

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
IMG_SIZE = (254, 254)
BATCH_SIZE = 32
```

```
train_datagen = ImageDataGenerator(rescale=1./255,
validation_split=0.2)
train_generator=train_datagen.flow_from_directory(
    '/content/drive/MyDrive/Flood Detection',
    target_size = IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)
```

Found 1600 images belonging to 1 classes.

```
val_generator=train_datagen.flow_from_directory(
    '/content/drive/MyDrive/Flood Detection',
    target_size = IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)
```

Found 400 images belonging to 1 classes.

```
model = keras.Sequential([
    layers.Conv2D(32, (3,3), activation='relu',
input_shape=(254,254,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential

models, prefer using an `Input(shape)` object as the first layer in the model instead.

```
super().__init__(activity_regularizer=activity_regularizer,  
**kwargs)
```

```
model.summary()
```

Model: "sequential"

Layer (type) Param #	Output Shape	
conv2d (Conv2D) 896	(None, 252, 252, 32)	
max_pooling2d (MaxPooling2D) 0	(None, 126, 126, 32)	
conv2d_1 (Conv2D) 18,496	(None, 124, 124, 64)	
max_pooling2d_1 (MaxPooling2D) 0	(None, 62, 62, 64)	
conv2d_2 (Conv2D) 73,856	(None, 60, 60, 128)	
max_pooling2d_2 (MaxPooling2D) 0	(None, 30, 30, 128)	
flatten (Flatten) 0	(None, 115200)	
dense (Dense) 14,745,728	(None, 128)	
dense_1 (Dense) 129	(None, 1)	

Total params: 14,839,105 (56.61 MB)

Trainable params: 14,839,105 (56.61 MB)

Non-trainable params: 0 (0.00 B)

```
model.compile(optimizer='adam', loss='binary_crossentropy',  
metrics=['accuracy'])  
model.fit(train_generator, epochs=5, validation_data=val_generator,  
batch_size=BATCH_SIZE)
```

Epoch 1/5

50/50 ————— 238s 5s/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00

Epoch 2/5

50/50 ————— 243s 5s/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00

Epoch 3/5

50/50 ————— 240s 5s/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00

Epoch 4/5

50/50 ————— 237s 5s/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00

Epoch 5/5

50/50 ————— 245s 5s/step - accuracy: 1.0000 - loss:
0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00

<keras.src.callbacks.history.History at 0x7cd87aa17f10>

```
model.save('/MyDrive/Flood Detection/model.h5')
```

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

```
from tensorflow.keras.models import load_model  
from tensorflow.keras.preprocessing import image  
import matplotlib.pyplot as plt  
import numpy as np  
model = load_model('/MyDrive/Flood Detection/model.h5')  
print("Model Loaded")
```

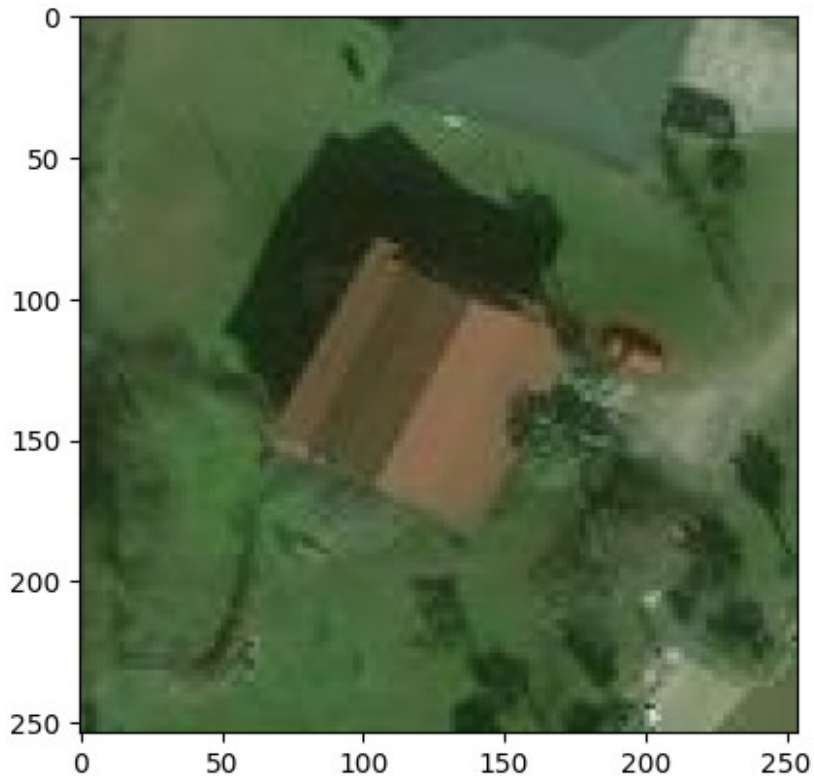
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.

Model Loaded

```

test_image_path="/content/drive/MyDrive/Flood
Detection/test/no_damage/-95.08088199999999_29.824021999999996.jpeg"
img = image.load_img(test_image_path, target_size=(254, 254))
plt.imshow(img)
plt.axis()
plt.show()

```



```

img_array=image.img_to_array(img)
img_array=np.expand_dims(img_array, axis=0)
img_array /=255.

prediction=model.predict(img_array)
print(prediction)

1/1 _____ 0s 74ms/step
[[0.]]

if prediction >= 0.5:
    print("Flood is Affected")
else:
    print("Flood is not Affected")

Flood is not Affected

```