pip install bing-image-downloader

!mkdir images

from bing\_image\_downloader import downloader

downloader.download("road damage",limit=101,output\_dir='dataset\_n')

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.preprocessing import image

from tensorflow.keras.optimizers import RMSprop

import matplotlib.pyplot as plt

import tensorflow as tf

import numpy as np

import cv2

import os

img=image.load\_img('/content/dataset\_n/smooth roads/Image\_1.jpg')

cv2.imread('/content/dataset\_n/smooth roads/Image\_1.jpg').shape

train= ImageDataGenerator(rescale=1/255)

Validation= ImageDataGenerator(rescale=1/255)

downloader.download("smooth roads",limit=60,output\_dir='images')

downloader.download("road damage",limit=60,output\_dir='images')

train\_dataset=train.flow\_from\_directory('/content/dataset\_n',target\_size=(200,200),batch\_size=3,class\_mode='binary')

validation\_dataset=Validation.flow\_from\_directory('/content/images',target\_size=(200,200),batch\_size=3,class\_mode='binary')

model=tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(3,3),activation='relu',input\_shape=(200,200,3)),

                                  tf.keras.layers.MaxPool2D(2,2),

                                  #

                                  tf.keras.layers.Conv2D(32,(3,3),activation='relu'),

                                  tf.keras.layers.MaxPool2D(2,2),

                                  #

                                  tf.keras.layers.Conv2D(64,(3,3),activation='relu'),

                                  tf.keras.layers.MaxPool2D(2,2),

                                  #

                                  tf.keras.layers.Flatten(),

                                  tf.keras.layers.Dense(512,activation='relu'),

                                  tf.keras.layers.Dense(1,activation='sigmoid'),

 ])

model.compile(loss='binary\_crossentropy',

              optimizer=RMSprop(lr=0.001),

              metrics=['accuracy'])

model\_fit=model.fit(train\_dataset,steps\_per\_epoch=5,epochs=30,validation\_data=validation\_dataset)

downloader.download("roads",limit=30,output\_dir='testing')

dir\_path='/content/testing/roads'

for i in os.listdir(dir\_path):

  img=image.load\_img(dir\_path+'/'+i,target\_size=(200,200))

  plt.imshow(img)

  plt.show()

  X=image.img\_to\_array(img)

  X=np.expand\_dims(X,axis=0)

  images=np.vstack([X])

  val=model.predict(images)

  if val==0:

    print("This road needs reparing.")

  else:

    print("This road is in good condition.")