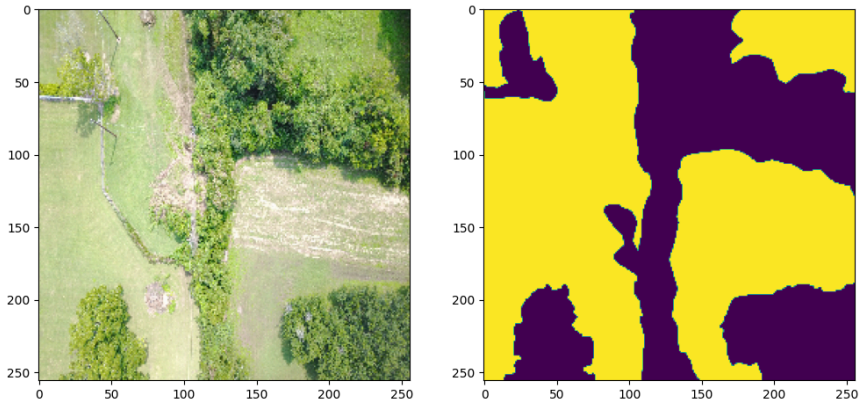
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np

# Load npz file
train_images = np.load('/content/drive/MyDrive/processed_images.npy')
train_masks = np.load('/content/drive/MyDrive/processed_masks.npy')
test_images = np.load('/content/drive/MyDrive/processed_images_test.npy')
test_masks = np.load('/content/drive/MyDrive/processed_masks_test.npy')

# Access the arrays in the npz file
print("train images:",train_images.shape)
print("train masks:",train_masks.shape)
print("test images:",test_images.shape)
print("test masks:",test_masks.shape)
```

```
train images: (450, 256, 256, 3)
train masks: (450, 256, 256)
test images: (448, 256, 256, 3)
test masks: (448, 256, 256)
```

```
import random
import numpy as np
import matplotlib.pyplot as plt
image_number = random.randint(0, len(train_images))
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.imshow(np.reshape(train_images[image_number], (256,256, 3)))
plt.subplot(122)
plt.imshow(np.reshape(train_masks[image_number], (256,256, 1)))
plt.show()
```




```
unique_labels = np.unique(train_masks)

# Count the number of unique labels
num_unique_labels = len(unique_labels)

print("Number of unique labels:", num_unique_labels)
```

```
Number of unique labels: 10
```

```
train_masks =  np.expand_dims(train_masks, axis=3)
```

```
train_masks.shape
```

```
(450, 256, 256, 1)
```

```
n_classes = len(np.unique(train_masks))
from keras.utils import to_categorical
train_masks_cat = to_categorical(train_masks, num_classes=n_classes)
train_masks_cat.shape
```

```
(450, 256, 256, 10)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train_images, train_masks_cat, test_size = 0.20, random_state = 42)
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Dropout, concatenate, Conv2DTranspose
from tensorflow.keras.models import Model
from tensorflow.keras.applications import VGG16

def create_unet_vgg16(input_shape, num_classes):
    # Load pre-trained VGG16 without the top layers
    vgg16_base = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)

    # Encoder
    encoder = vgg16_base.get_layer('block5_conv3').output

    # Decoder
    decoder = Conv2D(512, (3, 3), activation='relu', padding='same')(encoder)
    decoder = Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(decoder)
    decoder = concatenate([decoder, vgg16_base.get_layer('block4_conv3').output], axis=3)
    decoder = Conv2D(256, (3, 3), activation='relu', padding='same')(decoder)
    decoder = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(decoder)
    decoder = concatenate([decoder, vgg16_base.get_layer('block3_conv3').output], axis=3)
    decoder = Conv2D(128, (3, 3), activation='relu', padding='same')(decoder)
    decoder = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(decoder)
    decoder = concatenate([decoder, vgg16_base.get_layer('block2_conv2').output], axis=3)
    decoder = Conv2D(64, (3, 3), activation='relu', padding='same')(decoder)
    decoder = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(decoder)
    decoder = concatenate([decoder, vgg16_base.get_layer('block1_conv2').output], axis=3)
    decoder = Conv2D(32, (3, 3), activation='relu', padding='same')(decoder)

    # Output layer
    outputs = Conv2D(num_classes, (1, 1), activation='softmax')(decoder)

    # Create model
    model = Model(inputs=vgg16_base.input, outputs=outputs)

    return model

# Example usage
input_shape = (256, 256, 3)
num_classes = 10  # Number of classes

# Create model
model = create_unet_vgg16(input_shape, num_classes)

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Print model summary
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernel
58889256/58889256 [==============================] - 0s 0us/step
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0         []

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792      ['input_1[0][0]']

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928     ['block1_conv1[0][0]']

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0         ['block1_conv2[0][0]']

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856     ['block1_pool[0][0]']

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584    ['block2_conv1[0][0]']

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0         ['block2_conv2[0][0]']

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168    ['block2_pool[0][0]']

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080    ['block3_conv1[0][0]']
```

```
block3_conv3 (Conv2D)       (None, 64, 64, 256)          590080    ['block3_conv2[0][0]']

block3_pool (MaxPooling2D)  (None, 32, 32, 256)          0         ['block3_conv3[0][0]']

block4_conv1 (Conv2D)       (None, 32, 32, 512)          1180160   ['block3_pool[0][0]']

block4_conv2 (Conv2D)       (None, 32, 32, 512)          2359808   ['block4_conv1[0][0]']

block4_conv3 (Conv2D)       (None, 32, 32, 512)          2359808   ['block4_conv2[0][0]']

block4_pool (MaxPooling2D)  (None, 16, 16, 512)          0         ['block4_conv3[0][0]']

block5_conv1 (Conv2D)       (None, 16, 16, 512)          2359808   ['block4_pool[0][0]']

block5_conv2 (Conv2D)       (None, 16, 16, 512)          2359808   ['block5_conv1[0][0]']

block5_conv3 (Conv2D)       (None, 16, 16, 512)          2359808   ['block5_conv2[0][0]']

conv2d (Conv2D)             (None, 16, 16, 512)          2359808   ['block5_conv3[0][0]']

conv2d_transpose (Conv2DTr  (None, 32, 32, 256)          524544    ['conv2d[0][0]']
anspose)

concatenate (Concatenate)   (None, 32, 32, 768)          0         ['conv2d_transpose[0][0]',
                                                                     'block4_conv3[0][0]']

conv2d_1 (Conv2D)           (None, 32, 32, 256)          1769728   ['concatenate[0][0]']

conv2d_transpose_1 (Conv2D  (None, 64, 64, 128)          131200    ['conv2d_1[0][0]']
Transpose)

concatenate_1 (Concatenate  (None, 64, 64, 384)          0         ['conv2d_transpose_1[0][0]',
```

```
# Train the model
history1 = model.fit(X_train, y_train,
                     batch_size = 16,
                     verbose=1,
                     epochs=10,
                     validation_data=(X_test, y_test),
                     shuffle=False)
```

```
Epoch 1/10
23/23 [==============================] - 119s 5s/step - loss: 0.8845 - accuracy: 0.6935 - val_loss: 0.9541 - val_accuracy: 0.7077
Epoch 2/10
23/23 [==============================] - 119s 5s/step - loss: 0.8487 - accuracy: 0.7048 - val_loss: 0.9099 - val_accuracy: 0.7143
Epoch 3/10
23/23 [==============================] - 118s 5s/step - loss: 0.8142 - accuracy: 0.7172 - val_loss: 0.8668 - val_accuracy: 0.7283
Epoch 4/10
23/23 [==============================] - 118s 5s/step - loss: 0.7797 - accuracy: 0.7310 - val_loss: 0.9000 - val_accuracy: 0.7248
Epoch 5/10
23/23 [==============================] - 118s 5s/step - loss: 0.7771 - accuracy: 0.7333 - val_loss: 0.9109 - val_accuracy: 0.6918
Epoch 6/10
23/23 [==============================] - 119s 5s/step - loss: 0.7601 - accuracy: 0.7376 - val_loss: 0.8487 - val_accuracy: 0.7312
Epoch 7/10
23/23 [==============================] - 119s 5s/step - loss: 0.7360 - accuracy: 0.7472 - val_loss: 0.7901 - val_accuracy: 0.7507
Epoch 8/10
23/23 [==============================] - 120s 5s/step - loss: 0.7177 - accuracy: 0.7552 - val_loss: 0.7960 - val_accuracy: 0.7488
Epoch 9/10
23/23 [==============================] - 120s 5s/step - loss: 0.7060 - accuracy: 0.7592 - val_loss: 0.7732 - val_accuracy: 0.7593
Epoch 10/10
23/23 [==============================] - 120s 5s/step - loss: 0.7696 - accuracy: 0.7375 - val_loss: 0.8313 - val_accuracy: 0.7337
```

```
  Evaluate the model
loss, accuracy = model.evaluate(X_test,y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
3/3 [==============================] - 5s 1s/step - loss: 0.8313 - accuracy: 0.7337
Test Loss: 0.831282377243042
Test Accuracy: 0.7337412238121033
```

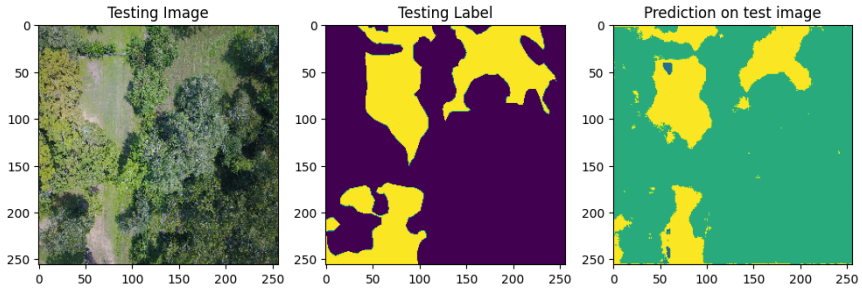
```
#IOU
y_pred=model.predict(X_test)
y_pred_argmax=np.argmax(y_pred, axis=3)
y_test_argmax=np.argmax(y_test, axis=3)


#Using built in keras function for IoU
from keras.metrics import MeanIoU
n_classes = 10
IOU_keras = MeanIoU(num_classes=n_classes)
IOU_keras.update_state(y_test_argmax, y_pred_argmax)
print("Mean IoU =", IOU_keras.result().numpy())
```

```
3/3 [==============================] - 4s 1s/step
Mean IoU = 0.27470604
```

```
import random
test_img_number = random.randint(0, len(X_test))
test_img = X_test[test_img_number]
ground_truth=y_test_argmax[test_img_number]
#test_img_norm=test_img[:,:,0][:,:,None]
test_img_input=np.expand_dims(test_img, 0)
prediction = (model.predict(test_img_input))
predicted_img=np.argmax(prediction, axis=3)[0,:,:]
```

```
1/1 [==============================] - 0s 115ms/step
```

```
plt.figure(figsize=(12, 8))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img)
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(ground_truth)
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(predicted_img)
plt.show()
```




```
# Alternatively, you can specify a filename for the model
#model.save('/content/drive/MyDrive/Engineering/')
```