**ALG.PY FILE:**

import os

import tensorflow as tf

from keras. preprocessing. Image import ImageDataGenerator from keras. models import Sequential

from keras. layers import Convolution2D, MaxPooling2D from keras. layers import Activation, Dropout, Flatten, Dense

oriole\_dir=os. path.join(r'C:\Users\Win10\Desktop\project\birds\oriole')

gull\_dir=os.path.join(r'C:\Users\Win10\Desktop\project\birds\gull') sparrow\_dir=os.path.join(r'C:\Users\Win10\Desktop\project\birds\sparrow') train\_oriole\_names=os.listdir(oriole\_dir)

print(train\_oriole\_names[:5]) train\_gull\_names=os.listdir(gull\_dir) print(train\_gull\_names[:5]) train\_sparrow\_names=os.listdir(sparrow\_dir) print(train\_sparrow\_names[:5]) batch\_size=60

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

train\_generator = train\_datagen. flow\_from\_directory(r'C:\Users\Win10\Desktop\project\birds',

target\_size=(300, 300), batch\_size=batch\_size,

classes = [ 'gull','oriole','sparrow'], class\_mode='categorical')

target\_size=(300,300)

model = tf.keras.Sequential([

tf.keras.layers.Conv2D(16, (3,3), activation='relu', input\_shape=(300, 300, 3)), tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Dropout(0.2), tf.keras.layers.Conv2D(32, (3,3),activation='relu'), tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Dropout(0.2), tf.keras.layers.Conv2D(64, (3,3),activation='relu'), tf.keras.layers.MaxPooling2D(2,2), tf.keras.layers.Dropout(0.2), tf.keras.layers.Conv2D(128, (3,3),activation='relu'), tf.keras.layers.MaxPooling2D(2,2), tf.keras.layers.Dropout(0.1), tf.keras.layers.Conv2D(256, (3,3),activation='relu'), tf.keras.layers.MaxPooling2D(2,2), tf.keras.layers.Dropout(0.1), tf.keras.layers.Flatten(),

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

])

model. Summary ()

model. Compile (loss='categorical\_crossentropy’, optimizer=tf. keras. optimizers. RMSprop (lr=0. 001),

metrics=['acc'] #RMSprop(lr=0.001) Adam(lr=0.4) total\_sample=train\_generator. n

num\_epochs = 10 model.fit\_generator (train\_generator, steps\_per\_epoch=int(total\_sample/batch\_size),

epochs=num\_epochs, verbose=1)

# serialize model to YAM

from keras. models import model\_from\_json from keras. models import load\_model model\_json=model.to\_json ()

with open ("model. json","w") as json\_file: json\_file.write(model\_json)

model. save\_weights ("model1.h5") print ("saved model to disk")

#### SAMPLE.PY FILE:

from flask import Flask,render\_template,send\_from\_directory,request,flash,jsonify import os

#import samp

#from samp import cnn\_alg

from keras.models import model\_from\_json import tensorflow as tf

import numpy as np

from keras.preprocessing import image from keras.models import load\_model from keras.backend import \* app=Flask( name )

APP\_ROOT=os.path.dirname(os.path.abspath( file )) @app .route('/')

def homepage():

return render\_template('front.html')

@app.route('/', methods=['POST']) def upload():

#image\_names=os.listdir('./images') if request.method=='POST':

f=request. files['file'] f.save(f.filename) fname=f.filename json\_file=open('model.json','r')

loaded\_model\_json=json\_file.read() json\_file.close()

loaded\_model=tf.keras.models.model\_from\_json(loaded\_model\_json) loaded\_model.load\_weights("model1.h5")

print("loaded model1 from disk") test\_image=image.load\_img(fname,target\_size=(300,300,3)) test\_image=image.img\_to\_array(test\_image) test\_image=np.expand\_dims(test\_image,axis=0) result=loaded\_model.predict(test\_image)

#print(result) if result[0][0]==1.0:

res="gull"

elif result[0][1]==1.0: res="oriole"

else:

res="sparrow" image\_names=os.path.join('C:/Users/Win10/Desktop/myproject/images/final.jpg') return render\_template('front.html',image\_names=image\_names,value=res)

@app.route('/<filename>') def send\_image(filename):

return send\_from\_directory('final',filename)

if name ==" main ": #app.run(debug=True) '''json\_file=open('model.json','r') loaded\_model\_json=json\_file.read() json\_file.close()

loaded\_model=tf.keras.models.model\_from\_json(loaded\_model\_json) loaded\_model.load\_weights("model1.h5")

print("loaded model1 from disk") #loaded\_model.\_make\_predict\_function()''' app.run(debug=True,use\_reloader=False)

**FRONT.HTML**

<html>

<head>

<style>

body {

background-image: url\_for("send\_image",filename=image\_names[0]); background-repeat:no-repeat;

background-attachment:fixed; background-size:100% 100%;

}

h1{

border: 1px solid #f1f1f1; margin-bottom: 25px;

}

. submit\_btn {

padding-left:20px;

}

.image{

padding-left:20px;

}

.sub{

height:auto; width:100px;

}

</style>

</head>

<body>

<center>

<form id="upload-form" action="{{url\_for('upload')}}" method="POST" enctype="multipart/form-data">

<div class="image">

<h1>Upload an image:</h1><input id="file-picker" type="file" name="file">

</div>

<br>

<div class="submit\_btn">

<input type="submit" value="upload!" id="upload-button">

</form>

<h1>{{value}}</h1>

</center>

</body>

</html>