Server code

*import* numpy *as* np

*from* flask *import* Flask, render\_template,request,jsonify,url\_for,json

*import* pickle

*from* PIL *import* Image

*import* json

*import* tensorflow *as* tf

*from* keras.preprocessing.image *import*  load\_img,img\_to\_array

*from* keras.models *import* load\_model

*import* requests *# to get image from the web*

*import* shutil *# to save it locally*

app = Flask(\_\_name\_\_, *static\_folder*="static")

model=load\_model('model.h5')

@app.route('/',*methods*=['POST','GET'])

def predict():

*if*(request.method=="POST"):

        img = Image.open(request.files['imageFile'].stream)

*# img.show()*

*# img = load\_img("img.jpg")*

*# img = img.resize((64, 64))*

        img.save("img.jpg")

*# img = img\_to\_array(img)*

*# result=model.predict(img)*

        test\_image = tf.keras.utils.load\_img('img.jpg', *target\_size* = (64, 64))

        test\_image =  tf.keras.utils.img\_to\_array(test\_image)

        test\_image = np.expand\_dims(test\_image, *axis* = 0)

        result = model.predict(test\_image)

        print(result)

*return* ""

*# img.save("img.jpeg")*

*return* "Hello from server"

*if* \_\_name\_\_=='\_\_main\_\_':

    app.run(*port*=8000,*host*='0.0.0.0')

Machine learning code:

*import* tensorflow *as* tf

*from* keras.preprocessing.image *import* ImageDataGenerator

train\_datagen = ImageDataGenerator(*rescale* = 1./255,

*shear\_range* = 0.2,

*zoom\_range* = 0.2,

*horizontal\_flip* = True)

training\_set = train\_datagen.flow\_from\_directory('fire\_dataset/training\_set',

*target\_size* = (64, 64),

*batch\_size* = 32,

*class\_mode* = 'binary')

test\_datagen = ImageDataGenerator(*rescale* = 1./255)

test\_set = test\_datagen.flow\_from\_directory('fire\_dataset/test\_set',

*target\_size* = (64, 64),

*batch\_size* = 32,

*class\_mode* = 'binary')

cnn = tf.keras.models.Sequential()

cnn.add(tf.keras.layers.Conv2D(*filters*=32, *kernel\_size*=3, *activation*='relu', *input\_shape*=[64, 64, 3]))

cnn.add(tf.keras.layers.MaxPool2D(*pool\_size*=2, *strides*=2))

cnn.add(tf.keras.layers.Conv2D(*filters*=32, *kernel\_size*=3, *activation*='relu'))

cnn.add(tf.keras.layers.MaxPool2D(*pool\_size*=2, *strides*=2))

cnn.add(tf.keras.layers.Flatten())

cnn.add(tf.keras.layers.Dense(*units*=128, *activation*='relu'))

cnn.add(tf.keras.layers.Dense(*units*=1, *activation*='sigmoid'))

cnn.compile(*optimizer* = 'adam', *loss* = 'binary\_crossentropy', *metrics* = ['accuracy'])

cnn.fit(*x* = training\_set,*validation\_data*=test\_set, *epochs* = 25)