EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES

VIDEO ANALYSIS SENDING ALERT MESSAGES

|  |  |
| --- | --- |
| **Date** | 11 November 2022 |
| **Team ID** | PNT2022TMID07051 |
| **Project Name** | Emerging Methods for Early Detection of Forest Fires |

**TWILIO SERVICE:** Twilio is **a customer engagement platform** used by hundreds of thousands of businesses and more than ten million developers worldwide to build unique, personalized experiences for their customers.

**SENDING ALERT MESSAGE:**

1. Purchase a Twilio Phone Number.
2. Create Messaging Service.
3. Set up Notify Service Instance.
4. Gather the Twilio account information.
5. Create a Binding for SMS.
6. Send an SMS Notification.

# Importing The Image Data Generator Library

import keras

from keras.preprocessing.image import ImageDataGenerator

# Define the parameters/arguments for ImageDataGenerator class

train\_datagen=ImageDataGenerator(rescale=1./255,shear\_range=0.2, rotation\_range=180,zoom\_range=0.2, horizontal\_flip=True) test\_datagen=ImageDataGenerator(rescale=1./255)

# Applying ImageDataGenerator functionality to trainset

x\_train=train\_datagen.flow\_from\_directory(r'/content/drive/MyDrive/Dataset/train\_set', target\_size=(128,128),batch\_size=32, class\_mode='binary')

Found 436 images belonging to 2 classes.

# Applying ImageDataGenerator functionality to testset

x\_test=test\_datagen.flow\_from\_directory(r'/content/drive/MyDrive/Dataset/test\_set'

, target\_size=(128,128),batch\_size=32, class\_mode='binary') Found 121 images belonging to 2 classes.

# Import model building libraries

*#To define Linear initialisation import Sequential*

from keras.models import Sequential

*#To add layers import Dense* from keras.layers import Dense

*#To create Convolution kernel import Convolution2D*

from keras.layers import Convolution2D

*#import Maxpooling layer*

from keras.layers import MaxPooling2D

*#import flatten layer*

from keras.layers import Flatten import warnings warnings.filterwarnings('ignore')

# Initializing the model

model=Sequential()

# Add CNN Layer

model.add(Convolution2D(32, (3,3),input\_shape=(128,128,3),activation='relu')) #add maxpooling layer

model.add(MaxPooling2D(pool\_size=(2,2))) #add flatten layer

model.add(Flatten())

# Add Dense Layer

*#add hidden layer* model.add(Dense(150,activation='relu')) *#add output layer* model.add(Dense(1,activation='sigmoid'))

# Configure the learning process

model.compile(loss='binary\_crossentropy',optimizer="adam",metrics=["ac curacy"])

# Train the model

model.fit\_generator(x\_train,steps\_per\_epoch=14,epochs=10,validation\_da ta=x\_test,validation\_steps=4)

Epoch 1/10

14/14 [==============================] - 205s 15s/step - loss: 2.7344 -

accuracy: 0.7454 - val\_loss: 0.2016 - val\_accuracy: 0.9256 Epoch 2/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.8945 -

val\_loss: 0.2290 - val\_accuracy: 0.9339 Epoch 3/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.8922 -

val\_loss: 0.0524 - val\_accuracy: 0.9835 Epoch 4/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.9174 -

val\_loss: 0.1570 - val\_accuracy: 0.9421 Epoch 5/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.9083 -

val\_loss: 0.0767 - val\_accuracy: 0.9752 Epoch 6/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.9335 -

val\_loss: 0.0749 - val\_accuracy: 0.9752 Epoch 7/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.9312 -

val\_loss: 0.1264 - val\_accuracy: 0.9421 Epoch 8/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.9266 -

val\_loss: 0.0652 - val\_accuracy: 0.9835 Epoch 9/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.9358 -

val\_loss: 0.0567 - val\_accuracy: 0.9835 Epoch 10/10

14/14 [==============================] - 20s 1s/step - loss: accuracy: 0.9404 -

val\_loss: 0.0448 - val\_accuracy: 0.9917

0.3267 -

0.2991 -

0.2418 -

0.1984 -

0.1643 -

0.1538 -

0.1732 -

0.1514 -

0.1445 -

<keras.callbacks.History at 0x7f51fdf33610>

# Save The Model

model.save("forest1.h5")

# Predictions

#import load\_model from keras.model from keras.models import load\_model

#import image class from keras

from tensorflow.keras.preprocessing import image #import numpy import numpy as np

#import cv2 import cv2

#load the saved model model = load\_model("forest1.h5")

img=image.load\_img(r'/content/drive/MyDrive/Dataset/test\_set/forest/

0.48007200\_1530881924\_final\_forest.jpg') x=image.img\_to\_array(img)

res = cv2.resize(x, dsize=(128, 128), interpolation=cv2.INTER\_CUBIC) #expand the image shape

x=np.expand\_dims(res,axis=0) pred= model.predict(x)

1/1 [==============================] - 0s 94ms/step pred array([[0.]],

dtype=float32)

# OpenCV For Video Processing

pip install twilio

Looking in indexes: https://pypi.org/simple, https://us- python.pkg.dev/colab- wheels/public/simple/

Collecting twilio

Downloading twilio-7.15.1-py2.py3-none-any.whl (1.4 MB)

ent already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from twilio) (2022.5)

Collecting PyJWT<3.0.0,>=2.0.0

Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB) Requirement already satisfied: requests>=2.0.0 in

/usr/local/lib/python3.7/distpackages (from twilio) (2.23.0) Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (3.0.4)

Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio)

(2.10)

Requirement already satisfied: certifi>=2017.4.17 in

/usr/local/lib/python3.7/distpackages (from requests>=2.0.0->twilio) (2022.9.24) Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in

/usr/local/lib/python3.7/dist-packages (from requests>=2.0.0- >twilio) (1.24.3) Installing collected packages: PyJWT, twilio

Successfully installed PyJWT-2.6.0 twilio-7.15.1 pip install playsound

Looking in indexes: https://pypi.org/simple, https://us- python.pkg.dev/colab- wheels/public/simple/

Collecting playsound

Downloading playsound-1.3.0.tar.gz (7.7 kB) Building wheels for collected packages: playsound

Building wheel for playsound (setup.py) ... e=playsound-1.3.0-py3- none-any.whl size=7035 sha256=e7e96c774a98522e182b59b7b292f0f932097658d8bfce86c922c363f862b0e 2

Stored in directory:

/root/.cache/pip/wheels/ba/f8/bb/ea57c0146b664dca3a0ada4199b0ecb5f9dfc b7b7e22b65ba2

Successfully built playsound

Installing collected packages: playsound Successfully installed playsound-1.3.0 #import opencv library

import cv2 #import numpy import numpy as np

#import image function from keras from keras.preprocessing import image #import load\_model from keras from keras.models import load\_model #import client from twilio API from twilio.rest import Client #import playsound package from playsound import playsound

WARNING:playsound:playsound is relying on another python subprocess. Please use `pip install pygobject` if you want playsound to run more efficiently.

#load the saved model

model=load\_model("forest1.h5") #define video video=cv2.VideoCapture(0) #define the features name=['forest','with fire']

# Creating An Account In Twilio Service

account\_sid='ACfb4e6d0e7b0d25def63044919f1b96e3' auth\_token='f9ae4fc4a617a527da8672e97eefb2d8' client=Client(account\_sid,auth\_token) message=client.messages

\

.create(

body='Forest Fire is detected, stay alert', from\_='+1 302 248 4366',

to='+91 99400 12164'

)

print(message.sid)

SM4aa5a4751b7bcec159dc4c695752293d

# Sending Alert Message

while(1):

sucess, frame= video.read() cv2.imwrite("image.jpg",frame)

img=image.load\_img("image.jpg",target\_size=(64,64)) x=image.img\_to\_array(img) x=np.expand\_dims(x,axis=0) pred=model.predict\_classes(x) p=pred[0] print(pred) cv2.putText(frame,"predicted class="+str(name[p]),(100,100),

cv2.FONT\_HERSHEY\_SIMPLEX,1, (0,0,0), 1) pred = model.predict\_classes(x) if pred[0]==1:

account\_sid='ACfb4e6d0e7b0d25def63044919f1b96e3' auth\_token='f9ae4fc4a617a527da8672e97eefb2d8' client=Client(account\_sid,auth\_token) message=client.messages \

.create(

body='Forest Fire is detected, stay alert', from\_='+1 302 248 4366', to='+91 99400 12164'

)

print(message.sid) print('Fire Detected') print('SMS sent!')

else:

print('No Danger') cv2.imshow("image",frame) if cv2.waitkey(1) & 0xFF == ord('a'): break video.release() cv2.destryoAllWindows()