In [7]: model.add(MaxPooling2D((2,2)))

In [8]: model.add(Flatten())

In [9]: model.add(Dense(units = 128, kernel_initializer= "random_uniform",activation = "relu"))

In [10]: model.add(Dense(units = 9, kernel_initializer= "random_uniform",activation = "softmax"))

In [11]: model.compile(optimizer= "rmsprop",loss = "categorical_crossentropy" , metrics =["accuracy"])

In [12]: model.fit_generator(x_train,steps_per_epoch =251 ,epochs = 50, validation_data = x_test,validation_steps =28 )
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_5604\1149245121.py:1: UserWarning: `Model.fit_generator` is deprecated and will be r
emoved in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(x_train,steps_per_epoch =251 ,epochs = 50, validation_data = x_test,validation_steps =28 )

Epoch 1/50
251/251 [==============================] - 341s 1s/step - loss: 1.5130 - accuracy: 0.4684 - val_loss: 1.5400 - val_accuracy:
0.4888
Epoch 2/50
251/251 [==============================] - 187s 743ms/step - loss: 1.1736 - accuracy: 0.5684 - val_loss: 1.2884 - val_accurac
y: 0.5647
Epoch 3/50
251/251 [==============================] - 108s 429ms/step - loss: 1.0558 - accuracy: 0.6086 - val_loss: 1.7249 - val_accurac
y: 0.4900
Epoch 4/50
251/251 [==============================] - 98s 390ms/step - loss: 0.9927 - accuracy: 0.6353 - val_loss: 1.2030 - val_accurac
y: 0.6016
```

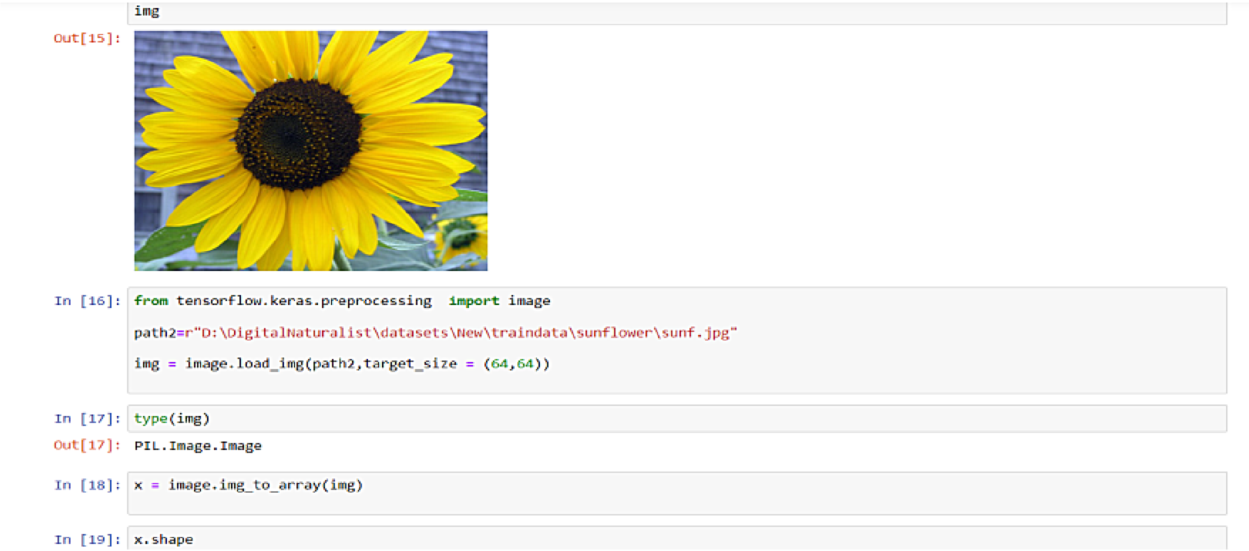


```
In [12]: model.fit_generator(x_train,steps_per_epoch =251 ,epochs = 50, validation_data = x_test,validation_steps =28 )
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_5604\1149245121.py:1: UserWarning: `Model.fit_generator` is deprecated and will be r
emoved in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(x_train,steps_per_epoch =251 ,epochs = 50, validation_data = x_test,validation_steps =28 )

Epoch 1/50
251/251 [==============================] - 341s 1s/step - loss: 1.5130 - accuracy: 0.4684 - val_loss: 1.5400 - val_accuracy:
0.4888
Epoch 2/50
251/251 [==============================] - 187s 743ms/step - loss: 1.1736 - accuracy: 0.5684 - val_loss: 1.2884 - val_accurac
y: 0.5647
Epoch 3/50
251/251 [==============================] - 108s 429ms/step - loss: 1.0558 - accuracy: 0.6086 - val_loss: 1.7249 - val_accurac
y: 0.4900
Epoch 4/50
251/251 [==============================] - 98s 390ms/step - loss: 0.9927 - accuracy: 0.6353 - val_loss: 1.2030 - val_accurac
y: 0.6016
```

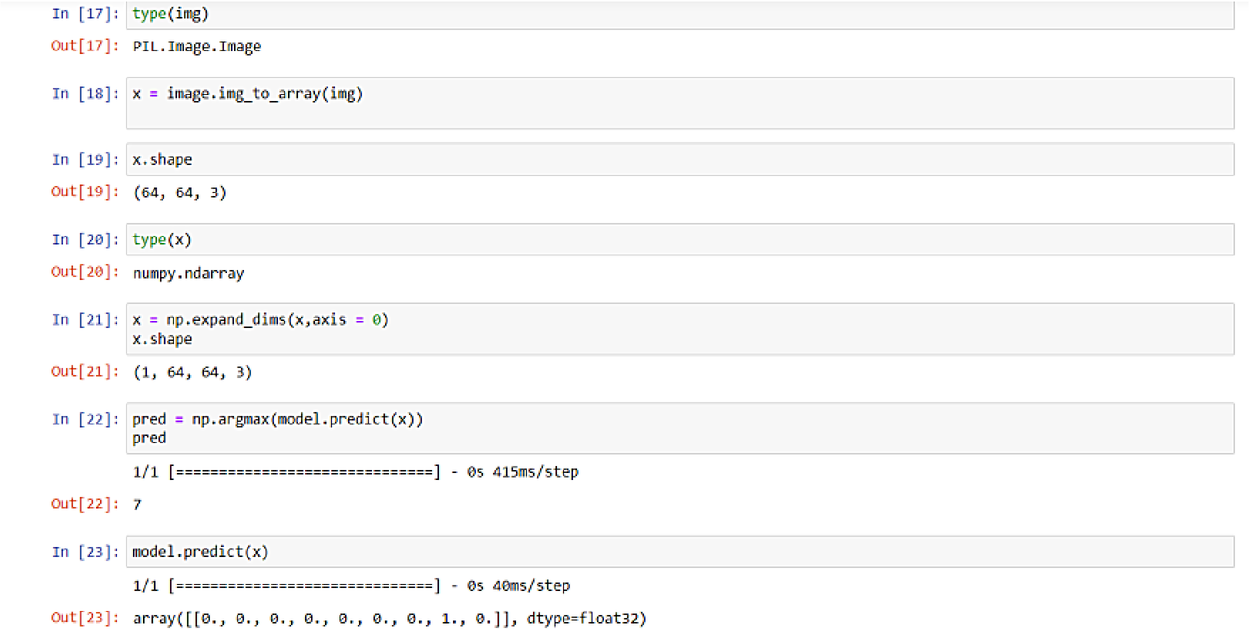
```
Epoch 5/50
251/251 [==============================] - 90s 358ms/step - loss: 0.9428 - accuracy: 0.6550 - val_loss: 1.1026 - val_accurac
y: 0.6339
Epoch 6/50
```

```
In [13]: model.save("natur1.h5")

In [14]: from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing  import image
         import numpy as np
         model = load_model("./natur1.h5")

In [15]: from IPython.display import Image
         img=Image(filename=r"D:\DigitalNaturalist\datasets\New\traindata\sunflower\sunf.jpg")
         img
```



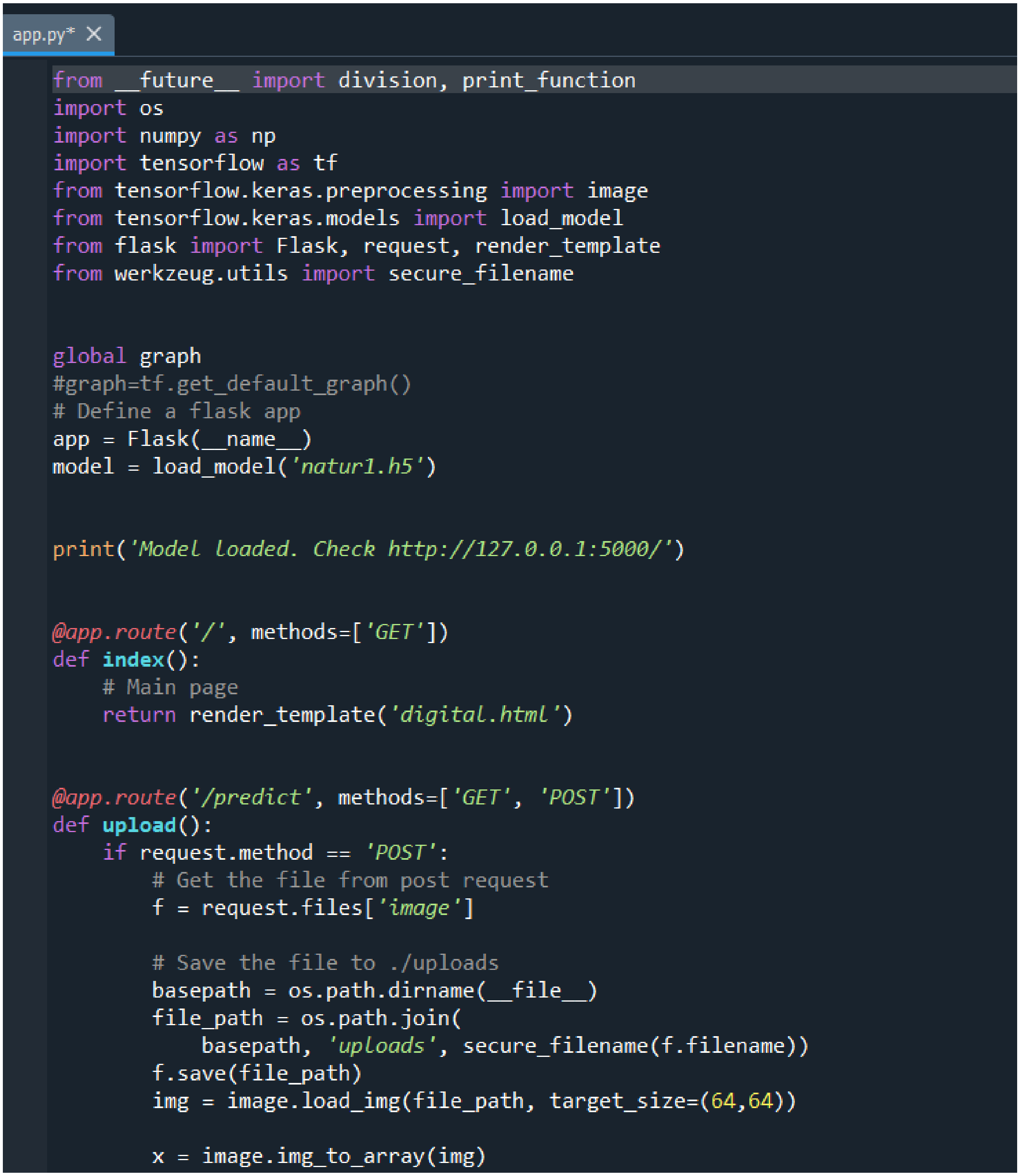
```
        img
Out[15]:
```

```
In [16]: from tensorflow.keras.preprocessing  import image
         path2=r"D:\DigitalNaturalist\datasets\New\traindata\sunflower\sunf.jpg"
         img = image.load_img(path2,target_size = (64,64))

In [17]: type(img)
Out[17]: PIL.Image.Image

In [18]: x = image.img_to_array(img)

In [19]: x.shape
```

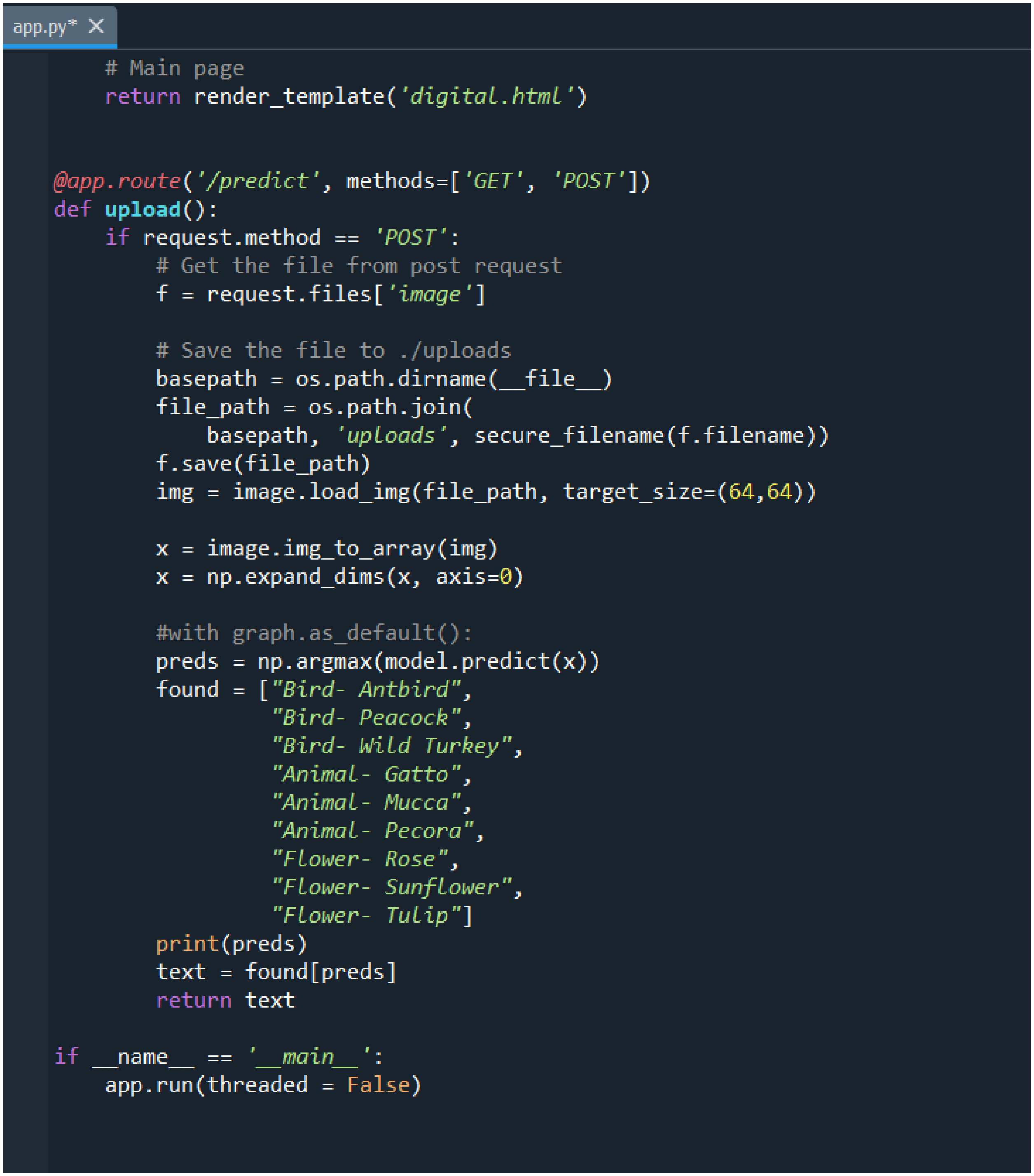


```
In [17]: type(img)
Out[17]: PIL.Image.Image

In [18]: x = image.img_to_array(img)

In [19]: x.shape
```

```
Out[19]: (64, 64, 3)

In [20]: type(x)
Out[20]: numpy.ndarray

In [21]: x = np.expand_dims(x,axis = 0)
         x.shape
Out[21]: (1, 64, 64, 3)

In [22]: pred = np.argmax(model.predict(x))
         pred
         1/1 [==============================] - 0s 415ms/step
Out[22]: 7

In [23]: model.predict(x)
         1/1 [==============================] - 0s 40ms/step
Out[23]: array([[0., 0., 0., 0., 0., 0., 0., 1., 0.]], dtype=float32)
```



```python
from __future__ import division, print_function
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename


global graph
#graph=tf.get_default_graph()
# Define a flask app
app = Flask(__name__)
model = load_model('natur1.h5')


print('Model loaded. Check http://127.0.0.1:5000/')


@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('digital.html')


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(64,64))

        x = image.img_to_array(img)
```



```
    # Main page
    return render_template('digital.html')


@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(64,64))

        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)

        #with graph.as_default():
        preds = np.argmax(model.predict(x))
        found = ["Bird- Antbird",
                 "Bird- Peacock",
                 "Bird- Wild Turkey",
                 "Animal- Gatto",
                 "Animal- Mucca",
                 "Animal- Pecora",
                 "Flower- Rose",
                 "Flower- Sunflower",
                 "Flower- Tulip"]
        print(preds)
        text = found[preds]
        return text

if __name__ == '__main__':
    app.run(threaded = False)
```