Assignment 4: TKO_7096-3001 Computer Vision and Sensor Fusion

Goal: develop a RGB-Depth fusion architecture for semantic segmentation based on Fully Convolutional Network (FCN) .

Deadline: 26.03.2024 at 24:00.

- Imports go here

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

from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input, Conv2D, Dropout,
Concatenate, Conv2DTranspose, Reshape, Activation
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.optimizers import SGD
import tensorflow as tf
import os
import random
import numpy as np
import cv2
import matplotlib.pyplot as plt
from tensorflow.keras.utils import Sequence
```

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

Load the dataset and Ground-truth

```
# Define directories
train_rgb_dir = '/content/drive/MyDrive/Cv/dataset/train/rgb'
train_depth_dir = '/content/drive/MyDrive/Cv/dataset/train/depth'
train_label_dir = '/content/drive/MyDrive/Cv/dataset/train/label'
test_rgb_dir = '/content/drive/MyDrive/Cv/dataset/test/rgb'
test_depth_dir = '/content/drive/MyDrive/Cv/dataset/test/depth'
test_label_dir = '/content/drive/MyDrive/Cv/dataset/test/label'


val_rgb_dir = '/content/drive/MyDrive/Cv/dataset/validation/rgb'
val_depth_dir = '/content/drive/MyDrive/Cv/dataset/validation/depth'
val_label_dir = '/content/drive/MyDrive/Cv/dataset/validation/label'
```

- Dataset consists of 1100 (per modality) images of road scenes. It is divided into train (600 images), test (200 images) and validation (300 images) datasets.
- Change the size of all images into 256*256.
- Converting the labels into one hot encoding
- Create a DataLoader for loading the files when training the model.

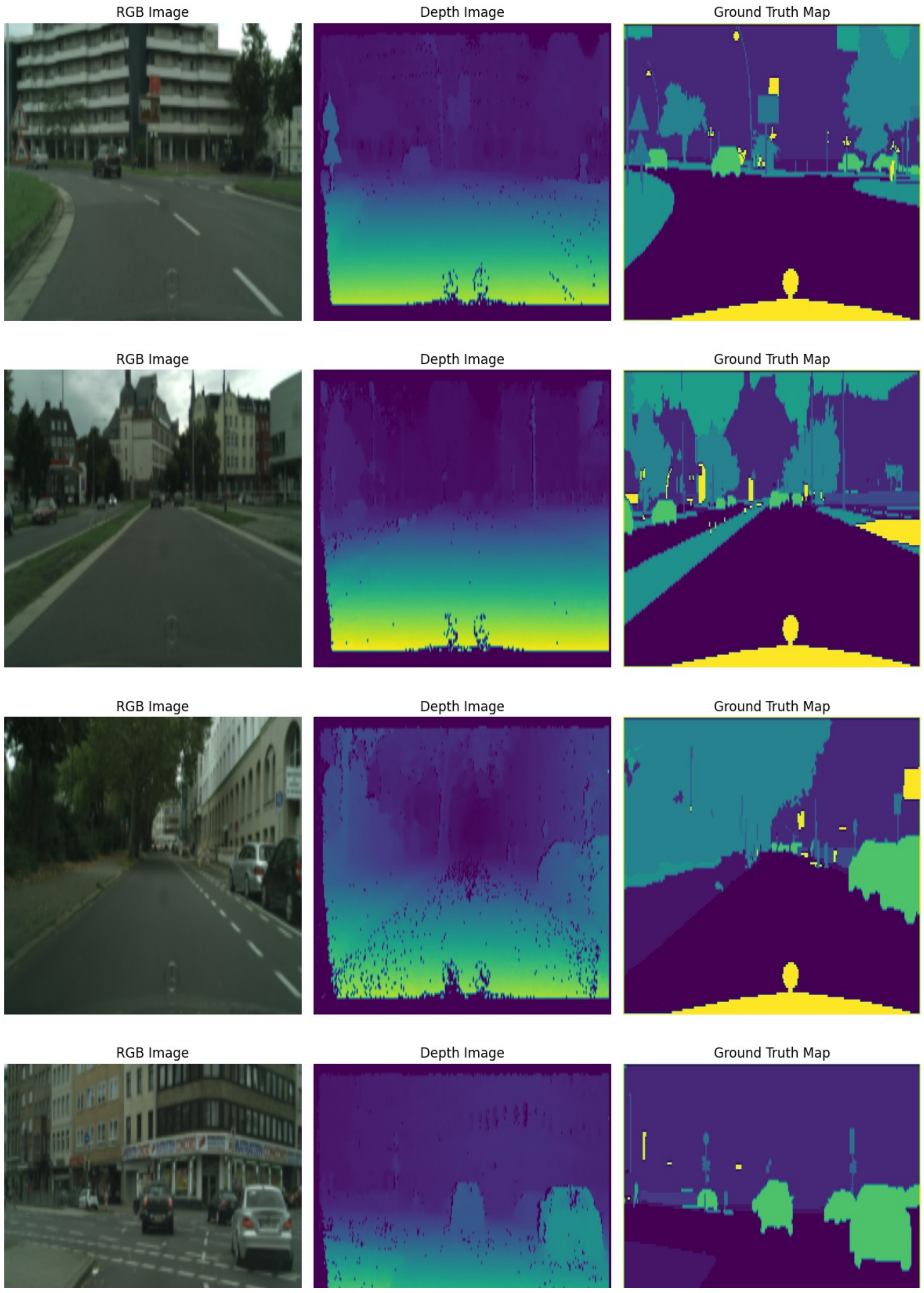
```python
# Define the number of classes
num_classes = 19

# Define a custom DataLoader class. Here I have also done necessary
image resizing and one hot encoded labelling.
class DataLoader(Sequence):
    def __init__(self, rgb_dir, depth_dir, label_dir, image_size,
num_classes, batch_size=32, num_images=None):
        """
        Constructor method to initialize DataLoader object.

        Parameters:
        - rgb_dir: Directory containing RGB images.
        - depth_dir: Directory containing depth images.
        - label_dir: Directory containing label images.
        - image_size: Tuple representing the size of images (height,
width, channels).
        - num_classes: Number of classes for segmentation.
        - batch_size: Size of the batch for training.
        - num_images: Total number of images in the dataset.
        """
        self.rgb_dir = rgb_dir
        self.depth_dir = depth_dir
        self.label_dir = label_dir
        self.image_size = image_size
        self.num_classes = num_classes
        self.batch_size = batch_size
        self.num_samples = num_images if num_images is not None else
len(os.listdir(label_dir))
        self.labels = self.load_labels()

    def __len__(self):
        """
        Method to calculate the number of batches per epoch.
        """
        return int(np.ceil(self.num_samples / self.batch_size))

    def __getitem__(self, idx):
        """
        Method to get a batch of data.

        Parameters:
        - idx: Index of the batch.

        Returns:
        - Tuple containing RGB images, depth images, and labels for
the batch.
        """
        start_index = idx * self.batch_size
        end_index = min((idx + 1) * self.batch_size, self.num_samples)
        batch_indices = range(start_index, end_index)

        # Adjust indices for validation data
        if self.label_dir == val_label_dir:
            batch_label_files = [os.path.join(self.label_dir,
str(index + 200) + '.npy') for index in batch_indices]
            batch_rgb_files = [os.path.join(self.rgb_dir, str(index +
200) + '.npy') for index in batch_indices]
            batch_depth_files = [os.path.join(self.depth_dir,
str(index + 200) + '.npy') for index in batch_indices]
        else:
            batch_label_files = [os.path.join(self.label_dir,
str(index) + '.npy') for index in batch_indices]
            batch_rgb_files = [os.path.join(self.rgb_dir, str(index) +
'.npy') for index in batch_indices]
            batch_depth_files = [os.path.join(self.depth_dir,
str(index) + '.npy') for index in batch_indices]

        return self.__generate_data(batch_rgb_files,
batch_depth_files, batch_label_files)

    def load_labels(self):
        """
        Method to load label images.
        """
        labels = []
        label_files = sorted(os.listdir(self.label_dir))
        for file in label_files:
            label = np.load(os.path.join(self.label_dir, file))
            labels.append(label)
        return np.stack(labels)

    def __generate_data(self, batch_rgb_files, batch_depth_files,
batch_label_files):
        """
        Method to generate data for a batch.

        Parameters:
        - batch_rgb_files: List of file paths for RGB images in the
batch.
        - batch_depth_files: List of file paths for depth images in
the batch.
        - batch_label_files: List of file paths for label images in
the batch.

        Returns:
        - Tuple containing arrays of RGB images, depth images, and
one-hot encoded labels for the batch.
        """
        batch_rgb_imgs = []
        batch_depth_imgs = []
        batch_labels_encoded = []

        for rgb_file, depth_file, label_file in zip(batch_rgb_files,
batch_depth_files, batch_label_files):
            rgb_img = np.load(rgb_file)
            depth_img = np.load(depth_file)
            label = np.load(label_file)

            # Resizing rgb and depth images into the desired format
(256, 256)

            rgb_img_resized = cv2.resize(rgb_img, (self.image_size[1],
self.image_size[0]))
            depth_img_resized = cv2.resize(depth_img,
(self.image_size[1], self.image_size[0]))
            depth_img_resized = np.repeat(depth_img_resized[:, :,
np.newaxis], 3, axis=2)

            # Resizing labels into the desired format (256, 256)

            label_resized = cv2.resize(label, (self.image_size[1],
self.image_size[0]))

            # COmversion into one hot encoded vectors
            label_encoded =
tf.keras.utils.to_categorical(label_resized,
num_classes=self.num_classes)

            batch_rgb_imgs.append(rgb_img_resized)
            batch_depth_imgs.append(depth_img_resized)
            batch_labels_encoded.append(label_encoded)

        return [np.array(batch_rgb_imgs), np.array(batch_depth_imgs)],
np.array(batch_labels_encoded)


# Create DataLoader instances for training, testing, and validation
datasets
train_loader = DataLoader(train_rgb_dir, train_depth_dir,
train_label_dir, image_size=(256, 256, 3), num_classes=num_classes,
batch_size=12, num_images=600)
test_loader = DataLoader(test_rgb_dir, test_depth_dir, test_label_dir,
image_size=(256, 256, 3), num_classes=num_classes, batch_size=6,
num_images=200)
val_loader = DataLoader(val_rgb_dir, val_depth_dir, val_label_dir,
image_size=(256, 256, 3), num_classes=num_classes, batch_size=6,
num_images=100)
```

```python
# Print a message indicating that DataLoader instances are created
print("Train, Test, Validation Loaders are made")
```

```
Train, Test, Validation Loaders are made
```

- Visualize the data you have prepared

```python
# Define the number of rows to visualize
num_rows_to_visualize = 4

# Get the first batch from the train_loader
(batch_rgb, batch_depth), batch_gt = train_loader[0]

# Set the size of the figure
plt.figure(figsize=(12, 18))

# Plot each batch
for i in range(min(num_rows_to_visualize, len(batch_rgb))):
    # Plot RGB image
    plt.subplot(num_rows_to_visualize, 3, i * 3 + 1)  # Define subplot
position for RGB image
    plt.imshow(batch_rgb[i])  # Display RGB image
    plt.title('RGB Image')  # Set title for RGB image
    plt.axis('off')  # Turn off axis for RGB image

    # Plot depth image
    plt.subplot(num_rows_to_visualize, 3, i * 3 + 2)  # Define subplot
position for depth image
    plt.imshow(batch_depth[i][:, :, 0])  # Display depth image
    plt.title('Depth Image')  # Set title for depth image
    plt.axis('off')  # Turn off axis for depth image

    # Plot Ground Truth Map
    plt.subplot(num_rows_to_visualize, 3, i * 3 + 3)  # Define subplot
position for ground truth map
    gt_map_single_channel = np.argmax(batch_gt[i], axis=-1)  # Get the
ground truth label map
    plt.imshow(gt_map_single_channel, vmin=0, vmax=18)  # Display
ground truth label map
    plt.title('Ground Truth Map')  # Set title for ground truth map
    plt.axis('off')  # Turn off axis for ground truth map

plt.tight_layout()  # Adjust layout to prevent overlap of subplots
plt.show()  # Show the plot
```


RGB Image
Depth Image
Ground Truth Map
RGB Image
Depth Image
Ground Truth Map
RGB Image
Depth Image
Ground Truth Map
RGB Image
Depth Image
Ground Truth Map

Define a Fully Convolutional Network (FCN) for image segmentaion by fusing RGB and depth images. The network consists of two sterams which each stream has following layers:

```
1. Use the pretrained ResNet50 on imageNet
2. Add two Conv layers with 128 and 256 nodes, respectively. Kernel
size (3,3), stride (1,1)
3. Top of the Conv layers, add dropout layer with 0.2
4. Concatenate two streams.
5. Add a transposed convolution layer (Conv2DTranspose)  with Kernel
size (64,64), stride (32,32)
6. Add a softmax activation layer
```

You can find the model summary and structure in the PDF file.

```python
# Define input shapes
input_rgb = Input(shape=(256, 256, 3), name='input_rgb')
input_depth = Input(shape=(256, 256, 3), name='input_depth')

# Number of classes
num_classes = 19

# Stream 1: Pretrained ResNet50 on ImageNet for RGB
pre_trained_model_rgb = ResNet50(weights='imagenet',
include_top=False, input_tensor=input_rgb)

# Stream 2: Pretrained ResNet50 on ImageNet for NIR
pre_trained_model_depth = ResNet50(weights='imagenet',
include_top=False, input_tensor=input_depth)

# Rename layers of the second model to avoid naming conflicts
for layer in pre_trained_model_depth.layers:
    layer._name = layer.name + '_depth'

# Rename layers of the second model to avoid naming conflicts
for layer in pre_trained_model_rgb.layers:
    layer._name = layer.name + 'rgb'

# Additional convolutional layers for each stream
conv2d_rgb_1 = Conv2D(128, (3, 3), strides=(1, 1), activation='relu',
padding='same', name='conv2d_rgb_1')(pre_trained_model_rgb.output)
conv2d_depth_1 = Conv2D(128, (3, 3), strides=(1, 1),
activation='relu', padding='same', name='conv2d_depth_1')
(pre_trained_model_depth.output)
```

```
conv2d_rgb_2 = Conv2D(256, (3, 3), strides=(1, 1), activation='relu',
padding='same', name='conv2d_rgb_2')(conv2d_rgb_1)
conv2d_depth_2 = Conv2D(256, (3, 3), strides=(1, 1),
activation='relu', padding='same', name='conv2d_depth_2')
(conv2d_depth_1)

# Dropout layers
dropout_rgb = Dropout(0.2, name='dropout_rgb')(conv2d_rgb_2)
dropout_depth = Dropout(0.2, name='dropout_depth')(conv2d_depth_2)

# Concatenate the outputs of both streams
concatenated = Concatenate(name='concatenate')([dropout_rgb,
dropout_depth])

# Transposed convolutional layer
conv2d_transpose = Conv2DTranspose(19, (64, 64), strides=(32, 32),
padding='same', name='conv2d_transpose')(concatenated)

# Reshape layer
reshaped_layer = Reshape((256, 256, num_classes), name='reshape')
(conv2d_transpose)

# Softmax activation layer
activation = Activation('softmax', name='activation')(reshaped_layer)

# Create the model
model = Model(inputs=[input_rgb, input_depth], outputs=activation)

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/
resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 1s 0us/step
Model: "model"
______________________________________________________________________
____________________________
 Layer (type)                Output Shape                 Param #
Connected to
======================================================================
============================
 input_rgbrgb (InputLayer)   [(None, 256, 256, 3)]        0         []



 input_depth_depth (InputLa  [(None, 256, 256, 3)]        0         []

 yer)
 conv1_padrgb (ZeroPadding2  (None, 262, 262, 3)          0
['input_rgbrgb[0][0]']
 D)



 conv1_pad_depth (ZeroPaddi  (None, 262, 262, 3)          0
['input_depth_depth[0][0]']
 ng2D)



 conv1_convrgb (Conv2D)      (None, 128, 128, 64)         9472
['conv1_padrgb[0][0]']


 conv1_conv_depth (Conv2D)   (None, 128, 128, 64)         9472
['conv1_pad_depth[0][0]']


 conv1_bnrgb (BatchNormaliz  (None, 128, 128, 64)         256
['conv1_convrgb[0][0]']
 ation)



 conv1_bn_depth (BatchNorma  (None, 128, 128, 64)         256
['conv1_conv_depth[0][0]']
 lization)



 conv1_relurgb (Activation)  (None, 128, 128, 64)         0
['conv1_bnrgb[0][0]']


 conv1_relu_depth (Activati  (None, 128, 128, 64)         0
['conv1_bn_depth[0][0]']
 on)



 pool1_padrgb (ZeroPadding2  (None, 130, 130, 64)         0
['conv1_relurgb[0][0]']
 D)



 pool1_pad_depth (ZeroPaddi  (None, 130, 130, 64)         0
['conv1_relu_depth[0][0]']
 ng2D)



 pool1_poolrgb (MaxPooling2  (None, 64, 64, 64)           0
['pool1_padrgb[0][0]']
 D)



 pool1_pool_depth (MaxPooli  (None, 64, 64, 64)           0
['pool1_pad_depth[0][0]']
 ng2D)



 conv2_block1_1_convrgb (Co  (None, 64, 64, 64)           4160
['pool1_poolrgb[0][0]']
 nv2D)



 conv2_block1_1_conv_depth   (None, 64, 64, 64)           4160
['pool1_pool_depth[0][0]']
 (Conv2D)



 conv2_block1_1_bnrgb (Batc  (None, 64, 64, 64)           256
['conv2_block1_1_convrgb[0][0]
 hNormalization)                                                   ']



 conv2_block1_1_bn_depth (B  (None, 64, 64, 64)           256
['conv2_block1_1_conv_depth[0]
 atchNormalization)
[0]']



 conv2_block1_1_relurgb (Ac  (None, 64, 64, 64)           0
['conv2_block1_1_bnrgb[0][0]']
 tivation)



 conv2_block1_1_relu_depth   (None, 64, 64, 64)           0
['conv2_block1_1_bn_depth[0][0

(Activation)                                                      ]']
```

```
 conv2_block1_2_convrgb (Co  (None, 64, 64, 64)           36928
['conv2_block1_1_relurgb[0][0]
 nv2D)                                                              ']



 conv2_block1_2_conv_depth   (None, 64, 64, 64)           36928
['conv2_block1_1_relu_depth[0]
 (Conv2D)
[0]']


 conv2_block1_2_bnrgb (Batc  (None, 64, 64, 64)           256
['conv2_block1_2_convrgb[0][0]
 hNormalization)                                                    ']



 conv2_block1_2_bn_depth (B  (None, 64, 64, 64)           256
['conv2_block1_2_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block1_2_relurgb (Ac  (None, 64, 64, 64)           0
['conv2_block1_2_bnrgb[0][0]']
 tivation)



 conv2_block1_2_relu_depth   (None, 64, 64, 64)           0
['conv2_block1_2_bn_depth[0][0

(Activation)                                                       ]']



 conv2_block1_0_convrgb (Co  (None, 64, 64, 256)          16640
['pool1_poolrgb[0][0]']
 nv2D)



 conv2_block1_3_convrgb (Co  (None, 64, 64, 256)          16640
['conv2_block1_2_relurgb[0][0]
 nv2D)                                                              ']
 conv2_block1_0_conv_depth   (None, 64, 64, 256)          16640
['pool1_pool_depth[0][0]']
 (Conv2D)



 conv2_block1_3_conv_depth   (None, 64, 64, 256)          16640
['conv2_block1_2_relu_depth[0]
 (Conv2D)
[0]']


 conv2_block1_0_bnrgb (Batc  (None, 64, 64, 256)          1024
['conv2_block1_0_convrgb[0][0]
 hNormalization)                                                    ']



 conv2_block1_3_bnrgb (Batc  (None, 64, 64, 256)          1024
['conv2_block1_3_convrgb[0][0]
 hNormalization)                                                    ']



 conv2_block1_0_bn_depth (B  (None, 64, 64, 256)          1024
['conv2_block1_0_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block1_3_bn_depth (B  (None, 64, 64, 256)          1024
['conv2_block1_3_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block1_addrgb (Add)   (None, 64, 64, 256)          0
['conv2_block1_0_bnrgb[0][0]',

'conv2_block1_3_bnrgb[0][0]']


 conv2_block1_add_depth (Ad  (None, 64, 64, 256)          0
['conv2_block1_0_bn_depth[0][0

d)                                                                 ]',


'conv2_block1_3_bn_depth[0][0
                                                                    ]'
]


 conv2_block1_outrgb (Activ  (None, 64, 64, 256)          0
['conv2_block1_addrgb[0][0]']
 ation)



 conv2_block1_out_depth (Ac  (None, 64, 64, 256)          0
['conv2_block1_add_depth[0][0]
 tivation)                                                                          ']



 conv2_block2_1_convrgb (Co  (None, 64, 64, 64)           16448
['conv2_block1_outrgb[0][0]']
 nv2D)



 conv2_block2_1_conv_depth   (None, 64, 64, 64)           16448
['conv2_block1_out_depth[0][0]
 (Conv2D)                                                                           ']



 conv2_block2_1_bnrgb (Batc  (None, 64, 64, 64)           256
['conv2_block2_1_convrgb[0][0]
 hNormalization)                                                                    ']



 conv2_block2_1_bn_depth (B  (None, 64, 64, 64)           256
['conv2_block2_1_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block2_1_relurgb (Ac  (None, 64, 64, 64)           0
['conv2_block2_1_bnrgb[0][0]']
 tivation)



 conv2_block2_1_relu_depth   (None, 64, 64, 64)           0
['conv2_block2_1_bn_depth[0][0

(Activation)                                                                       ]']
```

```
 conv2_block2_2_convrgb (Co  (None, 64, 64, 64)           36928
['conv2_block2_1_relurgb[0][0]
 nv2D)                                                                 ']



 conv2_block2_2_conv_depth   (None, 64, 64, 64)           36928
['conv2_block2_1_relu_depth[0]
 (Conv2D)
[0]']


 conv2_block2_2_bnrgb (Batc  (None, 64, 64, 64)           256
['conv2_block2_2_convrgb[0][0]
 hNormalization)                                                       ']



 conv2_block2_2_bn_depth (B  (None, 64, 64, 64)           256
['conv2_block2_2_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block2_2_relurgb (Ac  (None, 64, 64, 64)           0
['conv2_block2_2_bnrgb[0][0]']
 tivation)



 conv2_block2_2_relu_depth   (None, 64, 64, 64)           0
['conv2_block2_2_bn_depth[0][0

(Activation)                                                          ]']



 conv2_block2_3_convrgb (Co  (None, 64, 64, 256)          16640
['conv2_block2_2_relurgb[0][0]
 nv2D)                                                                 ']



 conv2_block2_3_conv_depth   (None, 64, 64, 256)          16640
['conv2_block2_2_relu_depth[0]
 (Conv2D)
[0]']


 conv2_block2_3_bnrgb (Batc  (None, 64, 64, 256)          1024
['conv2_block2_3_convrgb[0][0]
 hNormalization)                                                            ']



 conv2_block2_3_bn_depth (B  (None, 64, 64, 256)          1024
['conv2_block2_3_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block2_addrgb (Add)   (None, 64, 64, 256)          0
['conv2_block1_outrgb[0][0]',

'conv2_block2_3_bnrgb[0][0]']


 conv2_block2_add_depth (Ad  (None, 64, 64, 256)          0
['conv2_block1_out_depth[0][0]
 d)                                                                         ',


'conv2_block2_3_bn_depth[0][0
                                                                            ]'
]


 conv2_block2_outrgb (Activ  (None, 64, 64, 256)          0
['conv2_block2_addrgb[0][0]']
 ation)



 conv2_block2_out_depth (Ac  (None, 64, 64, 256)          0
['conv2_block2_add_depth[0][0]
 tivation)                                                                  ']



 conv2_block3_1_convrgb (Co  (None, 64, 64, 64)           16448
['conv2_block2_outrgb[0][0]']
 nv2D)



 conv2_block3_1_conv_depth   (None, 64, 64, 64)           16448
['conv2_block2_out_depth[0][0]
 (Conv2D)                                                                   ']
 conv2_block3_1_bnrgb (Batc  (None, 64, 64, 64)           256
['conv2_block3_1_convrgb[0][0]
 hNormalization)                                                    ']



 conv2_block3_1_bn_depth (B  (None, 64, 64, 64)           256
['conv2_block3_1_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block3_1_relurgb (Ac  (None, 64, 64, 64)           0
['conv2_block3_1_bnrgb[0][0]']
 tivation)



 conv2_block3_1_relu_depth   (None, 64, 64, 64)           0
['conv2_block3_1_bn_depth[0][0

(Activation)                                                       ]']



 conv2_block3_2_convrgb (Co  (None, 64, 64, 64)           36928
['conv2_block3_1_relurgb[0][0]
 nv2D)                                                              ']



 conv2_block3_2_conv_depth   (None, 64, 64, 64)           36928
['conv2_block3_1_relu_depth[0]
 (Conv2D)
[0]']


 conv2_block3_2_bnrgb (Batc  (None, 64, 64, 64)           256
['conv2_block3_2_convrgb[0][0]
 hNormalization)                                                    ']



 conv2_block3_2_bn_depth (B  (None, 64, 64, 64)           256
['conv2_block3_2_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block3_2_relurgb (Ac  (None, 64, 64, 64)           0
['conv2_block3_2_bnrgb[0][0]']
```

```
 tivation)



 conv2_block3_2_relu_depth   (None, 64, 64, 64)          0
['conv2_block3_2_bn_depth[0][0

(Activation)                                                                   ]']



 conv2_block3_3_convrgb (Co  (None, 64, 64, 256)         16640
['conv2_block3_2_relurgb[0][0]
 nv2D)                                                                          ']



 conv2_block3_3_conv_depth   (None, 64, 64, 256)         16640
['conv2_block3_2_relu_depth[0]
 (Conv2D)
[0]']


 conv2_block3_3_bnrgb (Batc  (None, 64, 64, 256)         1024
['conv2_block3_3_convrgb[0][0]
 hNormalization)                                                                ']



 conv2_block3_3_bn_depth (B  (None, 64, 64, 256)         1024
['conv2_block3_3_conv_depth[0]
 atchNormalization)
[0]']


 conv2_block3_addrgb (Add)   (None, 64, 64, 256)         0
['conv2_block2_outrgb[0][0]',

'conv2_block3_3_bnrgb[0][0]']


 conv2_block3_add_depth (Ad  (None, 64, 64, 256)         0
['conv2_block2_out_depth[0][0]
 d)                                                                             ',


'conv2_block3_3_bn_depth[0][0
                                                                                ]'
]
```

```
 conv2_block3_outrgb (Activ  (None, 64, 64, 256)          0
['conv2_block3_addrgb[0][0]']
 ation)



 conv2_block3_out_depth (Ac  (None, 64, 64, 256)          0
['conv2_block3_add_depth[0][0]
 tivation)                                                          ']



 conv3_block1_1_convrgb (Co  (None, 32, 32, 128)          32896
['conv2_block3_outrgb[0][0]']
 nv2D)



 conv3_block1_1_conv_depth   (None, 32, 32, 128)          32896
['conv2_block3_out_depth[0][0]
 (Conv2D)                                                           ']



 conv3_block1_1_bnrgb (Batc  (None, 32, 32, 128)          512
['conv3_block1_1_convrgb[0][0]
 hNormalization)                                                    ']



 conv3_block1_1_bn_depth (B  (None, 32, 32, 128)          512
['conv3_block1_1_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block1_1_relurgb (Ac  (None, 32, 32, 128)          0
['conv3_block1_1_bnrgb[0][0]']
 tivation)



 conv3_block1_1_relu_depth   (None, 32, 32, 128)          0
['conv3_block1_1_bn_depth[0][0

(Activation)                                                       ]']



 conv3_block1_2_convrgb (Co  (None, 32, 32, 128)          147584
['conv3_block1_1_relurgb[0][0]
```

```
 nv2D)                                                                   ']



 conv3_block1_2_conv_depth   (None, 32, 32, 128)          147584
['conv3_block1_1_relu_depth[0]
 (Conv2D)
[0]']


 conv3_block1_2_bnrgb (Batc  (None, 32, 32, 128)          512
['conv3_block1_2_convrgb[0][0]
 hNormalization)                                                         ']



 conv3_block1_2_bn_depth (B  (None, 32, 32, 128)          512
['conv3_block1_2_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block1_2_relurgb (Ac  (None, 32, 32, 128)          0
['conv3_block1_2_bnrgb[0][0]']
 tivation)



 conv3_block1_2_relu_depth   (None, 32, 32, 128)          0
['conv3_block1_2_bn_depth[0][0

(Activation)                                                            ]']



 conv3_block1_0_convrgb (Co  (None, 32, 32, 512)          131584
['conv2_block3_outrgb[0][0]']
 nv2D)



 conv3_block1_3_convrgb (Co  (None, 32, 32, 512)          66048
['conv3_block1_2_relurgb[0][0]
 nv2D)                                                                   ']



 conv3_block1_0_conv_depth   (None, 32, 32, 512)          131584
['conv2_block3_out_depth[0][0]
 (Conv2D)                                                                ']
 conv3_block1_3_conv_depth   (None, 32, 32, 512)          66048
['conv3_block1_2_relu_depth[0]
 (Conv2D)
[0]']


 conv3_block1_0_bnrgb (Batc  (None, 32, 32, 512)          2048
['conv3_block1_0_convrgb[0][0]
 hNormalization)                                                   ']



 conv3_block1_3_bnrgb (Batc  (None, 32, 32, 512)          2048
['conv3_block1_3_convrgb[0][0]
 hNormalization)                                                   ']



 conv3_block1_0_bn_depth (B  (None, 32, 32, 512)          2048
['conv3_block1_0_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block1_3_bn_depth (B  (None, 32, 32, 512)          2048
['conv3_block1_3_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block1_addrgb (Add)   (None, 32, 32, 512)          0
['conv3_block1_0_bnrgb[0][0]',

'conv3_block1_3_bnrgb[0][0]']


 conv3_block1_add_depth (Ad  (None, 32, 32, 512)          0
['conv3_block1_0_bn_depth[0][0

d)                                                                ]',


'conv3_block1_3_bn_depth[0][0
                                                                   ]'
]


 conv3_block1_outrgb (Activ  (None, 32, 32, 512)          0
['conv3_block1_addrgb[0][0]']
```

```
 ation)



 conv3_block1_out_depth (Ac  (None, 32, 32, 512)          0
['conv3_block1_add_depth[0][0]
 tivation)                                                          ']



 conv3_block2_1_convrgb (Co  (None, 32, 32, 128)          65664
['conv3_block1_outrgb[0][0]']
 nv2D)



 conv3_block2_1_conv_depth   (None, 32, 32, 128)          65664
['conv3_block1_out_depth[0][0]
 (Conv2D)                                                           ']



 conv3_block2_1_bnrgb (Batc  (None, 32, 32, 128)          512
['conv3_block2_1_convrgb[0][0]
 hNormalization)                                                    ']



 conv3_block2_1_bn_depth (B  (None, 32, 32, 128)          512
['conv3_block2_1_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block2_1_relurgb (Ac  (None, 32, 32, 128)          0
['conv3_block2_1_bnrgb[0][0]']
 tivation)



 conv3_block2_1_relu_depth   (None, 32, 32, 128)          0
['conv3_block2_1_bn_depth[0][0

(Activation)                                                       ]']



 conv3_block2_2_convrgb (Co  (None, 32, 32, 128)          147584
['conv3_block2_1_relurgb[0][0]
 nv2D)                                                              ']
 conv3_block2_2_conv_depth   (None, 32, 32, 128)          147584
['conv3_block2_1_relu_depth[0]
 (Conv2D)
[0]']


 conv3_block2_2_bnrgb (Batc  (None, 32, 32, 128)          512
['conv3_block2_2_convrgb[0][0]
 hNormalization)                                                   ']



 conv3_block2_2_bn_depth (B  (None, 32, 32, 128)          512
['conv3_block2_2_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block2_2_relurgb (Ac  (None, 32, 32, 128)          0
['conv3_block2_2_bnrgb[0][0]']
 tivation)



 conv3_block2_2_relu_depth   (None, 32, 32, 128)          0
['conv3_block2_2_bn_depth[0][0

(Activation)                                                      ]']



 conv3_block2_3_convrgb (Co  (None, 32, 32, 512)          66048
['conv3_block2_2_relurgb[0][0]
 nv2D)                                                             ']



 conv3_block2_3_conv_depth   (None, 32, 32, 512)          66048
['conv3_block2_2_relu_depth[0]
 (Conv2D)
[0]']


 conv3_block2_3_bnrgb (Batc  (None, 32, 32, 512)          2048
['conv3_block2_3_convrgb[0][0]
 hNormalization)                                                   ']
 conv3_block2_3_bn_depth (B  (None, 32, 32, 512)          2048
['conv3_block2_3_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block2_addrgb (Add)   (None, 32, 32, 512)          0
['conv3_block1_outrgb[0][0]',

'conv3_block2_3_bnrgb[0][0]']


 conv3_block2_add_depth (Ad  (None, 32, 32, 512)          0
['conv3_block1_out_depth[0][0]
 d)                                                                  ',


'conv3_block2_3_bn_depth[0][0
                                                                     ]'
]


 conv3_block2_outrgb (Activ  (None, 32, 32, 512)          0
['conv3_block2_addrgb[0][0]']
 ation)



 conv3_block2_out_depth (Ac  (None, 32, 32, 512)          0
['conv3_block2_add_depth[0][0]
 tivation)                                                           ']



 conv3_block3_1_convrgb (Co  (None, 32, 32, 128)          65664
['conv3_block2_outrgb[0][0]']
 nv2D)



 conv3_block3_1_conv_depth   (None, 32, 32, 128)          65664
['conv3_block2_out_depth[0][0]
 (Conv2D)                                                            ']



 conv3_block3_1_bnrgb (Batc  (None, 32, 32, 128)          512
['conv3_block3_1_convrgb[0][0]
 hNormalization)                                                     ']
 conv3_block3_1_bn_depth (B  (None, 32, 32, 128)          512
['conv3_block3_1_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block3_1_relurgb (Ac  (None, 32, 32, 128)          0
['conv3_block3_1_bnrgb[0][0]']
 tivation)



 conv3_block3_1_relu_depth   (None, 32, 32, 128)          0
['conv3_block3_1_bn_depth[0][0

(Activation)                                                        ]']



 conv3_block3_2_convrgb (Co  (None, 32, 32, 128)          147584
['conv3_block3_1_relurgb[0][0]
 nv2D)                                                               ']



 conv3_block3_2_conv_depth   (None, 32, 32, 128)          147584
['conv3_block3_1_relu_depth[0]
 (Conv2D)
[0]']


 conv3_block3_2_bnrgb (Batc  (None, 32, 32, 128)          512
['conv3_block3_2_convrgb[0][0]
 hNormalization)                                                     ']



 conv3_block3_2_bn_depth (B  (None, 32, 32, 128)          512
['conv3_block3_2_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block3_2_relurgb (Ac  (None, 32, 32, 128)          0
['conv3_block3_2_bnrgb[0][0]']
 tivation)



 conv3_block3_2_relu_depth   (None, 32, 32, 128)          0
```

```
['conv3_block3_2_bn_depth[0][0

(Activation)                                                         ]']



 conv3_block3_3_convrgb (Co  (None, 32, 32, 512)          66048
['conv3_block3_2_relurgb[0][0]
 nv2D)                                                                ']



 conv3_block3_3_conv_depth   (None, 32, 32, 512)          66048
['conv3_block3_2_relu_depth[0]
 (Conv2D)
[0]']


 conv3_block3_3_bnrgb (Batc  (None, 32, 32, 512)          2048
['conv3_block3_3_convrgb[0][0]
 hNormalization)                                                      ']



 conv3_block3_3_bn_depth (B  (None, 32, 32, 512)          2048
['conv3_block3_3_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block3_addrgb (Add)   (None, 32, 32, 512)          0
['conv3_block2_outrgb[0][0]',

'conv3_block3_3_bnrgb[0][0]']


 conv3_block3_add_depth (Ad  (None, 32, 32, 512)          0
['conv3_block2_out_depth[0][0]
 d)                                                                   ',


'conv3_block3_3_bn_depth[0][0
                                                                     ]'
]


 conv3_block3_outrgb (Activ  (None, 32, 32, 512)          0
['conv3_block3_addrgb[0][0]']
 ation)
 conv3_block3_out_depth (Ac  (None, 32, 32, 512)          0
['conv3_block3_add_depth[0][0]
 tivation)                                                                          ']



 conv3_block4_1_convrgb (Co  (None, 32, 32, 128)          65664
['conv3_block3_outrgb[0][0]']
 nv2D)



 conv3_block4_1_conv_depth   (None, 32, 32, 128)          65664
['conv3_block3_out_depth[0][0]
 (Conv2D)                                                                           ']



 conv3_block4_1_bnrgb (Batc  (None, 32, 32, 128)          512
['conv3_block4_1_convrgb[0][0]
 hNormalization)                                                                    ']



 conv3_block4_1_bn_depth (B  (None, 32, 32, 128)          512
['conv3_block4_1_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block4_1_relurgb (Ac  (None, 32, 32, 128)          0
['conv3_block4_1_bnrgb[0][0]']
 tivation)



 conv3_block4_1_relu_depth   (None, 32, 32, 128)          0
['conv3_block4_1_bn_depth[0][0

(Activation)                                                                       ]']



 conv3_block4_2_convrgb (Co  (None, 32, 32, 128)          147584
['conv3_block4_1_relurgb[0][0]
 nv2D)                                                                              ']



 conv3_block4_2_conv_depth   (None, 32, 32, 128)          147584
```

```
['conv3_block4_1_relu_depth[0]
 (Conv2D)
[0]']


 conv3_block4_2_bnrgb (Batc  (None, 32, 32, 128)          512
['conv3_block4_2_convrgb[0][0]
 hNormalization)                                                                 ']



 conv3_block4_2_bn_depth (B  (None, 32, 32, 128)          512
['conv3_block4_2_conv_depth[0]
 atchNormalization)
[0]']


 conv3_block4_2_relurgb (Ac  (None, 32, 32, 128)          0
['conv3_block4_2_bnrgb[0][0]']
 tivation)



 conv3_block4_2_relu_depth   (None, 32, 32, 128)          0
['conv3_block4_2_bn_depth[0][0

(Activation)                                                                    ]']



 conv3_block4_3_convrgb (Co  (None, 32, 32, 512)          66048
['conv3_block4_2_relurgb[0][0]
 nv2D)                                                                           ']



 conv3_block4_3_conv_depth   (None, 32, 32, 512)          66048
['conv3_block4_2_relu_depth[0]
 (Conv2D)
[0]']


 conv3_block4_3_bnrgb (Batc  (None, 32, 32, 512)          2048
['conv3_block4_3_convrgb[0][0]
 hNormalization)                                                                 ']



 conv3_block4_3_bn_depth (B  (None, 32, 32, 512)          2048
['conv3_block4_3_conv_depth[0]
 atchNormalization)
```

```
[0]']


 conv3_block4_addrgb (Add)   (None, 32, 32, 512)          0
['conv3_block3_outrgb[0][0]',

'conv3_block4_3_bnrgb[0][0]']


 conv3_block4_add_depth (Ad  (None, 32, 32, 512)          0
['conv3_block3_out_depth[0][0]
 d)                                                                             ',


'conv3_block4_3_bn_depth[0][0
                                                                                ]'
]


 conv3_block4_outrgb (Activ  (None, 32, 32, 512)          0
['conv3_block4_addrgb[0][0]']
 ation)



 conv3_block4_out_depth (Ac  (None, 32, 32, 512)          0
['conv3_block4_add_depth[0][0]
 tivation)                                                                      ']



 conv4_block1_1_convrgb (Co  (None, 16, 16, 256)          131328
['conv3_block4_outrgb[0][0]']
 nv2D)



 conv4_block1_1_conv_depth   (None, 16, 16, 256)          131328
['conv3_block4_out_depth[0][0]
 (Conv2D)                                                                       ']



 conv4_block1_1_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block1_1_convrgb[0][0]
 hNormalization)                                                                ']



 conv4_block1_1_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block1_1_conv_depth[0]
```

```
 atchNormalization)
[0]']


 conv4_block1_1_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block1_1_bnrgb[0][0]']
 tivation)



 conv4_block1_1_relu_depth   (None, 16, 16, 256)          0
['conv4_block1_1_bn_depth[0][0

(Activation)                                                                 ]']



 conv4_block1_2_convrgb (Co  (None, 16, 16, 256)          590080
['conv4_block1_1_relurgb[0][0]
 nv2D)                                                                        ']



 conv4_block1_2_conv_depth   (None, 16, 16, 256)          590080
['conv4_block1_1_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block1_2_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block1_2_convrgb[0][0]
 hNormalization)                                                              ']



 conv4_block1_2_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block1_2_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block1_2_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block1_2_bnrgb[0][0]']
 tivation)



 conv4_block1_2_relu_depth   (None, 16, 16, 256)          0
['conv4_block1_2_bn_depth[0][0

(Activation)                                                                 ]']
```

```
 conv4_block1_0_convrgb (Co  (None, 16, 16, 1024)         525312
['conv3_block4_outrgb[0][0]']
 nv2D)



 conv4_block1_3_convrgb (Co  (None, 16, 16, 1024)         263168
['conv4_block1_2_relurgb[0][0]
 nv2D)                                                              ']



 conv4_block1_0_conv_depth   (None, 16, 16, 1024)         525312
['conv3_block4_out_depth[0][0]
 (Conv2D)                                                           ']



 conv4_block1_3_conv_depth   (None, 16, 16, 1024)         263168
['conv4_block1_2_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block1_0_bnrgb (Batc  (None, 16, 16, 1024)         4096
['conv4_block1_0_convrgb[0][0]
 hNormalization)                                                    ']



 conv4_block1_3_bnrgb (Batc  (None, 16, 16, 1024)         4096
['conv4_block1_3_convrgb[0][0]
 hNormalization)                                                    ']



 conv4_block1_0_bn_depth (B  (None, 16, 16, 1024)         4096
['conv4_block1_0_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block1_3_bn_depth (B  (None, 16, 16, 1024)         4096
['conv4_block1_3_conv_depth[0]
 atchNormalization)
[0]']
```

```
 conv4_block1_addrgb (Add)   (None, 16, 16, 1024)         0
['conv4_block1_0_bnrgb[0][0]',

'conv4_block1_3_bnrgb[0][0]']


 conv4_block1_add_depth (Ad  (None, 16, 16, 1024)         0
['conv4_block1_0_bn_depth[0][0

d)                                                                   ]',


'conv4_block1_3_bn_depth[0][0
                                                                      ]'
]


 conv4_block1_outrgb (Activ  (None, 16, 16, 1024)         0
['conv4_block1_addrgb[0][0]']
 ation)



 conv4_block1_out_depth (Ac  (None, 16, 16, 1024)         0
['conv4_block1_add_depth[0][0]
 tivation)                                                            ']



 conv4_block2_1_convrgb (Co  (None, 16, 16, 256)          262400
['conv4_block1_outrgb[0][0]']
 nv2D)



 conv4_block2_1_conv_depth   (None, 16, 16, 256)          262400
['conv4_block1_out_depth[0][0]
 (Conv2D)                                                             ']



 conv4_block2_1_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block2_1_convrgb[0][0]
 hNormalization)                                                      ']



 conv4_block2_1_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block2_1_conv_depth[0]
 atchNormalization)
[0]']
```

```
 conv4_block2_1_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block2_1_bnrgb[0][0]']
 tivation)



 conv4_block2_1_relu_depth   (None, 16, 16, 256)          0
['conv4_block2_1_bn_depth[0][0

(Activation)                                                                   ]']



 conv4_block2_2_convrgb (Co  (None, 16, 16, 256)          590080
['conv4_block2_1_relurgb[0][0]
 nv2D)                                                                          ']



 conv4_block2_2_conv_depth   (None, 16, 16, 256)          590080
['conv4_block2_1_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block2_2_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block2_2_convrgb[0][0]
 hNormalization)                                                                ']



 conv4_block2_2_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block2_2_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block2_2_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block2_2_bnrgb[0][0]']
 tivation)



 conv4_block2_2_relu_depth   (None, 16, 16, 256)          0
['conv4_block2_2_bn_depth[0][0

(Activation)                                                                   ]']
```

```
 conv4_block2_3_convrgb (Co  (None, 16, 16, 1024)         263168
['conv4_block2_2_relurgb[0][0]
 nv2D)                                                                   ']



 conv4_block2_3_conv_depth   (None, 16, 16, 1024)         263168
['conv4_block2_2_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block2_3_bnrgb (Batc  (None, 16, 16, 1024)         4096
['conv4_block2_3_convrgb[0][0]
 hNormalization)                                                         ']



 conv4_block2_3_bn_depth (B  (None, 16, 16, 1024)         4096
['conv4_block2_3_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block2_addrgb (Add)   (None, 16, 16, 1024)         0
['conv4_block1_outrgb[0][0]',

'conv4_block2_3_bnrgb[0][0]']


 conv4_block2_add_depth (Ad  (None, 16, 16, 1024)         0
['conv4_block1_out_depth[0][0
 d)                                                                      ',


'conv4_block2_3_bn_depth[0][0
                                                                         ]'
]


 conv4_block2_outrgb (Activ  (None, 16, 16, 1024)         0
['conv4_block2_addrgb[0][0]']
 ation)



 conv4_block2_out_depth (Ac  (None, 16, 16, 1024)         0
['conv4_block2_add_depth[0][0
 tivation)                                                               ']
 conv4_block3_1_convrgb (Co  (None, 16, 16, 256)          262400
['conv4_block2_outrgb[0][0]']
 nv2D)



 conv4_block3_1_conv_depth   (None, 16, 16, 256)          262400
['conv4_block2_out_depth[0][0]
 (Conv2D)                                                            ']



 conv4_block3_1_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block3_1_convrgb[0][0]
 hNormalization)                                                     ']



 conv4_block3_1_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block3_1_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block3_1_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block3_1_bnrgb[0][0]']
 tivation)



 conv4_block3_1_relu_depth   (None, 16, 16, 256)          0
['conv4_block3_1_bn_depth[0][0

(Activation)                                                        ]']



 conv4_block3_2_convrgb (Co  (None, 16, 16, 256)          590080
['conv4_block3_1_relurgb[0][0]
 nv2D)                                                               ']



 conv4_block3_2_conv_depth   (None, 16, 16, 256)          590080
['conv4_block3_1_relu_depth[0]
 (Conv2D)
[0]']
```

```
 conv4_block3_2_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block3_2_convrgb[0][0]
 hNormalization)                                                    ']



 conv4_block3_2_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block3_2_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block3_2_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block3_2_bnrgb[0][0]']
 tivation)



 conv4_block3_2_relu_depth   (None, 16, 16, 256)          0
['conv4_block3_2_bn_depth[0][0

(Activation)                                                       ]']



 conv4_block3_3_convrgb (Co  (None, 16, 16, 1024)         263168
['conv4_block3_2_relurgb[0][0]
 nv2D)                                                              ']



 conv4_block3_3_conv_depth   (None, 16, 16, 1024)         263168
['conv4_block3_2_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block3_3_bnrgb (Batc  (None, 16, 16, 1024)         4096
['conv4_block3_3_convrgb[0][0]
 hNormalization)                                                    ']



 conv4_block3_3_bn_depth (B  (None, 16, 16, 1024)         4096
['conv4_block3_3_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block3_addrgb (Add)   (None, 16, 16, 1024)         0
['conv4_block2_outrgb[0][0]',
'conv4_block3_3_bnrgb[0][0]']


 conv4_block3_add_depth (Ad  (None, 16, 16, 1024)         0
['conv4_block2_out_depth[0][0]
 d)                                                                      ',


'conv4_block3_3_bn_depth[0][0
                                                                         ]'
]


 conv4_block3_outrgb (Activ  (None, 16, 16, 1024)         0
['conv4_block3_addrgb[0][0]']
 ation)



 conv4_block3_out_depth (Ac  (None, 16, 16, 1024)         0
['conv4_block3_add_depth[0][0]
 tivation)                                                               ']



 conv4_block4_1_convrgb (Co  (None, 16, 16, 256)          262400
['conv4_block3_outrgb[0][0]']
 nv2D)



 conv4_block4_1_conv_depth   (None, 16, 16, 256)          262400
['conv4_block3_out_depth[0][0]
 (Conv2D)                                                                ']



 conv4_block4_1_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block4_1_convrgb[0][0]
 hNormalization)                                                         ']



 conv4_block4_1_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block4_1_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block4_1_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block4_1_bnrgb[0][0]']
 tivation)



 conv4_block4_1_relu_depth   (None, 16, 16, 256)          0
['conv4_block4_1_bn_depth[0][0

(Activation)                                                                  ]']



 conv4_block4_2_convrgb (Co  (None, 16, 16, 256)          590080
['conv4_block4_1_relurgb[0][0]
 nv2D)                                                                         ']



 conv4_block4_2_conv_depth   (None, 16, 16, 256)          590080
['conv4_block4_1_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block4_2_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block4_2_convrgb[0][0]
 hNormalization)                                                               ']



 conv4_block4_2_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block4_2_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block4_2_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block4_2_bnrgb[0][0]']
 tivation)



 conv4_block4_2_relu_depth   (None, 16, 16, 256)          0
['conv4_block4_2_bn_depth[0][0

(Activation)                                                                  ]']



 conv4_block4_3_convrgb (Co  (None, 16, 16, 1024)         263168
['conv4_block4_2_relurgb[0][0]
```

```
 nv2D)                                                                   ']



 conv4_block4_3_conv_depth   (None, 16, 16, 1024)         263168
['conv4_block4_2_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block4_3_bnrgb (Batc  (None, 16, 16, 1024)         4096
['conv4_block4_3_convrgb[0][0]
 hNormalization)                                                         ']



 conv4_block4_3_bn_depth (B  (None, 16, 16, 1024)         4096
['conv4_block4_3_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block4_addrgb (Add)   (None, 16, 16, 1024)         0
['conv4_block3_outrgb[0][0]',

'conv4_block4_3_bnrgb[0][0]']


 conv4_block4_add_depth (Ad  (None, 16, 16, 1024)         0
['conv4_block3_out_depth[0][0]
 d)                                                                      ',


'conv4_block4_3_bn_depth[0][0
                                                                         ]'
]


 conv4_block4_outrgb (Activ  (None, 16, 16, 1024)         0
['conv4_block4_addrgb[0][0]']
 ation)



 conv4_block4_out_depth (Ac  (None, 16, 16, 1024)         0
['conv4_block4_add_depth[0][0]
 tivation)                                                               ']



 conv4_block5_1_convrgb (Co  (None, 16, 16, 256)          262400
['conv4_block4_outrgb[0][0]']
 nv2D)



 conv4_block5_1_conv_depth   (None, 16, 16, 256)         262400
['conv4_block4_out_depth[0][0]
 (Conv2D)                                                          ']



 conv4_block5_1_bnrgb (Batc  (None, 16, 16, 256)         1024
['conv4_block5_1_convrgb[0][0]
 hNormalization)                                                   ']



 conv4_block5_1_bn_depth (B  (None, 16, 16, 256)         1024
['conv4_block5_1_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block5_1_relurgb (Ac  (None, 16, 16, 256)         0
['conv4_block5_1_bnrgb[0][0]']
 tivation)



 conv4_block5_1_relu_depth   (None, 16, 16, 256)         0
['conv4_block5_1_bn_depth[0][0

(Activation)                                                      ]']



 conv4_block5_2_convrgb (Co  (None, 16, 16, 256)         590080
['conv4_block5_1_relurgb[0][0]
 nv2D)                                                             ']



 conv4_block5_2_conv_depth   (None, 16, 16, 256)         590080
['conv4_block5_1_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block5_2_bnrgb (Batc  (None, 16, 16, 256)         1024
['conv4_block5_2_convrgb[0][0]
 hNormalization)                                                   ']
 conv4_block5_2_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block5_2_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block5_2_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block5_2_bnrgb[0][0]']
 tivation)



 conv4_block5_2_relu_depth   (None, 16, 16, 256)          0
['conv4_block5_2_bn_depth[0][0

(Activation)                                                       ]']



 conv4_block5_3_convrgb (Co  (None, 16, 16, 1024)         263168
['conv4_block5_2_relurgb[0][0]
 nv2D)                                                              ']



 conv4_block5_3_conv_depth   (None, 16, 16, 1024)         263168
['conv4_block5_2_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block5_3_bnrgb (Batc  (None, 16, 16, 1024)         4096
['conv4_block5_3_convrgb[0][0]
 hNormalization)                                                    ']



 conv4_block5_3_bn_depth (B  (None, 16, 16, 1024)         4096
['conv4_block5_3_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block5_addrgb (Add)   (None, 16, 16, 1024)         0
['conv4_block4_outrgb[0][0]',

'conv4_block5_3_bnrgb[0][0]']
```

```
 conv4_block5_add_depth (Ad  (None, 16, 16, 1024)         0
['conv4_block4_out_depth[0][0]
 d)                                                                  ',


'conv4_block5_3_bn_depth[0][0
                                                                     ]'
]


 conv4_block5_outrgb (Activ  (None, 16, 16, 1024)         0
['conv4_block5_addrgb[0][0]']
 ation)



 conv4_block5_out_depth (Ac  (None, 16, 16, 1024)         0
['conv4_block5_add_depth[0][0]
 tivation)                                                           ']



 conv4_block6_1_convrgb (Co  (None, 16, 16, 256)          262400
['conv4_block5_outrgb[0][0]']
 nv2D)



 conv4_block6_1_conv_depth   (None, 16, 16, 256)          262400
['conv4_block5_out_depth[0][0]
 (Conv2D)                                                            ']



 conv4_block6_1_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block6_1_convrgb[0][0]
 hNormalization)                                                     ']



 conv4_block6_1_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block6_1_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block6_1_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block6_1_bnrgb[0][0]']
 tivation)
 conv4_block6_1_relu_depth   (None, 16, 16, 256)          0
['conv4_block6_1_bn_depth[0][0

(Activation)                                                                    ]']



 conv4_block6_2_convrgb (Co  (None, 16, 16, 256)          590080
['conv4_block6_1_relurgb[0][0]
 nv2D)                                                                           ']



 conv4_block6_2_conv_depth   (None, 16, 16, 256)          590080
['conv4_block6_1_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block6_2_bnrgb (Batc  (None, 16, 16, 256)          1024
['conv4_block6_2_convrgb[0][0]
 hNormalization)                                                                 ']



 conv4_block6_2_bn_depth (B  (None, 16, 16, 256)          1024
['conv4_block6_2_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block6_2_relurgb (Ac  (None, 16, 16, 256)          0
['conv4_block6_2_bnrgb[0][0]']
 tivation)



 conv4_block6_2_relu_depth   (None, 16, 16, 256)          0
['conv4_block6_2_bn_depth[0][0

(Activation)                                                                    ]']



 conv4_block6_3_convrgb (Co  (None, 16, 16, 1024)         263168
['conv4_block6_2_relurgb[0][0]
 nv2D)                                                                           ']
 conv4_block6_3_conv_depth   (None, 16, 16, 1024)         263168
['conv4_block6_2_relu_depth[0]
 (Conv2D)
[0]']


 conv4_block6_3_bnrgb (Batc  (None, 16, 16, 1024)         4096
['conv4_block6_3_convrgb[0][0]
 hNormalization)                                                    ']



 conv4_block6_3_bn_depth (B  (None, 16, 16, 1024)         4096
['conv4_block6_3_conv_depth[0]
 atchNormalization)
[0]']


 conv4_block6_addrgb (Add)   (None, 16, 16, 1024)         0
['conv4_block5_outrgb[0][0]',

'conv4_block6_3_bnrgb[0][0]']


 conv4_block6_add_depth (Ad  (None, 16, 16, 1024)         0
['conv4_block5_out_depth[0][0]
 d)                                                                 ',


'conv4_block6_3_bn_depth[0][0
                                                                    ]'
]


 conv4_block6_outrgb (Activ  (None, 16, 16, 1024)         0
['conv4_block6_addrgb[0][0]']
 ation)



 conv4_block6_out_depth (Ac  (None, 16, 16, 1024)         0
['conv4_block6_add_depth[0][0]
 tivation)                                                          ']



 conv5_block1_1_convrgb (Co  (None, 8, 8, 512)            524800
['conv4_block6_outrgb[0][0]']
 nv2D)
 conv5_block1_1_conv_depth   (None, 8, 8, 512)          524800
['conv4_block6_out_depth[0][0]
 (Conv2D)                                                        ']


 conv5_block1_1_bnrgb (Batc  (None, 8, 8, 512)          2048
['conv5_block1_1_convrgb[0][0]
 hNormalization)                                                 ']


 conv5_block1_1_bn_depth (B  (None, 8, 8, 512)          2048
['conv5_block1_1_conv_depth[0]
 atchNormalization)
[0]']


 conv5_block1_1_relurgb (Ac  (None, 8, 8, 512)          0
['conv5_block1_1_bnrgb[0][0]']
 tivation)


 conv5_block1_1_relu_depth   (None, 8, 8, 512)          0
['conv5_block1_1_bn_depth[0][0

(Activation)                                                    ]']


 conv5_block1_2_convrgb (Co  (None, 8, 8, 512)          2359808
['conv5_block1_1_relurgb[0][0]
 nv2D)                                                           ']


 conv5_block1_2_conv_depth   (None, 8, 8, 512)          2359808
['conv5_block1_1_relu_depth[0]
 (Conv2D)
[0]']


 conv5_block1_2_bnrgb (Batc  (None, 8, 8, 512)          2048
['conv5_block1_2_convrgb[0][0]
 hNormalization)                                                 ']
 conv5_block1_2_bn_depth (B  (None, 8, 8, 512)            2048
['conv5_block1_2_conv_depth[0]
 atchNormalization)
[0]']


 conv5_block1_2_relurgb (Ac  (None, 8, 8, 512)            0
['conv5_block1_2_bnrgb[0][0]']
 tivation)



 conv5_block1_2_relu_depth   (None, 8, 8, 512)            0
['conv5_block1_2_bn_depth[0][0

(Activation)                                                       ]']



 conv5_block1_0_convrgb (Co  (None, 8, 8, 2048)           2099200
['conv4_block6_outrgb[0][0]']
 nv2D)



 conv5_block1_3_convrgb (Co  (None, 8, 8, 2048)           1050624
['conv5_block1_2_relurgb[0][0]
 nv2D)                                                              ']



 conv5_block1_0_conv_depth   (None, 8, 8, 2048)           2099200
['conv4_block6_out_depth[0][0]
 (Conv2D)                                                           ']



 conv5_block1_3_conv_depth   (None, 8, 8, 2048)           1050624
['conv5_block1_2_relu_depth[0]
 (Conv2D)
[0]']


 conv5_block1_0_bnrgb (Batc  (None, 8, 8, 2048)           8192
['conv5_block1_0_convrgb[0][0]
 hNormalization)                                                    ']



 conv5_block1_3_bnrgb (Batc  (None, 8, 8, 2048)           8192
['conv5_block1_3_convrgb[0][0]
```

```
 hNormalization)                                                      ']



 conv5_block1_0_bn_depth (B  (None, 8, 8, 2048)          8192
['conv5_block1_0_conv_depth[0]
 atchNormalization)
[0]']


 conv5_block1_3_bn_depth (B  (None, 8, 8, 2048)          8192
['conv5_block1_3_conv_depth[0]
 atchNormalization)
[0]']


 conv5_block1_addrgb (Add)   (None, 8, 8, 2048)          0
['conv5_block1_0_bnrgb[0][0]',

'conv5_block1_3_bnrgb[0][0]']


 conv5_block1_add_depth (Ad  (None, 8, 8, 2048)          0
['conv5_block1_0_bn_depth[0][0

d)                                                                   ]',


'conv5_block1_3_bn_depth[0][0
                                                                      ]'
]


 conv5_block1_outrgb (Activ  (None, 8, 8, 2048)          0
['conv5_block1_addrgb[0][0]']
 ation)



 conv5_block1_out_depth (Ac  (None, 8, 8, 2048)          0
['conv5_block1_add_depth[0][0]
 tivation)                                                            ']



 conv5_block2_1_convrgb (Co  (None, 8, 8, 512)           1049088
['conv5_block1_outrgb[0][0]']
 nv2D)
 conv5_block2_1_conv_depth   (None, 8, 8, 512)            1049088
['conv5_block1_out_depth[0][0'
 (Conv2D)                                                           ']

 conv5_block2_1_bnrgb (Batc  (None, 8, 8, 512)            2048
['conv5_block2_1_convrgb[0][0'
 hNormalization)                                                    ']

 conv5_block2_1_bn_depth (B  (None, 8, 8, 512)            2048
['conv5_block2_1_conv_depth[0]
 atchNormalization)
[0]']

 conv5_block2_1_relurgb (Ac  (None, 8, 8, 512)            0
['conv5_block2_1_bnrgb[0][0]']
 tivation)

 conv5_block2_1_relu_depth   (None, 8, 8, 512)            0
['conv5_block2_1_bn_depth[0][0

(Activation)                                                       ]']

 conv5_block2_2_convrgb (Co  (None, 8, 8, 512)            2359808
['conv5_block2_1_relurgb[0][0'
 nv2D)                                                              ']

 conv5_block2_2_conv_depth   (None, 8, 8, 512)            2359808
['conv5_block2_1_relu_depth[0]
 (Conv2D)
[0]']

 conv5_block2_2_bnrgb (Batc  (None, 8, 8, 512)            2048
['conv5_block2_2_convrgb[0][0'
 hNormalization)                                                    ']

 conv5_block2_2_bn_depth (B  (None, 8, 8, 512)            2048
['conv5_block2_2_conv_depth[0]
```

```
 atchNormalization)
[0]']


 conv5_block2_2_relurgb (Ac  (None, 8, 8, 512)            0
['conv5_block2_2_bnrgb[0][0]']
 tivation)



 conv5_block2_2_relu_depth   (None, 8, 8, 512)            0
['conv5_block2_2_bn_depth[0][0

(Activation)                                                          ]']



 conv5_block2_3_convrgb (Co  (None, 8, 8, 2048)           1050624
['conv5_block2_2_relurgb[0][0
 nv2D)                                                                 ']



 conv5_block2_3_conv_depth   (None, 8, 8, 2048)           1050624
['conv5_block2_2_relu_depth[0]
 (Conv2D)
[0]']


 conv5_block2_3_bnrgb (Batc  (None, 8, 8, 2048)           8192
['conv5_block2_3_convrgb[0][0
 hNormalization)                                                       ']



 conv5_block2_3_bn_depth (B  (None, 8, 8, 2048)           8192
['conv5_block2_3_conv_depth[0]
 atchNormalization)
[0]']


 conv5_block2_addrgb (Add)   (None, 8, 8, 2048)           0
['conv5_block1_outrgb[0][0]',

'conv5_block2_3_bnrgb[0][0]']


 conv5_block2_add_depth (Ad  (None, 8, 8, 2048)           0
['conv5_block1_out_depth[0][0
 d)                                                                    ',
'conv5_block2_3_bn_depth[0][0
                                                                     ]'
]


 conv5_block2_outrgb (Activ  (None, 8, 8, 2048)          0
['conv5_block2_addrgb[0][0]']
 ation)



 conv5_block2_out_depth (Ac  (None, 8, 8, 2048)          0
['conv5_block2_add_depth[0][0]
 tivation)                                                           ']



 conv5_block3_1_convrgb (Co  (None, 8, 8, 512)           1049088
['conv5_block2_outrgb[0][0]']
 nv2D)



 conv5_block3_1_conv_depth   (None, 8, 8, 512)           1049088
['conv5_block2_out_depth[0][0]
 (Conv2D)                                                            ']



 conv5_block3_1_bnrgb (Batc  (None, 8, 8, 512)           2048
['conv5_block3_1_convrgb[0][0]
 hNormalization)                                                     ']



 conv5_block3_1_bn_depth (B  (None, 8, 8, 512)           2048
['conv5_block3_1_conv_depth[0]
 atchNormalization)
[0]']


 conv5_block3_1_relurgb (Ac  (None, 8, 8, 512)           0
['conv5_block3_1_bnrgb[0][0]']
 tivation)



 conv5_block3_1_relu_depth   (None, 8, 8, 512)           0
['conv5_block3_1_bn_depth[0][0
(Activation)                                                                   ]']



 conv5_block3_2_convrgb (Co  (None, 8, 8, 512)            2359808
['conv5_block3_1_relurgb[0][0]
 nv2D)                                                                          ']



 conv5_block3_2_conv_depth   (None, 8, 8, 512)            2359808
['conv5_block3_1_relu_depth[0]
 (Conv2D)
[0]']


 conv5_block3_2_bnrgb (Batc  (None, 8, 8, 512)            2048
['conv5_block3_2_convrgb[0][0]
 hNormalization)                                                                ']



 conv5_block3_2_bn_depth (B  (None, 8, 8, 512)            2048
['conv5_block3_2_conv_depth[0]
 atchNormalization)
[0]']


 conv5_block3_2_relurgb (Ac  (None, 8, 8, 512)            0
['conv5_block3_2_bnrgb[0][0]']
 tivation)



 conv5_block3_2_relu_depth   (None, 8, 8, 512)            0
['conv5_block3_2_bn_depth[0][0

(Activation)                                                                   ']'



 conv5_block3_3_convrgb (Co  (None, 8, 8, 2048)           1050624
['conv5_block3_2_relurgb[0][0]
 nv2D)                                                                          ']



 conv5_block3_3_conv_depth   (None, 8, 8, 2048)           1050624
['conv5_block3_2_relu_depth[0]
 (Conv2D)
[0]']
```

```
 conv5_block3_3_bnrgb (Batc  (None, 8, 8, 2048)           8192
['conv5_block3_3_convrgb[0][0]
 hNormalization)                                                                   ']



 conv5_block3_3_bn_depth (B  (None, 8, 8, 2048)           8192
['conv5_block3_3_conv_depth[0]
 atchNormalization)
[0]']


 conv5_block3_addrgb (Add)   (None, 8, 8, 2048)           0
['conv5_block2_outrgb[0][0]',

'conv5_block3_3_bnrgb[0][0]']


 conv5_block3_add_depth (Ad  (None, 8, 8, 2048)           0
['conv5_block2_out_depth[0][0]
 d)                                                                                ',


'conv5_block3_3_bn_depth[0][0
                                                                                   ]'
]


 conv5_block3_outrgb (Activ  (None, 8, 8, 2048)           0
['conv5_block3_addrgb[0][0]']
 ation)



 conv5_block3_out_depth (Ac  (None, 8, 8, 2048)           0
['conv5_block3_add_depth[0][0]
 tivation)                                                                         ']



 conv2d_rgb_1 (Conv2D)       (None, 8, 8, 128)            2359424
['conv5_block3_outrgb[0][0]']


 conv2d_depth_1 (Conv2D)     (None, 8, 8, 128)            2359424
['conv5_block3_out_depth[0][0]
                                                                                   ']
 conv2d_rgb_2 (Conv2D)       (None, 8, 8, 256)            295168
['conv2d_rgb_1[0][0]']


 conv2d_depth_2 (Conv2D)     (None, 8, 8, 256)            295168
['conv2d_depth_1[0][0]']


 dropout_rgb (Dropout)       (None, 8, 8, 256)            0
['conv2d_rgb_2[0][0]']


 dropout_depth (Dropout)     (None, 8, 8, 256)            0
['conv2d_depth_2[0][0]']


 concatenate (Concatenate)   (None, 8, 8, 512)            0
['dropout_rgb[0][0]',

'dropout_depth[0][0]']


 conv2d_transpose (Conv2DTr  (None, 256, 256, 19)         3984590
['concatenate[0][0]']
 anspose)                                                 7



 reshape (Reshape)           (None, 256, 256, 19)         0
['conv2d_transpose[0][0]']


 activation (Activation)     (None, 256, 256, 19)         0
['reshape[0][0]']


======================================================================
============================
Total params: 92330515 (352.21 MB)
Trainable params: 92224275 (351.81 MB)
Non-trainable params: 106240 (415.00 KB)
______________________________________________________________________
____________________________
```

It is mentionable that I have also tried freezing the layers of pretrained model by setting trainable = False but that way even though the accuracy for models seemed to improve and became 55.8%, the predicted segmented images just showed the same thing for all of them. The issue I am facing, where freezing the layers of a pretrained model leads to identical segmented images despite variations in inputs, likely stems from a limited capacity for learning in the subsequent layers. By freezing the pretrained layers, I am leveraging their feature extraction capabilities, but if the following layers lack the capacity to discern between different inputs effectively, they may produce uniform outputs. To address this, I considered fine-tuning the pretrained layers while allowing the layers to be trainable, ensuring they adapt to the specific task at hand.

```
def resize_labels(labels, target_shape):
    # Resize the labels to the target shape
    resized_labels = tf.image.resize(labels, target_shape[:2])
    return resized_labels

# Define the target shape
height = 256
width = 256

# Resize train labels
resized_train_labels = resize_labels(train_loader.labels,
target_shape=(height, width))

# Resize test labels
resized_test_labels = resize_labels(test_loader.labels,
target_shape=(height, width))

# Resize validation labels
resized_val_labels = resize_labels(val_loader.labels,
target_shape=(height, width))
```

Compile the model with SGD(learning_rate=0.01, decay=1e-5, momentum=0.9) and loss="categorical_crossentropy"

Train the model on the “train” dataset and “validation”dataset for epochs =10.

```
# Define the optimizer with specific parameters
optimizer = tf.keras.optimizers.legacy.SGD(learning_rate=0.01,
decay=1e-6, momentum=0.9)

# Compile the model with the specified optimizer, loss function, and
evaluation metrics
model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
```

Evaluate the model on the training and test dataset. The results must be shown as

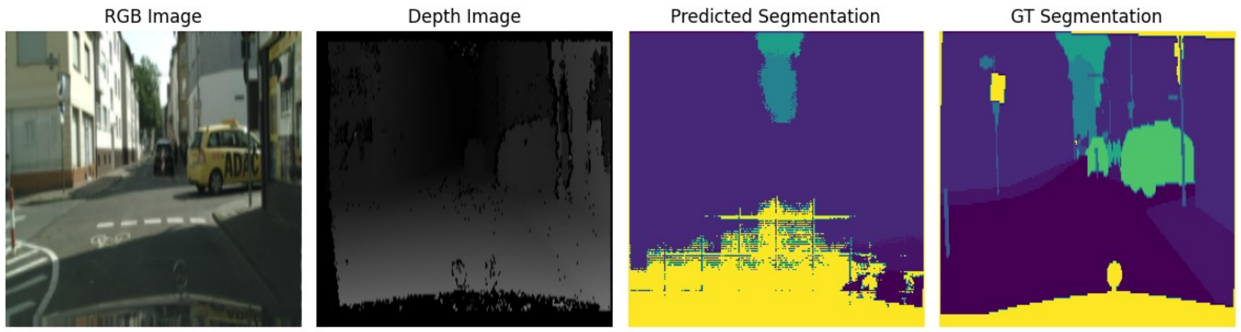
- Print loss and accuracy of model for test dataset.
- Predict semantically segmented images on 5 random example of test dataset.
- Visualize the 5 random examples alongside the ground truth and prediction.

```
epochs=10

# Train the model using the fit method with resized labels
history = model.fit(train_loader, epochs=epochs,
validation_data=val_loader,
                    )
# Evaluate the model on the test data with resized labels
test_loss, test_accuracy = model.evaluate(test_loader)


Epoch 1/10
50/50 [==============================] - 34s 680ms/step - loss: 0.7095
- accuracy: 0.7960 - val_loss: 3.2586 - val_accuracy: 0.2217
Epoch 2/10
50/50 [==============================] - 31s 617ms/step - loss: 0.6868
- accuracy: 0.8034 - val_loss: 3.2057 - val_accuracy: 0.2411
Epoch 3/10
50/50 [==============================] - 31s 611ms/step - loss: 0.6678
- accuracy: 0.8090 - val_loss: 3.1859 - val_accuracy: 0.2632
Epoch 4/10
50/50 [==============================] - 31s 622ms/step - loss: 0.6522
- accuracy: 0.8142 - val_loss: 3.0648 - val_accuracy: 0.2817
Epoch 5/10
50/50 [==============================] - 31s 614ms/step - loss: 0.6372
- accuracy: 0.8197 - val_loss: 2.9602 - val_accuracy: 0.3169
Epoch 6/10
50/50 [==============================] - 30s 604ms/step - loss: 0.6224
- accuracy: 0.8239 - val_loss: 3.0644 - val_accuracy: 0.2307
Epoch 7/10
50/50 [==============================] - 31s 610ms/step - loss: 0.6103
- accuracy: 0.8279 - val_loss: 2.8048 - val_accuracy: 0.3097
Epoch 8/10
50/50 [==============================] - 34s 670ms/step - loss: 0.5991
- accuracy: 0.8312 - val_loss: 2.6844 - val_accuracy: 0.3312
Epoch 9/10
50/50 [==============================] - 31s 609ms/step - loss: 0.5885
- accuracy: 0.8344 - val_loss: 2.3449 - val_accuracy: 0.3954
Epoch 10/10
50/50 [==============================] - 31s 606ms/step - loss: 0.5799
- accuracy: 0.8371 - val_loss: 1.9345 - val_accuracy: 0.4642
34/34 [==============================] - 42s 1s/step - loss: 2.1859 -
accuracy: 0.4108


# Print test accuracy, loss
print("Test Accuracy:", test_accuracy)
print("Test Loss:", test_loss)
test_accuracy_fusion=test_accuracy
```

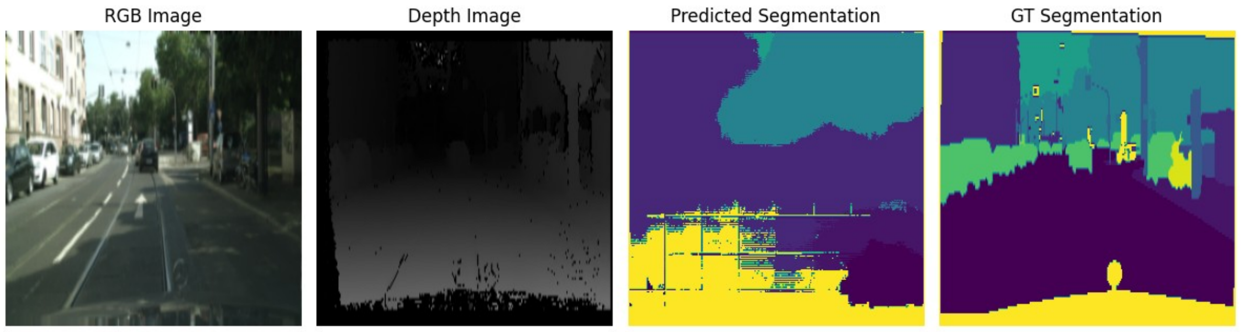
```
Test Accuracy: 0.41075384616851807
Test Loss: 2.1858878135681152
```

```python
# Predict semantically segmented images on 5 random examples from the
test dataset


def visualize(rgb_image, depth_image, label, prediction):
    # Create a figure with 1 row and 4 columns
    fig, axes = plt.subplots(1, 4, figsize=(12, 4))

    # Plot RGB image
    axes[0].imshow(rgb_image)
    axes[0].set_title('RGB Image')
    axes[0].axis('off')

    # Plot depth image
    axes[1].imshow(depth_image)
    axes[1].set_title('Depth Image')
    axes[1].axis('off')

    ## Plot ground truth label
    #plt.imshow(batch_depth[i][:, :, 0])

    axes[2].imshow(prediction.argmax(axis=2))
    axes[2].set_title('Predicted Segmentation')
    axes[2].axis('off')

    axes[3].imshow(label.argmax(axis=2))
    axes[3].set_title('GT Segmentation')
    axes[3].axis('off')

    # Plot model prediction


    plt.tight_layout()  # Adjust layout to prevent overlap
    plt.show()

# Visualize a single image and its prediction for each batch
for idx in range(5):
    data, label = test_loader[idx]
    prediction = model.predict(data)

    # Select the first image from the batch for visualization
    rgb_image = data[0][0]  # Selecting the first image from the batch
    depth_image = data[1][0]  # Selecting the first image from the
batch
    ground_truth_label = label[0]  # Selecting the label for the first
image from the batch
    prediction_label = prediction[0]  # Selecting the prediction for
the first image from the batch
```

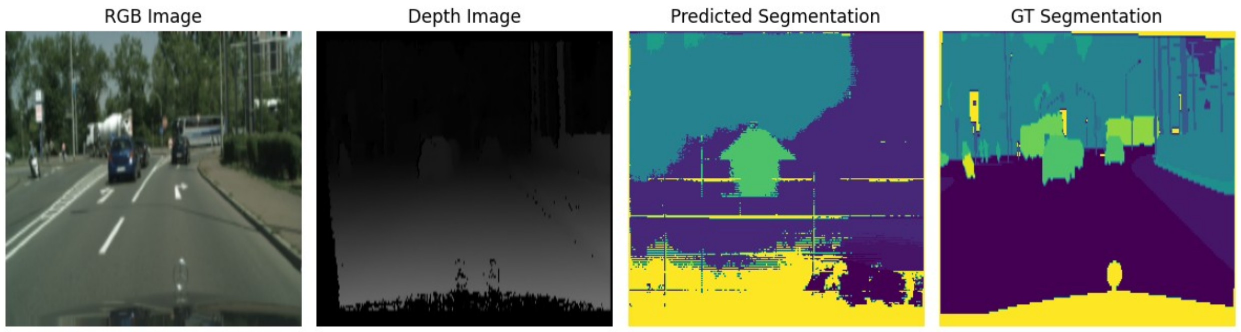
```
    # Visualize the image and its prediction
    visualize(rgb_image, depth_image, ground_truth_label,
prediction_label)

1/1 [==============================] - 0s 42ms/step
```



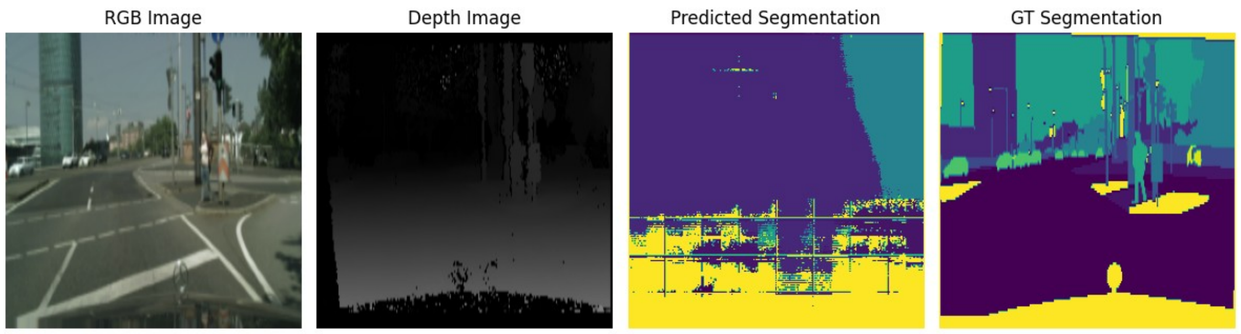
```
1/1 [==============================] - 0s 37ms/step
```



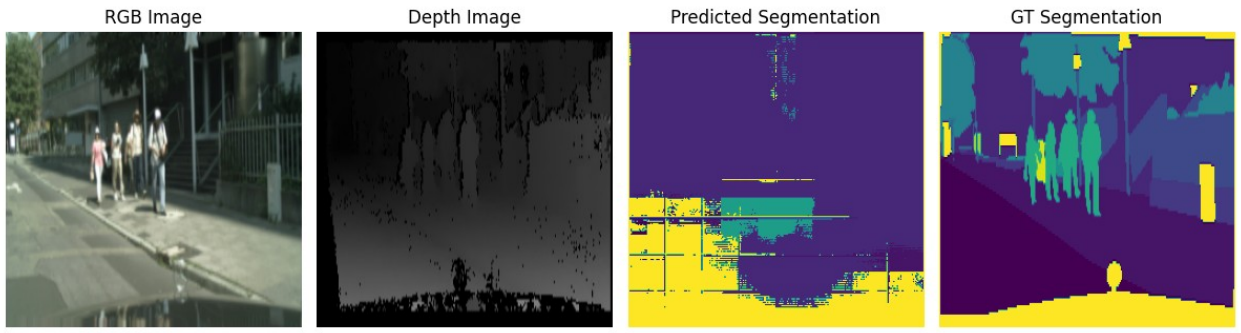
```
1/1 [==============================] - 0s 46ms/step
```



```
1/1 [==============================] - 0s 37ms/step
```



```
1/1 [==============================] - 0s 38ms/step
```



For my case, I did Training with batch size 12, testing with 6, and validation with 6 yielded an accuracy of 42% around and we can see that somehow it's detecting some borders and trying to detect cars, trees, building or road spaces but very hardly detecting humans. The details in the image are very intricate to learn indeed. I kept the hyperparameters just like how it is mentioned in the question. But batch size wasn't explicitly mentioned so I used my values and also experimented with other values to see the effects.

I conducted many additional experiments. But for the sake of simplicity of submission, this notebook just has the necessary parts that were asked in the question. For instance, when I used batch size 3, it detected almost everything very well and the prediction plotting showed how each component can be differentiated correctly as Car, Pedestrians, Trees, etc very sharply, and accuracy at that time was 78%. Training with a larger batch size and the same number of epochs and other conditions makes the model learn slower. But even with larger batch size, training for a higher number of epochs gradually increased training and validation accuracy while decreasing their losses.

Littler batch sizes combined with a set number of epochs typically outperform bigger batch sizes for a variety of reasons, which explains why lower batch sizes and epoch 10 performed better than larger batch sizes and epoch 10. First off, the model can update its parameters more often with lower batch sizes, which can result in faster convergence and improved generalization. The model is able to investigate a greater range of gradients and possibly avoid becoming trapped in local minima since it receives more varied samples with every batch. Smaller batches also demand less memory than bigger batch sizes, which allows for greater use of the computational resources that are available. Particularly in settings with constrained GPU memory or processing power, this can lead to more effective training. They are also preferred when the dataset is large or when dealing with diverse or complex images, as smaller batches allow for more varied gradient updates which are all applicable in our case of semantic segmentation.

Overall, even when the number of epochs is held constant, lower batch sizes generally result in higher performance compared to larger ones due to the combination of regularization effects, more frequent parameter updates, and effective resource management. More epochs are needed for convergence with bigger batch sizes due to the larger learning steps, less diversified representation learning, and noisier gradient estimations. Compared to smaller batch sizes, this slows down optimization and requires more epochs to find an ideal solution.

Therefore, the reason I'm using such a small one is due to computational problems in Google colab and kernel restart and GPU limitations. For some reason, I got logged out of Colab usage for a certain account within a maximum of 3 hours and wasn't allowed to use it for another 12 hours so I had to start from another account. Initially wanted to use 32 but my training was very slow at 32 and sometimes took hours to start only to get interrupted but kernel interruption or GPU usage stoppage midway. Hence for a very high batch size the training was very slow even on the Colab GPU and computational problems were more severe and that's why I opted for a batch size like this to showcase my results. In the future, I wanna go for Colab Pro to see how far this model can go.

Extra 5 points :

Implement FCNs for each sing modality and compare their accuracy with fusion model. I need the result of the following table in the same notebook.

```
class DataLoader(Sequence):
    def __init__(self, rgb_dir=None, depth_dir=None, label_dir=None,
image_size=(256, 256, 3), batch_size=32, max_samples=None,
num_images=None, num_classes=19):
        """
        DataLoader class for loading data and labels for semantic
segmentation tasks.

        Args:
            rgb_dir (str): Directory containing RGB images. Defaults
to None.
            depth_dir (str): Directory containing depth images.
Defaults to None.
            label_dir (str): Directory containing label images.
Defaults to None.
            image_size (tuple): Size of input images in the format
(height, width, channels). Defaults to (256, 256, 3).
            batch_size (int): Batch size for data loading. Defaults to
32.
            max_samples (int): Maximum number of samples to consider.
Defaults to None.
            num_images (int): Number of images in the dataset.
Defaults to None.
            num_classes (int): Number of classes in the segmentation
task. Defaults to 19.
```

```
        """
        # Initialize parameters
        self.rgb_dir = rgb_dir
        self.depth_dir = depth_dir
        self.label_dir = label_dir
        self.image_size = image_size
        self.batch_size = batch_size

        # Determine if RGB and depth modes are enabled
        self.rgb_mode = rgb_dir is not None
        self.depth_mode = depth_dir is not None

        # Set the number of samples
        self.num_samples = num_images if num_images is not None else
len(os.listdir(label_dir))

        # Load labels
        self.labels = self.load_labels()

        # Set the number of classes
        self.num_classes = num_classes

    def load_labels(self):
        """
        Load label images from the specified directory.

        Returns:
            np.array: Array containing loaded label images.
        """
        labels = []
        label_files = sorted(os.listdir(self.label_dir))
        for file in label_files:
            label = np.load(os.path.join(self.label_dir, file))
            labels.append(label)
        return np.stack(labels)

    def __len__(self):
        """
        Calculate the length of the DataLoader.

        Returns:
            int: Length of the DataLoader.
        """
        return int(np.ceil(self.num_samples / self.batch_size))

    def __getitem__(self, idx):
        """
        Retrieve data and labels for a given index.

        Args:
            idx (int): Index of the batch.

        Returns:
            tuple: Tuple containing data and labels.
        """
        # Define batch indices
        batch_indices = range(idx * self.batch_size, min((idx + 1) *
self.batch_size, self.num_samples))

        # Process data and labels based on mode and validation
condition
        if self.label_dir == val_label_dir:
            if self.rgb_mode:
                batch_rgb_files = [os.path.join(self.rgb_dir,
str(index + 200) + '.npy') for index in batch_indices]
                batch_rgb_data = self.__generate_data(batch_rgb_files)

            if self.depth_mode:
                batch_depth_files = [os.path.join(self.depth_dir,
str(index + 200) + '.npy') for index in batch_indices]
                batch_depth_data =
self.__generate_data(batch_depth_files)

            batch_label_files = [os.path.join(self.label_dir,
str(index + 200) + '.npy') for index in batch_indices]
            batch_labels_resized =
self.__generate_labels(batch_label_files)
        else:
            if self.rgb_mode:
                batch_rgb_files = [os.path.join(self.rgb_dir,
str(index) + '.npy') for index in batch_indices]
                batch_rgb_data = self.__generate_data(batch_rgb_files)

            if self.depth_mode:
                batch_depth_files = [os.path.join(self.depth_dir,
str(index) + '.npy') for index in batch_indices]
                batch_depth_data =
self.__generate_data(batch_depth_files)

            batch_label_files = [os.path.join(self.label_dir,
str(index) + '.npy') for index in batch_indices]
            batch_labels_resized =
self.__generate_labels(batch_label_files)

        # Return data and labels based on mode
        if self.rgb_mode and self.depth_mode:
            return [batch_rgb_data, batch_depth_data],
batch_labels_resized
        elif self.rgb_mode:
            return batch_rgb_data, batch_labels_resized
        elif self.depth_mode:
            return batch_depth_data, batch_labels_resized

    def __generate_data(self, files):
        """
        Generate data from the specified files.

        Args:
            files (list): List of file paths.

        Returns:
            np.array: Array containing generated data.
        """
        data = []
        for file in files:
            img = np.load(file)
            img_resized = cv2.resize(img, (self.image_size[1],
self.image_size[0]))
            if self.depth_mode:
                img_resized = np.repeat(img_resized[:, :, np.newaxis],
3, axis=2)
            data.append(img_resized)
        return np.array(data)

    def __generate_labels(self, labels):
        """
        Generate labels from the specified label files.

        Args:
            labels (list): List of label file paths.

        Returns:
            np.array: Array containing generated labels.
        """
        data = []
        for label in labels:
            label_data = np.load(label)
            unique_labels = np.unique(labels)
            num_classes = len(unique_labels)
            label_resized = cv2.resize(label_data,
(self.image_size[1], self.image_size[0]))
            label_encoded =
tf.keras.utils.to_categorical(label_resized,
num_classes=self.num_classes)
            data.append(label_encoded)
        return np.array(data)

# Create DataLoader instances for RGB and Depth modes with batch size
train_loader_rgb = DataLoader(rgb_dir=train_rgb_dir,
label_dir=train_label_dir, image_size=(256, 256, 3), batch_size=12,
```

```python
num_images=600)
test_loader_rgb = DataLoader(rgb_dir=test_rgb_dir,
label_dir=test_label_dir, image_size=(256, 256, 3), batch_size=6,
num_images=200)
val_loader_rgb = DataLoader(rgb_dir=val_rgb_dir,
label_dir=val_label_dir, image_size=(256, 256, 3), batch_size=6,
num_images=100)

train_loader_depth = DataLoader(depth_dir=train_depth_dir,
label_dir=train_label_dir, image_size=(256, 256, 3), batch_size=12,
num_images=600)
test_loader_depth = DataLoader(depth_dir=test_depth_dir,
label_dir=test_label_dir, image_size=(256, 256, 3), batch_size=6,
num_images=200)
val_loader_depth = DataLoader(depth_dir=val_depth_dir,
label_dir=val_label_dir, image_size=(256, 256, 3), batch_size=6,
num_images=100)


# Define input shape for RGB modality
input_rgb = Input(shape=(256, 256, 3), name='input_rgb')

# Load the pretrained ResNet50 model for RGB modality
pre_trained_model_rgb = ResNet50(weights='imagenet',
include_top=False, input_tensor=input_rgb)

# Remove unnecessary layers from the ResNet50 model
last_layer_rgb = pre_trained_model_rgb.layers[-1].output

# Additional layers specific to RGB modality
conv2d_rgb_1 = Conv2D(128, (3, 3), strides=(1, 1), activation='relu',
padding='same', name='conv2d_rgb_1')(last_layer_rgb)
conv2d_rgb_2 = Conv2D(256, (3, 3), strides=(1, 1), activation='relu',
padding='same', name='conv2d_rgb_2')(conv2d_rgb_1)
dropout_rgb = Dropout(0.2, name='dropout_rgb')(conv2d_rgb_2)

# Transposed convolutional layer
conv2d_transpose = Conv2DTranspose(19, (64, 64), strides=(32, 32),
padding='same', name='conv2d_transpose')(dropout_rgb)

# Reshape layer
reshaped_layer = Reshape((256, 256, 19), name='reshape')
(conv2d_transpose)

# Softmax activation layer
activation = Activation('softmax', name='activation')(reshaped_layer)

# Create the model for RGB modality
model_rgb = Model(inputs=[input_rgb], outputs=activation)
```

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

```
Model: "model_1"
________________________________________________________________________________
__________________________________
 Layer (type)                Output Shape                 Param #
Connected to
================================================================================
==================================
 input_rgb (InputLayer)      [(None, 256, 256, 3)]        0          []



 conv1_pad (ZeroPadding2D)   (None, 262, 262, 3)          0
['input_rgb[0][0]']


 conv1_conv (Conv2D)         (None, 128, 128, 64)         9472
['conv1_pad[0][0]']


 conv1_bn (BatchNormalizati  (None, 128, 128, 64)         256
['conv1_conv[0][0]']
 on)



 conv1_relu (Activation)     (None, 128, 128, 64)         0
['conv1_bn[0][0]']


 pool1_pad (ZeroPadding2D)   (None, 130, 130, 64)         0
['conv1_relu[0][0]']


 pool1_pool (MaxPooling2D)   (None, 64, 64, 64)           0
['pool1_pad[0][0]']


 conv2_block1_1_conv (Conv2  (None, 64, 64, 64)           4160
['pool1_pool[0][0]']
 D)



 conv2_block1_1_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block1_1_conv[0][0]']
 rmalization)
 conv2_block1_1_relu (Activ  (None, 64, 64, 64)           0
['conv2_block1_1_bn[0][0]']
 ation)



 conv2_block1_2_conv (Conv2  (None, 64, 64, 64)           36928
['conv2_block1_1_relu[0][0]']
 D)



 conv2_block1_2_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block1_2_conv[0][0]']
 rmalization)



 conv2_block1_2_relu (Activ  (None, 64, 64, 64)           0
['conv2_block1_2_bn[0][0]']
 ation)



 conv2_block1_0_conv (Conv2  (None, 64, 64, 256)          16640
['pool1_pool[0][0]']
 D)



 conv2_block1_3_conv (Conv2  (None, 64, 64, 256)          16640
['conv2_block1_2_relu[0][0]']
 D)



 conv2_block1_0_bn (BatchNo  (None, 64, 64, 256)          1024
['conv2_block1_0_conv[0][0]']
 rmalization)



 conv2_block1_3_bn (BatchNo  (None, 64, 64, 256)          1024
['conv2_block1_3_conv[0][0]']
 rmalization)



 conv2_block1_add (Add)      (None, 64, 64, 256)          0
```

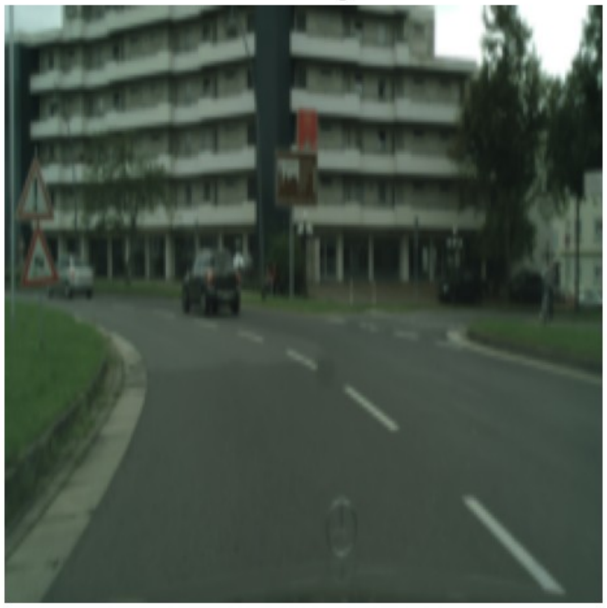
```
['conv2_block1_0_bn[0][0]',

'conv2_block1_3_bn[0][0]']


 conv2_block1_out (Activati  (None, 64, 64, 256)          0
['conv2_block1_add[0][0]']
 on)



 conv2_block2_1_conv (Conv2  (None, 64, 64, 64)           16448
['conv2_block1_out[0][0]']
 D)



 conv2_block2_1_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block2_1_conv[0][0]']
 rmalization)



 conv2_block2_1_relu (Activ  (None, 64, 64, 64)           0
['conv2_block2_1_bn[0][0]']
 ation)



 conv2_block2_2_conv (Conv2  (None, 64, 64, 64)           36928
['conv2_block2_1_relu[0][0]']
 D)



 conv2_block2_2_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block2_2_conv[0][0]']
 rmalization)



 conv2_block2_2_relu (Activ  (None, 64, 64, 64)           0
['conv2_block2_2_bn[0][0]']
 ation)



 conv2_block2_3_conv (Conv2  (None, 64, 64, 256)          16640
['conv2_block2_2_relu[0][0]']
 D)
 conv2_block2_3_bn (BatchNo  (None, 64, 64, 256)          1024
['conv2_block2_3_conv[0][0]']
 rmalization)



 conv2_block2_add (Add)      (None, 64, 64, 256)          0
['conv2_block1_out[0][0]',

'conv2_block2_3_bn[0][0]']


 conv2_block2_out (Activati  (None, 64, 64, 256)          0
['conv2_block2_add[0][0]']
 on)



 conv2_block3_1_conv (Conv2  (None, 64, 64, 64)           16448
['conv2_block2_out[0][0]']
 D)



 conv2_block3_1_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block3_1_conv[0][0]']
 rmalization)



 conv2_block3_1_relu (Activ  (None, 64, 64, 64)           0
['conv2_block3_1_bn[0][0]']
 ation)



 conv2_block3_2_conv (Conv2  (None, 64, 64, 64)           36928
['conv2_block3_1_relu[0][0]']
 D)



 conv2_block3_2_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block3_2_conv[0][0]']
 rmalization)



 conv2_block3_2_relu (Activ  (None, 64, 64, 64)           0
['conv2_block3_2_bn[0][0]']
 ation)



 conv2_block3_3_conv (Conv2  (None, 64, 64, 256)          16640
['conv2_block3_2_relu[0][0]']
 D)



 conv2_block3_3_bn (BatchNo  (None, 64, 64, 256)          1024
['conv2_block3_3_conv[0][0]']
 rmalization)



 conv2_block3_add (Add)      (None, 64, 64, 256)          0
['conv2_block2_out[0][0]',

'conv2_block3_3_bn[0][0]']


 conv2_block3_out (Activati  (None, 64, 64, 256)          0
['conv2_block3_add[0][0]']
 on)



 conv3_block1_1_conv (Conv2  (None, 32, 32, 128)          32896
['conv2_block3_out[0][0]']
 D)



 conv3_block1_1_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block1_1_conv[0][0]']
 rmalization)



 conv3_block1_1_relu (Activ  (None, 32, 32, 128)          0
['conv3_block1_1_bn[0][0]']
 ation)



 conv3_block1_2_conv (Conv2  (None, 32, 32, 128)          147584
['conv3_block1_1_relu[0][0]']
 D)
 conv3_block1_2_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block1_2_conv[0][0]']
 rmalization)



 conv3_block1_2_relu (Activ  (None, 32, 32, 128)          0
['conv3_block1_2_bn[0][0]']
 ation)



 conv3_block1_0_conv (Conv2  (None, 32, 32, 512)          131584
['conv2_block3_out[0][0]']
 D)



 conv3_block1_3_conv (Conv2  (None, 32, 32, 512)          66048
['conv3_block1_2_relu[0][0]']
 D)



 conv3_block1_0_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block1_0_conv[0][0]']
 rmalization)



 conv3_block1_3_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block1_3_conv[0][0]']
 rmalization)



 conv3_block1_add (Add)      (None, 32, 32, 512)          0
['conv3_block1_0_bn[0][0]',

'conv3_block1_3_bn[0][0]']


 conv3_block1_out (Activati  (None, 32, 32, 512)          0
['conv3_block1_add[0][0]']
 on)



 conv3_block2_1_conv (Conv2  (None, 32, 32, 128)          65664
['conv3_block1_out[0][0]']
 D)



 conv3_block2_1_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block2_1_conv[0][0]']
 rmalization)



 conv3_block2_1_relu (Activ  (None, 32, 32, 128)          0
['conv3_block2_1_bn[0][0]']
 ation)



 conv3_block2_2_conv (Conv2  (None, 32, 32, 128)          147584
['conv3_block2_1_relu[0][0]']
 D)



 conv3_block2_2_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block2_2_conv[0][0]']
 rmalization)



 conv3_block2_2_relu (Activ  (None, 32, 32, 128)          0
['conv3_block2_2_bn[0][0]']
 ation)



 conv3_block2_3_conv (Conv2  (None, 32, 32, 512)          66048
['conv3_block2_2_relu[0][0]']
 D)



 conv3_block2_3_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block2_3_conv[0][0]']
 rmalization)



 conv3_block2_add (Add)      (None, 32, 32, 512)          0
['conv3_block1_out[0][0]',

'conv3_block2_3_bn[0][0]']
```

```
 conv3_block2_out (Activati  (None, 32, 32, 512)          0
['conv3_block2_add[0][0]']
 on)



 conv3_block3_1_conv (Conv2  (None, 32, 32, 128)          65664
['conv3_block2_out[0][0]']
 D)



 conv3_block3_1_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block3_1_conv[0][0]']
 rmalization)



 conv3_block3_1_relu (Activ  (None, 32, 32, 128)          0
['conv3_block3_1_bn[0][0]']
 ation)



 conv3_block3_2_conv (Conv2  (None, 32, 32, 128)          147584
['conv3_block3_1_relu[0][0]']
 D)



 conv3_block3_2_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block3_2_conv[0][0]']
 rmalization)



 conv3_block3_2_relu (Activ  (None, 32, 32, 128)          0
['conv3_block3_2_bn[0][0]']
 ation)



 conv3_block3_3_conv (Conv2  (None, 32, 32, 512)          66048
['conv3_block3_2_relu[0][0]']
 D)



 conv3_block3_3_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block3_3_conv[0][0]']
 rmalization)



 conv3_block3_add (Add)      (None, 32, 32, 512)          0
['conv3_block2_out[0][0]',

'conv3_block3_3_bn[0][0]']


 conv3_block3_out (Activati  (None, 32, 32, 512)          0
['conv3_block3_add[0][0]']
 on)



 conv3_block4_1_conv (Conv2  (None, 32, 32, 128)          65664
['conv3_block3_out[0][0]']
 D)



 conv3_block4_1_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block4_1_conv[0][0]']
 rmalization)



 conv3_block4_1_relu (Activ  (None, 32, 32, 128)          0
['conv3_block4_1_bn[0][0]']
 ation)



 conv3_block4_2_conv (Conv2  (None, 32, 32, 128)          147584
['conv3_block4_1_relu[0][0]']
 D)



 conv3_block4_2_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block4_2_conv[0][0]']
 rmalization)



 conv3_block4_2_relu (Activ  (None, 32, 32, 128)          0
['conv3_block4_2_bn[0][0]']
 ation)
 conv3_block4_3_conv (Conv2  (None, 32, 32, 512)          66048
['conv3_block4_2_relu[0][0]']
 D)



 conv3_block4_3_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block4_3_conv[0][0]']
 rmalization)



 conv3_block4_add (Add)      (None, 32, 32, 512)          0
['conv3_block3_out[0][0]',

'conv3_block4_3_bn[0][0]']


 conv3_block4_out (Activati  (None, 32, 32, 512)          0
['conv3_block4_add[0][0]']
 on)



 conv4_block1_1_conv (Conv2  (None, 16, 16, 256)          131328
['conv3_block4_out[0][0]']
 D)



 conv4_block1_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block1_1_conv[0][0]']
 rmalization)



 conv4_block1_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block1_1_bn[0][0]']
 ation)



 conv4_block1_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block1_1_relu[0][0]']
 D)



 conv4_block1_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block1_2_conv[0][0]']
 rmalization)



 conv4_block1_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block1_2_bn[0][0]']
 ation)



 conv4_block1_0_conv (Conv2  (None, 16, 16, 1024)         525312
['conv3_block4_out[0][0]']
 D)



 conv4_block1_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block1_2_relu[0][0]']
 D)



 conv4_block1_0_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block1_0_conv[0][0]']
 rmalization)



 conv4_block1_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block1_3_conv[0][0]']
 rmalization)



 conv4_block1_add (Add)      (None, 16, 16, 1024)         0
['conv4_block1_0_bn[0][0]',

'conv4_block1_3_bn[0][0]']


 conv4_block1_out (Activati  (None, 16, 16, 1024)         0
['conv4_block1_add[0][0]']
 on)



 conv4_block2_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block1_out[0][0]']
 D)
 conv4_block2_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block2_1_conv[0][0]']
 rmalization)



 conv4_block2_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block2_1_bn[0][0]']
 ation)



 conv4_block2_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block2_1_relu[0][0]']
 D)



 conv4_block2_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block2_2_conv[0][0]']
 rmalization)



 conv4_block2_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block2_2_bn[0][0]']
 ation)



 conv4_block2_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block2_2_relu[0][0]']
 D)



 conv4_block2_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block2_3_conv[0][0]']
 rmalization)



 conv4_block2_add (Add)      (None, 16, 16, 1024)         0
['conv4_block1_out[0][0]',

'conv4_block2_3_bn[0][0]']


 conv4_block2_out (Activati  (None, 16, 16, 1024)         0
['conv4_block2_add[0][0]']
 on)



 conv4_block3_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block2_out[0][0]']
 D)



 conv4_block3_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block3_1_conv[0][0]']
 rmalization)



 conv4_block3_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block3_1_bn[0][0]']
 ation)



 conv4_block3_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block3_1_relu[0][0]']
 D)



 conv4_block3_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block3_2_conv[0][0]']
 rmalization)



 conv4_block3_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block3_2_bn[0][0]']
 ation)



 conv4_block3_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block3_2_relu[0][0]']
 D)



 conv4_block3_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block3_3_conv[0][0]']
 rmalization)
 conv4_block3_add (Add)      (None, 16, 16, 1024)        0
['conv4_block2_out[0][0]',

'conv4_block3_3_bn[0][0]']


 conv4_block3_out (Activati  (None, 16, 16, 1024)        0
['conv4_block3_add[0][0]']
 on)



 conv4_block4_1_conv (Conv2  (None, 16, 16, 256)         262400
['conv4_block3_out[0][0]']
 D)



 conv4_block4_1_bn (BatchNo  (None, 16, 16, 256)         1024
['conv4_block4_1_conv[0][0]']
 rmalization)



 conv4_block4_1_relu (Activ  (None, 16, 16, 256)         0
['conv4_block4_1_bn[0][0]']
 ation)



 conv4_block4_2_conv (Conv2  (None, 16, 16, 256)         590080
['conv4_block4_1_relu[0][0]']
 D)



 conv4_block4_2_bn (BatchNo  (None, 16, 16, 256)         1024
['conv4_block4_2_conv[0][0]']
 rmalization)



 conv4_block4_2_relu (Activ  (None, 16, 16, 256)         0
['conv4_block4_2_bn[0][0]']
 ation)



 conv4_block4_3_conv (Conv2  (None, 16, 16, 1024)        263168
['conv4_block4_2_relu[0][0]']
 D)



 conv4_block4_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block4_3_conv[0][0]']
 rmalization)



 conv4_block4_add (Add)      (None, 16, 16, 1024)         0
['conv4_block3_out[0][0]',

'conv4_block4_3_bn[0][0]']


 conv4_block4_out (Activati  (None, 16, 16, 1024)         0
['conv4_block4_add[0][0]']
 on)



 conv4_block5_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block4_out[0][0]']
 D)



 conv4_block5_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block5_1_conv[0][0]']
 rmalization)



 conv4_block5_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block5_1_bn[0][0]']
 ation)



 conv4_block5_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block5_1_relu[0][0]']
 D)



 conv4_block5_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block5_2_conv[0][0]']
 rmalization)
 conv4_block5_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block5_2_bn[0][0]']
 ation)



 conv4_block5_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block5_2_relu[0][0]']
 D)



 conv4_block5_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block5_3_conv[0][0]']
 rmalization)



 conv4_block5_add (Add)      (None, 16, 16, 1024)         0
['conv4_block4_out[0][0]',

'conv4_block5_3_bn[0][0]']


 conv4_block5_out (Activati  (None, 16, 16, 1024)         0
['conv4_block5_add[0][0]']
 on)



 conv4_block6_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block5_out[0][0]']
 D)



 conv4_block6_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block6_1_conv[0][0]']
 rmalization)



 conv4_block6_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block6_1_bn[0][0]']
 ation)



 conv4_block6_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block6_1_relu[0][0]']
 D)



 conv4_block6_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block6_2_conv[0][0]']
 rmalization)



 conv4_block6_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block6_2_bn[0][0]']
 ation)



 conv4_block6_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block6_2_relu[0][0]']
 D)



 conv4_block6_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block6_3_conv[0][0]']
 rmalization)



 conv4_block6_add (Add)      (None, 16, 16, 1024)         0
['conv4_block5_out[0][0]',

'conv4_block6_3_bn[0][0]']


 conv4_block6_out (Activati  (None, 16, 16, 1024)         0
['conv4_block6_add[0][0]']
 on)



 conv5_block1_1_conv (Conv2  (None, 8, 8, 512)            524800
['conv4_block6_out[0][0]']
 D)



 conv5_block1_1_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block1_1_conv[0][0]']
 rmalization)
 conv5_block1_1_relu (Activ  (None, 8, 8, 512)            0
['conv5_block1_1_bn[0][0]']
 ation)



 conv5_block1_2_conv (Conv2  (None, 8, 8, 512)            2359808
['conv5_block1_1_relu[0][0]']
 D)



 conv5_block1_2_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block1_2_conv[0][0]']
 rmalization)



 conv5_block1_2_relu (Activ  (None, 8, 8, 512)            0
['conv5_block1_2_bn[0][0]']
 ation)



 conv5_block1_0_conv (Conv2  (None, 8, 8, 2048)           2099200
['conv4_block6_out[0][0]']
 D)



 conv5_block1_3_conv (Conv2  (None, 8, 8, 2048)           1050624
['conv5_block1_2_relu[0][0]']
 D)



 conv5_block1_0_bn (BatchNo  (None, 8, 8, 2048)           8192
['conv5_block1_0_conv[0][0]']
 rmalization)



 conv5_block1_3_bn (BatchNo  (None, 8, 8, 2048)           8192
['conv5_block1_3_conv[0][0]']
 rmalization)



 conv5_block1_add (Add)      (None, 8, 8, 2048)           0
```

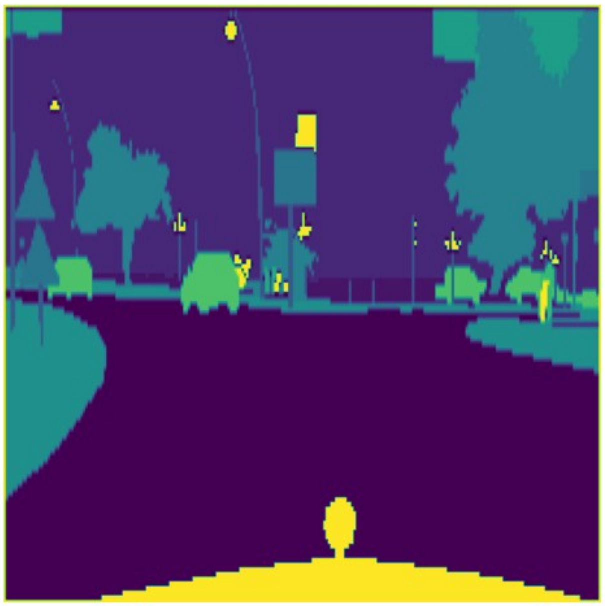
```
['conv5_block1_0_bn[0][0]',

'conv5_block1_3_bn[0][0]']


 conv5_block1_out (Activati  (None, 8, 8, 2048)           0
['conv5_block1_add[0][0]']
 on)



 conv5_block2_1_conv (Conv2  (None, 8, 8, 512)            1049088
['conv5_block1_out[0][0]']
 D)



 conv5_block2_1_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block2_1_conv[0][0]']
 rmalization)



 conv5_block2_1_relu (Activ  (None, 8, 8, 512)            0
['conv5_block2_1_bn[0][0]']
 ation)



 conv5_block2_2_conv (Conv2  (None, 8, 8, 512)            2359808
['conv5_block2_1_relu[0][0]']
 D)



 conv5_block2_2_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block2_2_conv[0][0]']
 rmalization)



 conv5_block2_2_relu (Activ  (None, 8, 8, 512)            0
['conv5_block2_2_bn[0][0]']
 ation)



 conv5_block2_3_conv (Conv2  (None, 8, 8, 2048)           1050624
['conv5_block2_2_relu[0][0]']
 D)
 conv5_block2_3_bn (BatchNo  (None, 8, 8, 2048)           8192
['conv5_block2_3_conv[0][0]']
 rmalization)



 conv5_block2_add (Add)      (None, 8, 8, 2048)           0
['conv5_block1_out[0][0]',

'conv5_block2_3_bn[0][0]']


 conv5_block2_out (Activati  (None, 8, 8, 2048)           0
['conv5_block2_add[0][0]']
 on)



 conv5_block3_1_conv (Conv2  (None, 8, 8, 512)            1049088
['conv5_block2_out[0][0]']
 D)



 conv5_block3_1_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block3_1_conv[0][0]']
 rmalization)



 conv5_block3_1_relu (Activ  (None, 8, 8, 512)            0
['conv5_block3_1_bn[0][0]']
 ation)



 conv5_block3_2_conv (Conv2  (None, 8, 8, 512)            2359808
['conv5_block3_1_relu[0][0]']
 D)



 conv5_block3_2_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block3_2_conv[0][0]']
 rmalization)



 conv5_block3_2_relu (Activ  (None, 8, 8, 512)            0
['conv5_block3_2_bn[0][0]']
 ation)



 conv5_block3_3_conv (Conv2  (None, 8, 8, 2048)           1050624
['conv5_block3_2_relu[0][0]']
 D)



 conv5_block3_3_bn (BatchNo  (None, 8, 8, 2048)           8192
['conv5_block3_3_conv[0][0]']
 rmalization)



 conv5_block3_add (Add)      (None, 8, 8, 2048)           0
['conv5_block2_out[0][0]',

'conv5_block3_3_bn[0][0]']


 conv5_block3_out (Activati  (None, 8, 8, 2048)           0
['conv5_block3_add[0][0]']
 on)



 conv2d_rgb_1 (Conv2D)       (None, 8, 8, 128)            2359424
['conv5_block3_out[0][0]']


 conv2d_rgb_2 (Conv2D)       (None, 8, 8, 256)            295168
['conv2d_rgb_1[0][0]']


 dropout_rgb (Dropout)       (None, 8, 8, 256)            0
['conv2d_rgb_2[0][0]']


 conv2d_transpose (Conv2DTr  (None, 256, 256, 19)         1992296
['dropout_rgb[0][0]']
 anspose)                                                 3



 reshape (Reshape)           (None, 256, 256, 19)         0
['conv2d_transpose[0][0]']
```

```
 activation (Activation)     (None, 256, 256, 19)         0
['reshape[0][0]']


=======================================================================
============================
Total params: 46165267 (176.11 MB)
Trainable params: 46112147 (175.90 MB)
Non-trainable params: 53120 (207.50 KB)
_______________________________________________________________________
____________________________


def plot_rgb_with_gt_and_shapes(rgb_images, gt_labels, num_samples=5):
    fig, axes = plt.subplots(num_samples, 2, figsize=(10, num_samples
* 5))

    for i in range(num_samples):
        # Plot RGB image
        axes[i, 0].imshow(rgb_images[i])
        axes[i, 0].set_title('RGB Image')
        axes[i, 0].axis('off')

        # Plot Ground Truth label
        gt_map_single_channel = np.argmax(gt_labels[i], axis=-1)
        axes[i, 1].imshow(gt_map_single_channel, vmin=0, vmax=18,
cmap='viridis')
        axes[i, 1].set_title('Ground Truth Label')
        axes[i, 1].axis('off')

    plt.tight_layout()
    plt.show()

# Plotting the first 5 samples
# Iterate through train_loader_rgb to access RGB data and labels
rgb_images = []
gt_labels = []

for i in range(len(train_loader_rgb)):
    batch_rgb_data, batch_labels = train_loader_rgb[i]
    rgb_images.extend(batch_rgb_data)
    gt_labels.extend(batch_labels)

# Plotting the first 5 samples
plot_rgb_with_gt_and_shapes(rgb_images, gt_labels, num_samples=5)
```

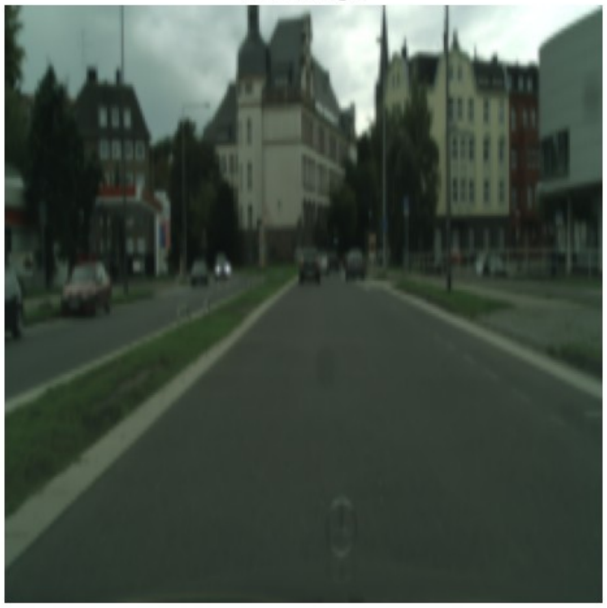
RGB Image



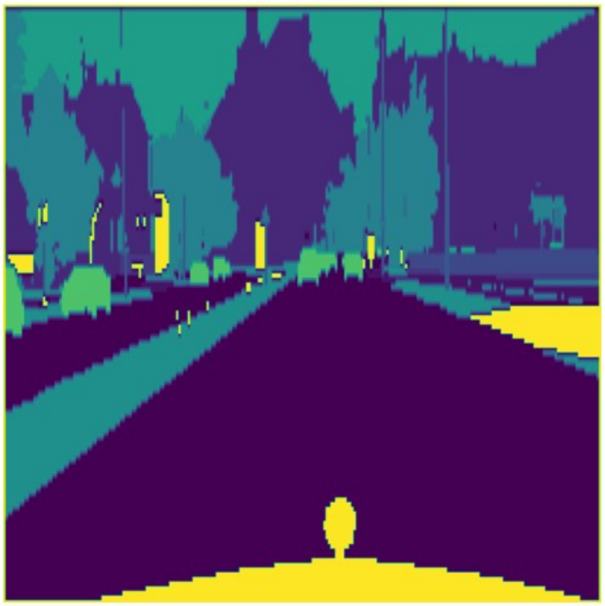
Ground Truth Label



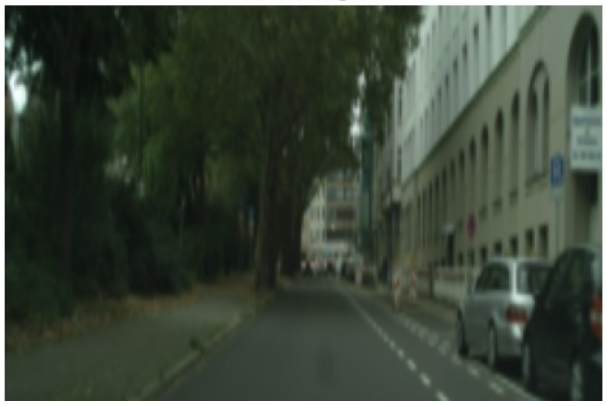
RGB Image



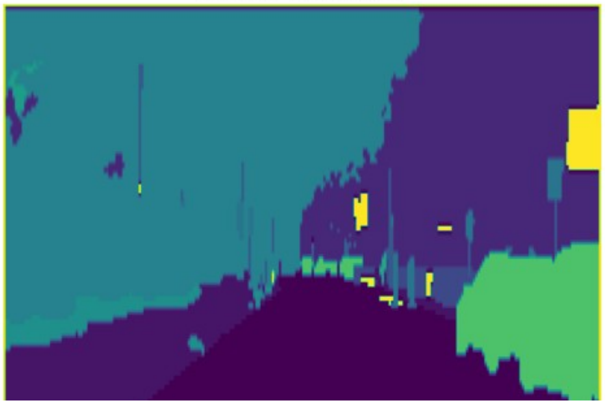
Ground Truth Label



RGB Image



Ground Truth Label



```
def resize_labels(labels, target_shape):
    # Resize labels using bilinear interpolation
    resized_labels = tf.image.resize(labels, target_shape[:2])
    return resized_labels

# Define the height and width for resizing
height = 256
width = 256
num_classes = 19  # Since there are 19 classes

# Define the target shape for resizing
target_shape = (height, width)

# Resize train labels
resized_train_labels = resize_labels(train_loader_rgb.labels,
target_shape=target_shape + (num_classes,))

# Resize test labels
resized_test_labels = resize_labels(test_loader_rgb.labels,
target_shape=target_shape + (num_classes,))
print(resized_test_labels.shape)  # Print the shape of resized test
labels

# Resize validation labels
resized_val_labels = resize_labels(val_loader_rgb.labels,
target_shape=target_shape + (num_classes,))



# Define input shape for RGB modality
input_shape = (256, 256, 3)  # Define the input shape for RGB images

# Define the number of classes
num_classes = 12  # Define the number of classes for classification

# Define the number of epochs for training
epochs = 10  # Define the number of epochs to train the model

# Compile the model
model_rgb.compile(optimizer=optimizer,  # Use the defined optimizer
                  loss='categorical_crossentropy',  # Use categorical
cross-entropy loss
                  metrics=['accuracy'])  # Evaluate model performance
using accuracy metric

# Train the model
model_rgb.fit(train_loader_rgb, epochs=epochs,
validation_data=val_loader_rgb)  # Train the model using training and
validation data loaders
```

```
Epoch 1/10
50/50 [==============================] - 271s 3s/step - loss: 2.5701 -
accuracy: 0.3014 - val_loss: 2.3294 - val_accuracy: 0.3330
Epoch 2/10
50/50 [==============================] - 18s 364ms/step - loss: 1.7262
- accuracy: 0.4250 - val_loss: 1.9740 - val_accuracy: 0.4424
Epoch 3/10
50/50 [==============================] - 18s 358ms/step - loss: 1.4175
- accuracy: 0.5510 - val_loss: 1.8351 - val_accuracy: 0.4927
Epoch 4/10
50/50 [==============================] - 18s 350ms/step - loss: 1.2266
- accuracy: 0.5784 - val_loss: 1.7841 - val_accuracy: 0.4639
Epoch 5/10
50/50 [==============================] - 18s 359ms/step - loss: 1.1017
- accuracy: 0.6448 - val_loss: 1.8049 - val_accuracy: 0.4245
Epoch 6/10
50/50 [==============================] - 18s 349ms/step - loss: 0.9942
- accuracy: 0.6980 - val_loss: 2.0184 - val_accuracy: 0.4469
Epoch 7/10
50/50 [==============================] - 18s 347ms/step - loss: 0.9257
- accuracy: 0.7202 - val_loss: 2.0959 - val_accuracy: 0.3762
Epoch 8/10
50/50 [==============================] - 19s 369ms/step - loss: 0.8709
- accuracy: 0.7373 - val_loss: 2.0207 - val_accuracy: 0.4395
Epoch 9/10
50/50 [==============================] - 17s 346ms/step - loss: 0.8271
- accuracy: 0.7525 - val_loss: 1.7974 - val_accuracy: 0.4697
Epoch 10/10
50/50 [==============================] - 18s 353ms/step - loss: 0.7880
- accuracy: 0.7666 - val_loss: 1.8520 - val_accuracy: 0.4715

<keras.src.callbacks.History at 0x7804ef929c30>

# Evaluate the model on the test data
test_loss_rgb, test_accuracy_rgb = model_rgb.evaluate(test_loader_rgb)

# Print test accuracy
print("Test Accuracy:", test_accuracy_rgb)

20/20 [==============================] - 91s 226ms/step - loss: 1.7733
- accuracy: 0.4858
Test Accuracy: 0.4857841432094574



# Define input shape for Depth modality
input_depth = Input(shape=(256, 256, 3), name='input_depth')

# Load the pretrained ResNet50 model for Depth modality
pre_trained_model_depth = ResNet50(weights='imagenet',
include_top=False, input_tensor=input_depth)
```

```
# Remove unnecessary layers from the ResNet50 model
last_layer_depth = pre_trained_model_depth.layers[-1].output

# Additional layers specific to Depth modality
conv2d_depth_1 = Conv2D(128, (3, 3), strides=(1, 1),
activation='relu', padding='same', name='conv2d_depth_1')
(last_layer_depth)
conv2d_depth_2 = Conv2D(256, (3, 3), strides=(1, 1),
activation='relu', padding='same', name='conv2d_depth_2')
(conv2d_depth_1)
dropout_depth = Dropout(0.2, name='dropout_depth')(conv2d_depth_2)

# Transposed convolutional layer
conv2d_transpose = Conv2DTranspose(19, (64, 64), strides=(32, 32),
padding='same', name='conv2d_transpose')(dropout_depth)

# Reshape layer
reshaped_layer = Reshape((256, 256, 19), name='reshape')
(conv2d_transpose)

# Softmax activation layer
activation = Activation('softmax', name='activation')(reshaped_layer)

# Create the model for Depth modality
model_depth = Model(inputs=[input_depth], outputs=activation)

# Print model summary
model_depth.summary()
optimizer = tf.keras.optimizers.legacy.SGD(learning_rate=0.01,
decay=1e-6, momentum=0.9)

# Compile the model
model_depth.compile(optimizer=optimizer,  # Use the defined optimizer
                  loss='categorical_crossentropy',  # Use categorical
cross-entropy loss
                  metrics=['accuracy'])  # Evaluate model performance
using accuracy metric
```

```
Model: "model_3"
______________________________________________________________________
____________________________
 Layer (type)                 Output Shape                 Param #
Connected to
======================================================================
============================
 input_depth (InputLayer)     [(None, 256, 256, 3)]        0         []
 conv1_pad (ZeroPadding2D)   (None, 262, 262, 3)          0
['input_depth[0][0]']


 conv1_conv (Conv2D)         (None, 128, 128, 64)         9472
['conv1_pad[0][0]']


 conv1_bn (BatchNormalizati  (None, 128, 128, 64)         256
['conv1_conv[0][0]']
 on)


 conv1_relu (Activation)     (None, 128, 128, 64)         0
['conv1_bn[0][0]']


 pool1_pad (ZeroPadding2D)   (None, 130, 130, 64)         0
['conv1_relu[0][0]']


 pool1_pool (MaxPooling2D)   (None, 64, 64, 64)           0
['pool1_pad[0][0]']


 conv2_block1_1_conv (Conv2  (None, 64, 64, 64)           4160
['pool1_pool[0][0]']
 D)


 conv2_block1_1_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block1_1_conv[0][0]']
 rmalization)


 conv2_block1_1_relu (Activ  (None, 64, 64, 64)           0
['conv2_block1_1_bn[0][0]']
 ation)


 conv2_block1_2_conv (Conv2  (None, 64, 64, 64)           36928
['conv2_block1_1_relu[0][0]']
 D)
 conv2_block1_2_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block1_2_conv[0][0]']
 rmalization)



 conv2_block1_2_relu (Activ  (None, 64, 64, 64)           0
['conv2_block1_2_bn[0][0]']
 ation)



 conv2_block1_0_conv (Conv2  (None, 64, 64, 256)          16640
['pool1_pool[0][0]']
 D)



 conv2_block1_3_conv (Conv2  (None, 64, 64, 256)          16640
['conv2_block1_2_relu[0][0]']
 D)



 conv2_block1_0_bn (BatchNo  (None, 64, 64, 256)          1024
['conv2_block1_0_conv[0][0]']
 rmalization)



 conv2_block1_3_bn (BatchNo  (None, 64, 64, 256)          1024
['conv2_block1_3_conv[0][0]']
 rmalization)



 conv2_block1_add (Add)      (None, 64, 64, 256)          0
['conv2_block1_0_bn[0][0]',

'conv2_block1_3_bn[0][0]']


 conv2_block1_out (Activati  (None, 64, 64, 256)          0
['conv2_block1_add[0][0]']
 on)



 conv2_block2_1_conv (Conv2  (None, 64, 64, 64)           16448
['conv2_block1_out[0][0]']
 D)
 conv2_block2_1_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block2_1_conv[0][0]']
 rmalization)



 conv2_block2_1_relu (Activ  (None, 64, 64, 64)           0
['conv2_block2_1_bn[0][0]']
 ation)



 conv2_block2_2_conv (Conv2  (None, 64, 64, 64)           36928
['conv2_block2_1_relu[0][0]']
 D)



 conv2_block2_2_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block2_2_conv[0][0]']
 rmalization)



 conv2_block2_2_relu (Activ  (None, 64, 64, 64)           0
['conv2_block2_2_bn[0][0]']
 ation)



 conv2_block2_3_conv (Conv2  (None, 64, 64, 256)          16640
['conv2_block2_2_relu[0][0]']
 D)



 conv2_block2_3_bn (BatchNo  (None, 64, 64, 256)          1024
['conv2_block2_3_conv[0][0]']
 rmalization)



 conv2_block2_add (Add)      (None, 64, 64, 256)          0
['conv2_block1_out[0][0]',

'conv2_block2_3_bn[0][0]']
```

```
 conv2_block2_out (Activati  (None, 64, 64, 256)          0
['conv2_block2_add[0][0]']
 on)



 conv2_block3_1_conv (Conv2  (None, 64, 64, 64)           16448
['conv2_block2_out[0][0]']
 D)



 conv2_block3_1_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block3_1_conv[0][0]']
 rmalization)



 conv2_block3_1_relu (Activ  (None, 64, 64, 64)           0
['conv2_block3_1_bn[0][0]']
 ation)



 conv2_block3_2_conv (Conv2  (None, 64, 64, 64)           36928
['conv2_block3_1_relu[0][0]']
 D)



 conv2_block3_2_bn (BatchNo  (None, 64, 64, 64)           256
['conv2_block3_2_conv[0][0]']
 rmalization)



 conv2_block3_2_relu (Activ  (None, 64, 64, 64)           0
['conv2_block3_2_bn[0][0]']
 ation)



 conv2_block3_3_conv (Conv2  (None, 64, 64, 256)          16640
['conv2_block3_2_relu[0][0]']
 D)



 conv2_block3_3_bn (BatchNo  (None, 64, 64, 256)          1024
['conv2_block3_3_conv[0][0]']
 rmalization)
 conv2_block3_add (Add)      (None, 64, 64, 256)          0
['conv2_block2_out[0][0]',

'conv2_block3_3_bn[0][0]']


 conv2_block3_out (Activati  (None, 64, 64, 256)          0
['conv2_block3_add[0][0]']
 on)



 conv3_block1_1_conv (Conv2  (None, 32, 32, 128)          32896
['conv2_block3_out[0][0]']
 D)



 conv3_block1_1_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block1_1_conv[0][0]']
 rmalization)



 conv3_block1_1_relu (Activ  (None, 32, 32, 128)          0
['conv3_block1_1_bn[0][0]']
 ation)



 conv3_block1_2_conv (Conv2  (None, 32, 32, 128)          147584
['conv3_block1_1_relu[0][0]']
 D)



 conv3_block1_2_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block1_2_conv[0][0]']
 rmalization)



 conv3_block1_2_relu (Activ  (None, 32, 32, 128)          0
['conv3_block1_2_bn[0][0]']
 ation)
 conv3_block1_0_conv (Conv2  (None, 32, 32, 512)          131584
['conv2_block3_out[0][0]']
 D)



 conv3_block1_3_conv (Conv2  (None, 32, 32, 512)          66048
['conv3_block1_2_relu[0][0]']
 D)



 conv3_block1_0_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block1_0_conv[0][0]']
 rmalization)



 conv3_block1_3_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block1_3_conv[0][0]']
 rmalization)



 conv3_block1_add (Add)      (None, 32, 32, 512)          0
['conv3_block1_0_bn[0][0]',

'conv3_block1_3_bn[0][0]']


 conv3_block1_out (Activati  (None, 32, 32, 512)          0
['conv3_block1_add[0][0]']
 on)



 conv3_block2_1_conv (Conv2  (None, 32, 32, 128)          65664
['conv3_block1_out[0][0]']
 D)



 conv3_block2_1_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block2_1_conv[0][0]']
 rmalization)



 conv3_block2_1_relu (Activ  (None, 32, 32, 128)          0
['conv3_block2_1_bn[0][0]']
 ation)
 conv3_block2_2_conv (Conv2  (None, 32, 32, 128)          147584
['conv3_block2_1_relu[0][0]']
 D)



 conv3_block2_2_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block2_2_conv[0][0]']
 rmalization)



 conv3_block2_2_relu (Activ  (None, 32, 32, 128)          0
['conv3_block2_2_bn[0][0]']
 ation)



 conv3_block2_3_conv (Conv2  (None, 32, 32, 512)          66048
['conv3_block2_2_relu[0][0]']
 D)



 conv3_block2_3_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block2_3_conv[0][0]']
 rmalization)



 conv3_block2_add (Add)      (None, 32, 32, 512)          0
['conv3_block1_out[0][0]',

'conv3_block2_3_bn[0][0]']


 conv3_block2_out (Activati  (None, 32, 32, 512)          0
['conv3_block2_add[0][0]']
 on)



 conv3_block3_1_conv (Conv2  (None, 32, 32, 128)          65664
['conv3_block2_out[0][0]']
 D)
 conv3_block3_1_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block3_1_conv[0][0]']
 rmalization)



 conv3_block3_1_relu (Activ  (None, 32, 32, 128)          0
['conv3_block3_1_bn[0][0]']
 ation)



 conv3_block3_2_conv (Conv2  (None, 32, 32, 128)          147584
['conv3_block3_1_relu[0][0]']
 D)



 conv3_block3_2_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block3_2_conv[0][0]']
 rmalization)



 conv3_block3_2_relu (Activ  (None, 32, 32, 128)          0
['conv3_block3_2_bn[0][0]']
 ation)



 conv3_block3_3_conv (Conv2  (None, 32, 32, 512)          66048
['conv3_block3_2_relu[0][0]']
 D)



 conv3_block3_3_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block3_3_conv[0][0]']
 rmalization)



 conv3_block3_add (Add)      (None, 32, 32, 512)          0
['conv3_block2_out[0][0]',

'conv3_block3_3_bn[0][0]']


 conv3_block3_out (Activati  (None, 32, 32, 512)          0
['conv3_block3_add[0][0]']
 on)
 conv3_block4_1_conv (Conv2  (None, 32, 32, 128)          65664
['conv3_block3_out[0][0]']
 D)


 conv3_block4_1_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block4_1_conv[0][0]']
 rmalization)


 conv3_block4_1_relu (Activ  (None, 32, 32, 128)          0
['conv3_block4_1_bn[0][0]']
 ation)


 conv3_block4_2_conv (Conv2  (None, 32, 32, 128)          147584
['conv3_block4_1_relu[0][0]']
 D)


 conv3_block4_2_bn (BatchNo  (None, 32, 32, 128)          512
['conv3_block4_2_conv[0][0]']
 rmalization)


 conv3_block4_2_relu (Activ  (None, 32, 32, 128)          0
['conv3_block4_2_bn[0][0]']
 ation)


 conv3_block4_3_conv (Conv2  (None, 32, 32, 512)          66048
['conv3_block4_2_relu[0][0]']
 D)


 conv3_block4_3_bn (BatchNo  (None, 32, 32, 512)          2048
['conv3_block4_3_conv[0][0]']
 rmalization)
 conv3_block4_add (Add)      (None, 32, 32, 512)          0
['conv3_block3_out[0][0]',

'conv3_block4_3_bn[0][0]']


 conv3_block4_out (Activati  (None, 32, 32, 512)          0
['conv3_block4_add[0][0]']
 on)



 conv4_block1_1_conv (Conv2  (None, 16, 16, 256)          131328
['conv3_block4_out[0][0]']
 D)



 conv4_block1_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block1_1_conv[0][0]']
 rmalization)



 conv4_block1_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block1_1_bn[0][0]']
 ation)



 conv4_block1_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block1_1_relu[0][0]']
 D)



 conv4_block1_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block1_2_conv[0][0]']
 rmalization)



 conv4_block1_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block1_2_bn[0][0]']
 ation)



 conv4_block1_0_conv (Conv2  (None, 16, 16, 1024)         525312
['conv3_block4_out[0][0]']
 D)
 conv4_block1_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block1_2_relu[0][0]']
 D)



 conv4_block1_0_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block1_0_conv[0][0]']
 rmalization)



 conv4_block1_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block1_3_conv[0][0]']
 rmalization)



 conv4_block1_add (Add)      (None, 16, 16, 1024)         0
['conv4_block1_0_bn[0][0]',

'conv4_block1_3_bn[0][0]']


 conv4_block1_out (Activati  (None, 16, 16, 1024)         0
['conv4_block1_add[0][0]']
 on)



 conv4_block2_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block1_out[0][0]']
 D)



 conv4_block2_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block2_1_conv[0][0]']
 rmalization)



 conv4_block2_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block2_1_bn[0][0]']
 ation)
 conv4_block2_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block2_1_relu[0][0]']
 D)



 conv4_block2_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block2_2_conv[0][0]']
 rmalization)



 conv4_block2_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block2_2_bn[0][0]']
 ation)



 conv4_block2_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block2_2_relu[0][0]']
 D)



 conv4_block2_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block2_3_conv[0][0]']
 rmalization)



 conv4_block2_add (Add)      (None, 16, 16, 1024)         0
['conv4_block1_out[0][0]',

'conv4_block2_3_bn[0][0]']


 conv4_block2_out (Activati  (None, 16, 16, 1024)         0
['conv4_block2_add[0][0]']
 on)



 conv4_block3_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block2_out[0][0]']
 D)



 conv4_block3_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block3_1_conv[0][0]']
 rmalization)
 conv4_block3_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block3_1_bn[0][0]']
 ation)



 conv4_block3_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block3_1_relu[0][0]']
 D)



 conv4_block3_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block3_2_conv[0][0]']
 rmalization)



 conv4_block3_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block3_2_bn[0][0]']
 ation)



 conv4_block3_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block3_2_relu[0][0]']
 D)



 conv4_block3_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block3_3_conv[0][0]']
 rmalization)



 conv4_block3_add (Add)      (None, 16, 16, 1024)         0
['conv4_block2_out[0][0]',

'conv4_block3_3_bn[0][0]']


 conv4_block3_out (Activati  (None, 16, 16, 1024)         0
['conv4_block3_add[0][0]']
 on)
 conv4_block4_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block3_out[0][0]']
 D)



 conv4_block4_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block4_1_conv[0][0]']
 rmalization)



 conv4_block4_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block4_1_bn[0][0]']
 ation)



 conv4_block4_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block4_1_relu[0][0]']
 D)



 conv4_block4_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block4_2_conv[0][0]']
 rmalization)



 conv4_block4_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block4_2_bn[0][0]']
 ation)



 conv4_block4_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block4_2_relu[0][0]']
 D)



 conv4_block4_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block4_3_conv[0][0]']
 rmalization)



 conv4_block4_add (Add)      (None, 16, 16, 1024)         0
['conv4_block3_out[0][0]',
'conv4_block4_3_bn[0][0]']


 conv4_block4_out (Activati  (None, 16, 16, 1024)         0
['conv4_block4_add[0][0]']
 on)



 conv4_block5_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block4_out[0][0]']
 D)



 conv4_block5_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block5_1_conv[0][0]']
 rmalization)



 conv4_block5_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block5_1_bn[0][0]']
 ation)



 conv4_block5_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block5_1_relu[0][0]']
 D)



 conv4_block5_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block5_2_conv[0][0]']
 rmalization)



 conv4_block5_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block5_2_bn[0][0]']
 ation)



 conv4_block5_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block5_2_relu[0][0]']
 D)
 conv4_block5_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block5_3_conv[0][0]']
 rmalization)



 conv4_block5_add (Add)      (None, 16, 16, 1024)         0
['conv4_block4_out[0][0]',

'conv4_block5_3_bn[0][0]']


 conv4_block5_out (Activati  (None, 16, 16, 1024)         0
['conv4_block5_add[0][0]']
 on)



 conv4_block6_1_conv (Conv2  (None, 16, 16, 256)          262400
['conv4_block5_out[0][0]']
 D)



 conv4_block6_1_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block6_1_conv[0][0]']
 rmalization)



 conv4_block6_1_relu (Activ  (None, 16, 16, 256)          0
['conv4_block6_1_bn[0][0]']
 ation)



 conv4_block6_2_conv (Conv2  (None, 16, 16, 256)          590080
['conv4_block6_1_relu[0][0]']
 D)



 conv4_block6_2_bn (BatchNo  (None, 16, 16, 256)          1024
['conv4_block6_2_conv[0][0]']
 rmalization)



 conv4_block6_2_relu (Activ  (None, 16, 16, 256)          0
['conv4_block6_2_bn[0][0]']
 ation)
 conv4_block6_3_conv (Conv2  (None, 16, 16, 1024)         263168
['conv4_block6_2_relu[0][0]']
 D)



 conv4_block6_3_bn (BatchNo  (None, 16, 16, 1024)         4096
['conv4_block6_3_conv[0][0]']
 rmalization)



 conv4_block6_add (Add)      (None, 16, 16, 1024)         0
['conv4_block5_out[0][0]',

'conv4_block6_3_bn[0][0]']


 conv4_block6_out (Activati  (None, 16, 16, 1024)         0
['conv4_block6_add[0][0]']
 on)



 conv5_block1_1_conv (Conv2  (None, 8, 8, 512)            524800
['conv4_block6_out[0][0]']
 D)



 conv5_block1_1_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block1_1_conv[0][0]']
 rmalization)



 conv5_block1_1_relu (Activ  (None, 8, 8, 512)            0
['conv5_block1_1_bn[0][0]']
 ation)



 conv5_block1_2_conv (Conv2  (None, 8, 8, 512)            2359808
['conv5_block1_1_relu[0][0]']
 D)
 conv5_block1_2_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block1_2_conv[0][0]']
 rmalization)



 conv5_block1_2_relu (Activ  (None, 8, 8, 512)            0
['conv5_block1_2_bn[0][0]']
 ation)



 conv5_block1_0_conv (Conv2  (None, 8, 8, 2048)           2099200
['conv4_block6_out[0][0]']
 D)



 conv5_block1_3_conv (Conv2  (None, 8, 8, 2048)           1050624
['conv5_block1_2_relu[0][0]']
 D)



 conv5_block1_0_bn (BatchNo  (None, 8, 8, 2048)           8192
['conv5_block1_0_conv[0][0]']
 rmalization)



 conv5_block1_3_bn (BatchNo  (None, 8, 8, 2048)           8192
['conv5_block1_3_conv[0][0]']
 rmalization)



 conv5_block1_add (Add)      (None, 8, 8, 2048)           0
['conv5_block1_0_bn[0][0]',

'conv5_block1_3_bn[0][0]']


 conv5_block1_out (Activati  (None, 8, 8, 2048)           0
['conv5_block1_add[0][0]']
 on)



 conv5_block2_1_conv (Conv2  (None, 8, 8, 512)            1049088
['conv5_block1_out[0][0]']
 D)
 conv5_block2_1_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block2_1_conv[0][0]']
 rmalization)


 conv5_block2_1_relu (Activ  (None, 8, 8, 512)            0
['conv5_block2_1_bn[0][0]']
 ation)


 conv5_block2_2_conv (Conv2  (None, 8, 8, 512)            2359808
['conv5_block2_1_relu[0][0]']
 D)


 conv5_block2_2_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block2_2_conv[0][0]']
 rmalization)


 conv5_block2_2_relu (Activ  (None, 8, 8, 512)            0
['conv5_block2_2_bn[0][0]']
 ation)


 conv5_block2_3_conv (Conv2  (None, 8, 8, 2048)           1050624
['conv5_block2_2_relu[0][0]']
 D)


 conv5_block2_3_bn (BatchNo  (None, 8, 8, 2048)           8192
['conv5_block2_3_conv[0][0]']
 rmalization)


 conv5_block2_add (Add)      (None, 8, 8, 2048)           0
['conv5_block1_out[0][0]',

'conv5_block2_3_bn[0][0]']
```

```
 conv5_block2_out (Activati  (None, 8, 8, 2048)           0
['conv5_block2_add[0][0]']
 on)



 conv5_block3_1_conv (Conv2  (None, 8, 8, 512)            1049088
['conv5_block2_out[0][0]']
 D)



 conv5_block3_1_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block3_1_conv[0][0]']
 rmalization)



 conv5_block3_1_relu (Activ  (None, 8, 8, 512)            0
['conv5_block3_1_bn[0][0]']
 ation)



 conv5_block3_2_conv (Conv2  (None, 8, 8, 512)            2359808
['conv5_block3_1_relu[0][0]']
 D)



 conv5_block3_2_bn (BatchNo  (None, 8, 8, 512)            2048
['conv5_block3_2_conv[0][0]']
 rmalization)



 conv5_block3_2_relu (Activ  (None, 8, 8, 512)            0
['conv5_block3_2_bn[0][0]']
 ation)



 conv5_block3_3_conv (Conv2  (None, 8, 8, 2048)           1050624
['conv5_block3_2_relu[0][0]']
 D)



 conv5_block3_3_bn (BatchNo  (None, 8, 8, 2048)           8192
['conv5_block3_3_conv[0][0]']
 rmalization)
 conv5_block3_add (Add)      (None, 8, 8, 2048)           0
['conv5_block2_out[0][0]',

'conv5_block3_3_bn[0][0]']


 conv5_block3_out (Activati  (None, 8, 8, 2048)           0
['conv5_block3_add[0][0]']
 on)



 conv2d_depth_1 (Conv2D)     (None, 8, 8, 128)            2359424
['conv5_block3_out[0][0]']


 conv2d_depth_2 (Conv2D)     (None, 8, 8, 256)            295168
['conv2d_depth_1[0][0]']


 dropout_depth (Dropout)     (None, 8, 8, 256)            0
['conv2d_depth_2[0][0]']


 conv2d_transpose (Conv2DTr  (None, 256, 256, 19)         1992296
['dropout_depth[0][0]']
 anspose)                                                 3



 reshape (Reshape)           (None, 256, 256, 19)         0
['conv2d_transpose[0][0]']


 activation (Activation)     (None, 256, 256, 19)         0
['reshape[0][0]']


======================================================================
============================
Total params: 46165267 (176.11 MB)
Trainable params: 46112147 (175.90 MB)
Non-trainable params: 53120 (207.50 KB)
______________________________________________________________________
____________________________

def resize_labels(labels, target_shape):
    resized_labels = tf.image.resize(labels, target_shape[:2])
    return resized_labels

height = 256
width = 256
num_classes = 19  # there are 19 classes
target_shape = (height, width)

# Resize train labels
resized_train_labels = resize_labels(train_loader_depth.labels,
target_shape=target_shape + (num_classes,))

# Resize test labels
resized_test_labels = resize_labels(test_loader_depth.labels,
target_shape=target_shape + (num_classes,))
print(resized_test_labels.shape)

# Resize validation labels
resized_val_labels = resize_labels(val_loader_depth.labels,
target_shape=target_shape + (num_classes,))




# Define the number of epochs
epochs = 10

# Train the model using the fit method
history = model_depth.fit(train_loader_depth, epochs=epochs,
validation_data=val_loader_depth)

# Evaluate the model on the test data
test_loss_depth, test_accuracy_depth =
model_depth.evaluate(test_loader_depth)

# Print test accuracy
print("Test Accuracy:", test_accuracy_depth)

Epoch 1/10
50/50 [==============================] - 25s 364ms/step - loss: 2.5400
- accuracy: 0.3023 - val_loss: 2.2961 - val_accuracy: 0.3330
Epoch 2/10
50/50 [==============================] - 18s 350ms/step - loss: 1.7355
- accuracy: 0.4111 - val_loss: 2.1043 - val_accuracy: 0.4415
Epoch 3/10
50/50 [==============================] - 18s 366ms/step - loss: 1.5280
- accuracy: 0.5402 - val_loss: 2.1774 - val_accuracy: 0.1897
Epoch 4/10
50/50 [==============================] - 18s 349ms/step - loss: 1.3248
- accuracy: 0.5596 - val_loss: 2.4294 - val_accuracy: 0.1915
Epoch 5/10
50/50 [==============================] - 18s 350ms/step - loss: 1.1968
```

```
- accuracy: 0.5787 - val_loss: 2.7164 - val_accuracy: 0.1981
Epoch 6/10
50/50 [==============================] - 19s 370ms/step - loss: 1.1299
- accuracy: 0.6022 - val_loss: 2.7773 - val_accuracy: 0.1964
Epoch 7/10
50/50 [==============================] - 17s 345ms/step - loss: 1.0730
- accuracy: 0.6257 - val_loss: 2.9384 - val_accuracy: 0.1950
Epoch 8/10
50/50 [==============================] - 18s 348ms/step - loss: 1.0120
- accuracy: 0.6576 - val_loss: 3.4144 - val_accuracy: 0.1839
Epoch 9/10
50/50 [==============================] - 18s 366ms/step - loss: 0.9548
- accuracy: 0.6901 - val_loss: 3.6499 - val_accuracy: 0.1616
Epoch 10/10
50/50 [==============================] - 18s 349ms/step - loss: 0.9039
- accuracy: 0.7161 - val_loss: 3.5068 - val_accuracy: 0.1614
20/20 [==============================] - 4s 203ms/step - loss: 3.6844
- accuracy: 0.1534
Test Accuracy: 0.15336501598358154
```

RESULT

Modality	Test Accuracy(%)
RGB only	
Depth Only	
RBB and Depth Fusion	

```
# Define the test accuracies for each modality


# Print the table
print("| Modality   | Test Accuracy (%) |")
print("|-----------|-------------------|")
print(f"| RGB       |       {test_accuracy_rgb}        |")
print(f"| Depth     |       {test_accuracy_depth}        |")
print(f"| RGB+Depth |       {test_accuracy_fusion}        |")

| Modality  | Test Accuracy (%) |
|-----------|-------------------|
| RGB       |       0.4857841432094574        |
| Depth     |       0.15336501598358154        |
| RGB+Depth |       0.41075384616851807        |
```

The results indicate that the RGB modality achieved the highest test accuracy of approximately 48.6%, followed by the RGB+Depth modality with around 41.1% accuracy, and the Depth modality with the lowest accuracy of about 15.3%.

The higher accuracy of the RGB modality compared to Depth alone suggests that RGB images contain more discriminative information for the segmentation task. The combination of RGB and Depth modalities in RGB+Depth likely provides complementary information, leading to a moderate improvement in accuracy compared to Depth alone. However, Depth alone might struggle due to its limited discriminative features for the segmentation task, resulting in the lowest accuracy

I also attempted to freeze the pretrained model's layers in rgb only and depth only by setting trainable = False causing the rgb and depth models' accuracy appear to increase to 56%. Like before, since I'm taking use of the pretrained layers' feature extraction powers by freezing them, the subsequent layers aren't able to distinguish between various inputs well if they are segmented, but might yield consistent results and higher ones when compared to trainable pretrained layers.