Ramaiah Institute of Technology

(Autonomous Institute Affiliated to VTU)

Department of Information Science and Engineering

DEEP LEARNING - ASSIGNMENT II

Course Code:ISE741

Credits: 3:0:0

Name of the Facilitator: **Dr. Vijaya Kumar .**

Department of ISE

Submitted By:

Mudduluru Charan Teja-1MS19IS072
Obireddy Bhaswanth Reddy-1MS19IS083

Satellite Image Classification as either waterbody,Forest, cloud or dessert.

Introduction

Satellite image classification is the most significant technique used in remote sensing for the computerized study and pattern recognition of satellite information, which is based on diversity structures of the image that involve rigorous validation of the training samples depending on the used classification algorithm.

EXECUTION OF THE PROGRAM :

Step 1: IMPORTING LIBRARIES and INITIALIZING

```
[ ] import tensorflow as tf
    from tensorflow.keras import models,layers
    import matplotlib.pyplot as plt
```

```
[ ] IMAGE_SIZE=256
    BATCH_SIZE=100
    CHANNELS=3
    EPOCHS=5
```

Step-2: Data Collection

```
dataset=tf.keras.preprocessing.image_dataset_from_directory(
    r"C:\Users\chara\OneDrive\Documents\data",
    shuffle=True,
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)
```

```
Found 5600 files belonging to 4 classes.
```

```
class_names=dataset.class_names
class_names
```

```
['cloudy', 'desert', 'green_area', 'water']
```

```
[ ] len(dataset)
```

```
56
```

Satellite image Classification Dataset-RSI-CB256 , This dataset has 4 different classes mixed from Sensors and google map snapshot.

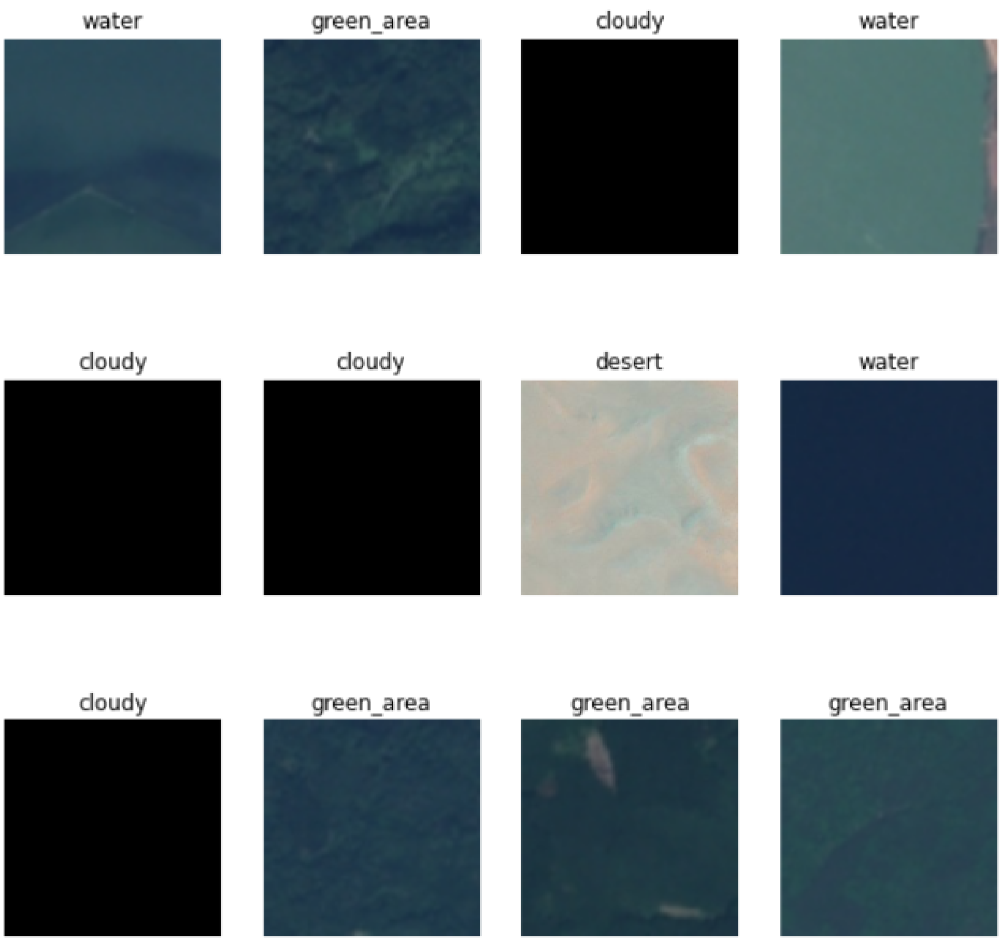
Four classes are:
1)Cloudy
2)Desert
3)Green_area
4)water

Step-3

```python
plt.figure(figsize=(10,10))
for image_batch,label_batch in dataset.take(1):

    print(image_batch.shape)
    print(label_batch.numpy())
    for i in range(12):
        ax=plt.subplot(3,4,i+1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[label_batch[i]])
        plt.axis("off")
```

output:

```
(100, 256, 256, 3)
[3 2 0 3 0 0 1 3 0 2 2 2 1 2 0 0 2 3 0 2 2 2 0 3 3 3 2 0 0 2 2 0 0 3 1 2 3
 2 2 2 3 2 3 0 2 0 0 0 1 2 3 0 3 1 0 0 2 2 1 3 0 3 0 0 0 2 3 2 0 3 3 3 3 0
 3 1 1 1 1 3 2 1 3 0 0 1 0 2 1 2 3 1 1 3 1 3 1 2 3 0]
```



Step-4:Dividing Data into Train Test and Validation

```python
def get_dataset_partitons_tf(ds,train_split=0.8,val_split=0.1,test_split=0.1,shuffle=True,shuffle_size=10000):
    ds_size=len(ds)
    if shuffle:
        ds=ds.shuffle(shuffle_size,seed=12)
    train_size=int(train_split*ds_size)
    val_size=int(val_split*ds_size)
    test_size=int(test_split*ds_size)
    train_ds=dataset.take(train_size)
    test_ds=dataset.skip(train_size)
    val_ds=test_ds.take(val_size)
    test_ds=test_ds.skip(val_size)
    return train_ds,val_ds,test_ds
```

```python
train_ds,val_ds,test_ds=get_dataset_partitons_tf(dataset)
```

```python
len(train_ds)
```

```
44
```

```python
len(val_ds)
```

```
5
```

```python
len(test_ds)
```

```
7
```

Step -5 : Model Building

```
input_shape=(BATCH_SIZE,IMAGE_SIZE,IMAGE_SIZE,CHANNELS)
n_classes=4
model=models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32,(3,3),activation='relu',input_shape=input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64,activation='relu'),
    layers.Dense(n_classes,activation='softmax'),

])

model.build(input_shape=input_shape)
```

Step 6-Model Summary

```python
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential_2 (Sequential)   (100, 256, 256, 3)        0         
                                                                 
 conv2d_6 (Conv2D)           (100, 254, 254, 32)       896       
                                                                 
 max_pooling2d_6 (MaxPooling  (100, 127, 127, 32)      0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (100, 125, 125, 64)       18496     
                                                                 
 max_pooling2d_7 (MaxPooling  (100, 62, 62, 64)        0         
 2D)                                                             
                                                                 
 conv2d_8 (Conv2D)           (100, 60, 60, 64)         36928     
                                                                 
 max_pooling2d_8 (MaxPooling  (100, 30, 30, 64)        0         
 2D)                                                             
                                                                 
 conv2d_9 (Conv2D)           (100, 28, 28, 64)         36928     
                                                                 
 max_pooling2d_9 (MaxPooling  (100, 14, 14, 64)        0         
 2D)                                                             
                                                                 
 conv2d_10 (Conv2D)          (100, 12, 12, 64)         36928     
                                                                 
 max_pooling2d_10 (MaxPoolin  (100, 6, 6, 64)          0         
 g2D)                                                            
                                                                 
 conv2d_11 (Conv2D)          (100, 4, 4, 64)           36928     
                                                                 
 max_pooling2d_11 (MaxPoolin  (100, 2, 2, 64)          0         
 g2D)                                                            
                                                                 
 flatten_1 (Flatten)         (100, 256)                0         
                                                                 
 dense_2 (Dense)             (100, 64)                 16448     
                                                                 
 dense_3 (Dense)             (100, 4)                  260       
                                                                 
=================================================================
Total params: 183,812
Trainable params: 183,812
Non-trainable params: 0
```

```
history=model.fit(
    train_ds,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    verbose=1,
    validation_data=val_ds
)
```

```
Epoch 1/5
44/44 [==============================] - 274s 6s/step - loss: 0.6923 - accuracy: 0.6630 - val_loss: 0.3780 - val_accuracy: 0.8240
Epoch 2/5
44/44 [==============================] - 259s 6s/step - loss: 0.3066 - accuracy: 0.8541 - val_loss: 0.3824 - val_accuracy: 0.8100
Epoch 3/5
44/44 [==============================] - 244s 6s/step - loss: 0.2591 - accuracy: 0.8814 - val_loss: 0.3784 - val_accuracy: 0.7340
Epoch 4/5
44/44 [==============================] - 247s 6s/step - loss: 0.2528 - accuracy: 0.8816 - val_loss: 0.2563 - val_accuracy: 0.9000
Epoch 5/5
44/44 [==============================] - 239s 5s/step - loss: 0.2331 - accuracy: 0.8950 - val_loss: 0.2219 - val_accuracy: 0.9160
```

Step 7-Model Evaluation:

```
scores=model.evaluate(test_ds)
```

```
7/7 [==============================] - 15s 1s/step - loss: 0.1791 - accuracy: 0.9271
```

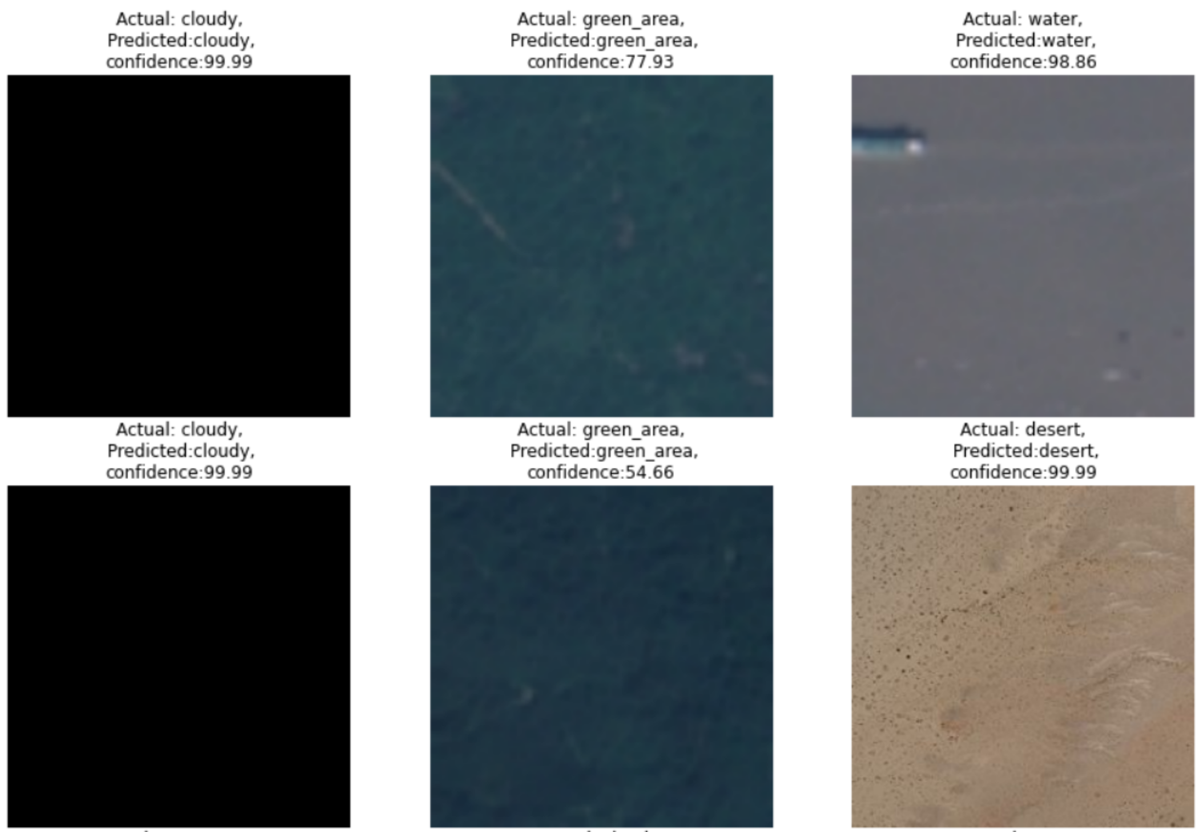
```
scores
```

```
[0.17912176251411438, 0.927142858505249]
```

Step 8-Prediction:

```
def predict(model,img):
    img_array=tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array=tf.expand_dims(img_array,0)

    predictions=model.predict(img_array)

    predicted_class=class_names[np.argmax(predictions[0])]
    confidence=round(100*(np.max(predictions[0])),2)

    return predicted_class,confidence
```

```
plt.figure(figsize=(15,15))
for images,labels in test_ds.take(1):
    for i in range(9):
        ax=plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype('uint8'))

        predicted_class,confidence=predict(model,images[i].numpy())
        actual_class=class_names[labels[i]]

        plt.title(f"Actual: {actual_class},\n Predicted:{predicted_class},\nconfidence:{confidence}")

        plt.axis('off')
```



conclusion

Regions in the satellite images are not understandable every time by eye site. So Satellite image classification is very helpful to gather information about a particular region. Similar structures of different class of images and environmental variations makes the work difficult. Images used for testing is checked to determine the accuracy.