1. Data preprocessing

In [1]:

```
import imageio
import random
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import glob
import cv2
import tensorflow as tf
import tensorflow.keras.layers as tfl

from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.layers import Activation, Input, Conv2D, MaxPooli
ng2D, BatchNormalization, Conv2DTranspose, concatenate, Concatenate, Up
Sampling2D, AveragePooling2D
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.applications import ResNet50
from sklearn.model_selection import train_test_split
import tensorflow.keras as keras


import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

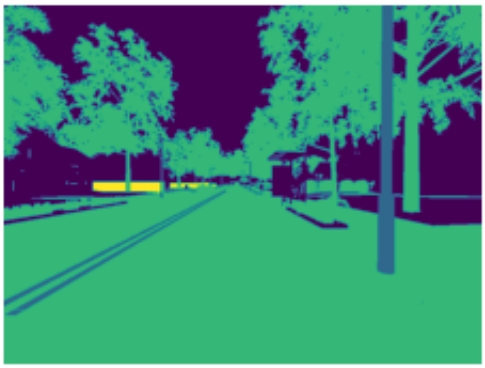
In [2]:

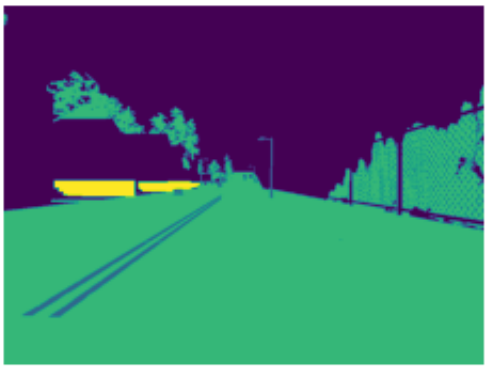
```
images_list = []
masks_list = []
def store_images(data):
    image = glob.glob('/kaggle/input/lyft-udacity-challenge/'+data+'/'+
data+'/CameraRGB/*.png')
    mask = glob.glob('/kaggle/input/lyft-udacity-challenge/'+data+'/'+d
ata+'/CameraSeg/*.png')
    images_list.extend(image)
    masks_list.extend(mask)
for data in ['dataA', 'dataB', 'dataC', 'dataD', 'dataE']:
    store_images(data)

dataset = tf.data.Dataset.from_tensor_slices((images_list, masks_list))
dataset_length = sum(1 for _ in dataset)

print("Length of dataset:", dataset_length)
# for image, mask in dataset.take(1):
#     print("Image File:", image.numpy().decode(), "Mask File:", mask.nu
mpy().decode())
```

```
Length of dataset: 5000
```

In [3]:

```
def show_images(data, num):
    # load images
    images = glob.glob('/kaggle/input/lyft-udacity-challenge/'+data+'/'
+data+'/CameraRGB/*.png')[:num]
    masks = glob.glob('/kaggle/input/lyft-udacity-challenge/'+data+'/'+
data+'/CameraSeg/*.png')[:num]

    for i in range(len(images)):
        images[i] = cv2.imread(images[i])
        masks[i] = cv2.imread(masks[i], cv2.IMREAD_GRAYSCALE)

    fig, axes = plt.subplots(num, 2)

    for i in range(len(images)):
        axes[i][0].imshow(cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB))
        axes[i][1].imshow(masks[i])
        axes[i][0].axis('off')
        axes[i][1].axis('off')
    plt.show()

show_images('dataB', 2)
```





In [4]:
```
def load_datas(image_path, mask_path):
    image = tf.io.read_file(image_path)
    image = tf.image.decode_png(image, channels=3)
    image = tf.image.convert_image_dtype(image, tf.float32)

    mask = tf.io.read_file(mask_path)
    mask = tf.image.decode_png(mask, channels=3)
    mask = tf.math.reduce_max(mask, axis=-1, keepdims=True)
    return image, mask

def preprocess(image, mask, height=256, width=256):
    input_image = tf.image.resize(image, (height, width), method='neare
st')
    input_mask = tf.image.resize(mask, (height, width), method='nearest
')

    return input_image, input_mask

dataset_load = dataset.map(load_datas)
dataset_processed = dataset_load.map(preprocess)
```

In [5]:
```
EPOCHS = 15
BATCH_SIZE = 32
```

In [6]:
```
def split_dataset(data, train_split=0.7, val_split=0.15, test_split=0.1
5, shuffle=True, shuffle_size=1000, batch_size=BATCH_SIZE):
    """
    Splits the dataset into training, validation, and test sets.
    Returns: (train_dataset, val_dataset, test_dataset).
    """

    assert train_split + val_split + test_split == 1, "Split ratios mus
t sum to 1"
```

```
    # Determine the size of the dataset
    dataset_size = len(list(data))

    # Calculate split sizes
    train_size = int(train_split * dataset_size)
    val_size = int(val_split * dataset_size)

    # Shuffle the dataset
    if shuffle:
        data = data.shuffle(shuffle_size, reshuffle_each_iteration=Fals
e)

    # Split the dataset
    train_dataset = data.take(train_size)
    val_dataset = data.skip(train_size).take(val_size)
    test_dataset = data.skip(train_size + val_size)

    # Batch the datasets
    train_dataset = train_dataset.batch(batch_size)
    val_dataset = val_dataset.batch(batch_size)
    test_dataset = test_dataset.batch(batch_size)

    return train_dataset, val_dataset, test_dataset

train_dataset, val_dataset, test_dataset = split_dataset(dataset_proces
sed)
```

2. Model Architectures

In this project, 3 model architectures are built, trained and compared, to see each performance on semantic segmentation tasks for autonomous driving.

1. Fully Convolutional Network (FCN)

FCN is a deep learning architecture for semantic segmentation tasks, retaining spatial information with convolutional layers instead of fully connected layers.

2. U-Net

U-Net is a CNN architecture initally used for biomedical image segmentation, featuring a contracting and expansive path with skip connections for precise localization.

3. DeepLabV3

DeepLabV3, developed by Google Research, is a cutting-edge architecture for semantic image segmentation tasks.

- ### Model architecture for FCN

In [7]:
```
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization,
ReLU, Conv2DTranspose, concatenate, Activation

def conv_block_3(inputs=None, n_filters=32, dropout_prob=0):
    layer = Conv2D(n_filters, 3, padding='same', kernel_initializer='he
_normal')(inputs)
    layer = BatchNormalization(axis=3)(layer)
    layer = Activation("relu")(layer)
    layer = Conv2D(n_filters, 3, padding='same', kernel_initializer='he
_normal')(layer)
    layer = BatchNormalization(axis=3)(layer)
    layer = Activation("relu")(layer)
```

```
    if dropout_prob > 0:
        layer = tf.keras.layers.Dropout(dropout_prob)(layer)

    return layer

def decoder_block(input_tensor, concat_tensor, n_filters):
    # Upsample the input
    decoder = Conv2DTranspose(n_filters, (3, 3), strides=(1, 1), paddin
g='same')(input_tensor)
    # Concatenate with the corresponding layer from the encoder
    decoder = concatenate([decoder, concat_tensor], axis=-1)
    # Additional convolutional layers
    decoder = Conv2D(n_filters, (3, 3), padding='same')(decoder)
    decoder = BatchNormalization()(decoder)
    decoder = Activation('relu')(decoder)

    return decoder

def fcn_model(input_size=(256, 256, 3), n_filters=32, n_classes=13):
    inputs = Input(input_size)

    # Encoder
    cblock1 = conv_block_3(inputs, n_filters) # (None, 256, 256, 32)
    cblock2 = conv_block_3(cblock1, n_filters * 2) # (None, 256, 256, 6
4)
    cblock3 = conv_block_3(cblock2, n_filters * 4) # (None, 256, 256, 1
28)

    # Decoder
    dblock1 = decoder_block(cblock3, cblock2, n_filters * 2) # (None, 2
56, 256, 64)
    dblock2 = decoder_block(dblock1, cblock1, n_filters) # (None, 256,
256, 32)

    # Output layer: (None, 256, 256, 13)
    layer = Conv2D(n_classes, 3, padding='same', kernel_initializer='he
```

```
_normal')(cblock3)
    layer = BatchNormalization(axis=3)(layer)

    model = tf.keras.Model(inputs=inputs, outputs=layer)
    model.compile(optimizer=tf.keras.optimizers.Adam(),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(fr
om_logits=True),
                  metrics=['accuracy'])

    return model
```

- ### Model architecture for U-Net

In [8]:

```
def conv_layer(inputs, n_filters, kernel_size=3):
    """
    Apply convolution, batch normalization, and ReLU activation.

    Args:
    - inputs: Input tensor.
    - n_filters: Number of filters for the convolutional layer.
    - kernel_size: Size of the convolution kernel.

    Returns:
    - Output tensor after the operations.
    """
    layer = tfl.Conv2D(n_filters, kernel_size=kernel_size, padding='sam
e', kernel_initializer='he_normal')(inputs)
    layer = tfl.BatchNormalization(axis=3)(layer)
    layer = tfl.ReLU()(layer)
    return layer
```

In [9]:

```
def conv_block(inputs, n_filters, dropout_prob=0, max_pooling=True):
    """
    Perform two convolutions with optional dropout and max pooling.

    Args:
    - inputs: Input tensor.
    - n_filters: Number of filters for the convolutional layers.
    - dropout_prob: Dropout rate.
    - max_pooling: Boolean, whether to include a max pooling layer.

    Returns:
    - next_layer: Output tensor for the next layer.
    - skip_connection: Output tensor for the skip connection.
    """
    # Two convolutional layers
    layer = conv_layer(inputs, n_filters)
    layer = conv_layer(layer, n_filters)

    # Optional dropout
    if dropout_prob > 0:
        layer = tfl.Dropout(dropout_prob)(layer)
    # Optional max pooling
    if max_pooling:
        next_layer = tfl.MaxPooling2D(pool_size=(2, 2))(layer)
    else:
        next_layer = layer

    skip_connection = layer

    return next_layer, skip_connection
```

In [10]:

```
def upsampling_block(inputs, skip_connection_inputs, n_filters):
    """
    Upsample the input and merge with the skip connection.

    Args:
    - inputs: Input tensor from the previous layer.
    - skip_connection_inputs: Input tensor from the corresponding contra
ction block (for the skip connection).
    - n_filters: Number of filters for the convolutional layers.

    Returns:
    - layer: Output tensor after upsampling and convolution.
    """
    # Upsampling
    up = tfl.Conv2DTranspose(n_filters, kernel_size=3, strides=(2, 2),
padding='same')(inputs)
    # Merging with skip connection
    merge = tfl.concatenate([up, skip_connection_inputs], axis=3)
    # Two convolutional layers
    layer = conv_layer(merge, n_filters)
    layer = conv_layer(layer, n_filters)

    return layer
```

In [11]:

```
# U-Net model
def unet_model(input_size=(256, 256, 3), n_filters=32, n_classes=13):
    """
    Define the U-Net model architecture.

    Args:
    - input_size: Shape of the input images.
    - n_filters: Number of filters for the convolutional layers in the f
irst block. Gets doubled in each subsequent block.
    - n_classes: Number of output classes.
```

```
    Returns:
    - model: Compiled U-Net model.
    """
    inputs = tfl.Input(input_size)

    # Encoding path
    cblock1 = conv_block(inputs, n_filters)
    cblock2 = conv_block(cblock1[0], n_filters * 2)
    cblock3 = conv_block(cblock2[0], n_filters * 4)
    cblock4 = conv_block(cblock3[0], n_filters * 8, dropout_prob=0.3)
    cblock5 = conv_block(cblock4[0], n_filters * 16, dropout_prob=0.3,
max_pooling=False)

    # Decoding path
    ublock6 = upsampling_block(cblock5[0], cblock4[1], n_filters * 8)
    ublock7 = upsampling_block(ublock6, cblock3[1], n_filters * 4)
    ublock8 = upsampling_block(ublock7, cblock2[1], n_filters * 2)
    ublock9 = upsampling_block(ublock8, cblock1[1], n_filters)

    # Output layer
    output_layer = conv_layer(ublock9, n_filters)
    output_layer = tfl.Conv2D(n_classes, kernel_size=1, padding='same')
(output_layer)

    model = tf.keras.Model(inputs=inputs, outputs=output_layer)
    model.compile(optimizer=tf.keras.optimizers.Adam(),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_l
ogits=True),
                  metrics=['accuracy'])

    return model
```

- ### Model architecture for DeepLabV3

In [12]:

```
def aspp(input):
    shape = input.shape
    y_pool = AveragePooling2D(pool_size = (shape[1], shape[2]))(input)
    y_pool = Conv2D(filters=256, padding='same', use_bias=False, kernel
_size=1)(y_pool)
    y_pool = BatchNormalization()(y_pool)
    y_pool = Activation("relu")(y_pool)
    y_pool = UpSampling2D((shape[1], shape[2]), interpolation="bilinear
")(y_pool)

    out_1 = Conv2D(filters=256, padding='same', dilation_rate = 1, use_
bias=False, kernel_size=1)(input)
    out_1 = BatchNormalization()(out_1)
    out_1 = Activation("relu")(out_1)

    out_6 = Conv2D(filters=256, padding='same', dilation_rate = 6, use_
bias=False, kernel_size=1)(input)
    out_6 = BatchNormalization()(out_6)
    out_6 = Activation("relu")(out_6)

    out_12 = Conv2D(filters=256, padding='same', dilation_rate = 12, us
e_bias=False, kernel_size=1)(input)
    out_12 = BatchNormalization()(out_12)
    out_12 = Activation("relu")(out_12)

    out_18 = Conv2D(filters=256, padding='same', dilation_rate = 18, us
e_bias=False, kernel_size=1)(input)
    out_18 = BatchNormalization()(out_18)
    out_18 = Activation("relu")(out_18)

    y = Concatenate()([y_pool, out_1, out_6, out_12, out_18])
    y = Conv2D(filters=256, padding='same', dilation_rate = 1, use_bias
=False, kernel_size=1)(y)
    y = BatchNormalization()(y)
    y = Activation("relu")(y)
```

```
    return y

def DeeplabV3(image_size=(256, 256, 3), num_classes=13):

    inputs = Input(image_size)

    resnet50 = keras.applications.ResNet50(
        weights="imagenet", include_top=False, input_tensor=inputs
    )

    # pretrained Resnet50 output
#     x1 = resnet50.get_layer("conv4_block6_out").output
    x1 = resnet50.get_layer("conv4_block6_2_relu").output
    x1 = aspp(x1)
    x1 = UpSampling2D((4, 4), interpolation="bilinear")(x1)

    # low level features
#     x2 = resnet50.get_layer("conv2_block2_out").output
    x2 = resnet50.get_layer("conv2_block3_2_relu").output
    x2 = Conv2D(filters=48, padding='same', use_bias=False, kernel_size
=1)(x2)
    x2 = BatchNormalization()(x2)
    x2 = Activation("relu")(x2)

    x = Concatenate()([x1, x2])

    x = Conv2D(filters=256, padding='same', activation='relu', use_bias
=False, kernel_size=3)(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)

    x = Conv2D(filters=256, padding='same', activation='relu', use_bias
=False, kernel_size=3)(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)

    x = UpSampling2D((4, 4), interpolation="bilinear")(x)
```

```
    # output
    x = Conv2D(num_classes, kernel_size=(1, 1), name="output_layer", pa
dding="same")(x)
    x = Activation('sigmoid')(x)

    model = Model(inputs=inputs, outputs=x)

    model.compile(optimizer=tf.keras.optimizers.Adam(),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_l
ogits=True),
                  metrics=['accuracy'])

    return model
```

3. Model Training

Hyperparameter toning and model configerations.

In [13]:

```
img_height = 256
img_width = 256
num_channels = 3
filters = 32
n_classes = 13
```

Function to plot training history

In [14]:

```
def plot_training_history(history):
    acc = [0.] + history.history.get('accuracy', [])
    val_acc = [0.] + history.history.get('val_accuracy', [])
    loss = history.history.get('loss', [])
    val_loss = history.history.get('val_loss', [])

    # Plotting
    plt.figure(figsize=(8, 8))

    # Accuracy subplot
    plt.subplot(2, 1, 1)
    plt.plot(acc, label='Training Accuracy')
    plt.plot(val_acc, label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.ylabel('Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.ylim([0, 1])

    # Loss subplot
    plt.subplot(2, 1, 2)
    plt.plot(loss, label='Training Loss')
    plt.plot(val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.ylabel('Cross Entropy')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylim([0, 1])

    plt.tight_layout()
    plt.show()
```

- ### Training FCN

In [15]:
```
import time

# FCN model

fcn_model1 = fcn_model()
fcn_model1.summary()
```

Model: "functional_1"

Layer (type)	Output Shape	Param
input_layer (InputLayer)	(None, 256, 256, 3)	
conv2d (Conv2D)	(None, 256, 256, 32)	89
batch_normalization (BatchNormalization)	(None, 256, 256, 32)	12
activation (Activation)	(None, 256, 256, 32)	
conv2d_1 (Conv2D)	(None, 256, 256, 32)	9,24
batch_normalization_1 (BatchNormalization)	(None, 256, 256, 32)	12
activation_1 (Activation)	(None, 256, 256, 32)	
conv2d_2 (Conv2D)	(None, 256, 256, 64)	18,49
batch_normalization_2 (BatchNormalization)	(None, 256, 256, 64)	25
activation_2 (Activation)	(None, 256, 256, 64)	
conv2d_3 (Conv2D)	(None, 256, 256, 64)	36,92
batch_normalization_3 (BatchNormalization)	(None, 256, 256, 64)	25
activation_3 (Activation)	(None, 256, 256, 64)	
conv2d_4 (Conv2D)	(None, 256, 256, 128)	73,85

batch_normalization_4 (BatchNormalization)	(None, 256, 256, 128)	51
activation_4 (Activation)	(None, 256, 256, 128)	
conv2d_5 (Conv2D)	(None, 256, 256, 128)	147,58
batch_normalization_5 (BatchNormalization)	(None, 256, 256, 128)	51
activation_5 (Activation)	(None, 256, 256, 128)	
conv2d_8 (Conv2D)	(None, 256, 256, 13)	14,98
batch_normalization_8 (BatchNormalization)	(None, 256, 256, 13)	5

Total params: 303,841 (1.16 MB)

Trainable params: 302,919 (1.16 MB)

Non-trainable params: 922 (3.60 KB)

In [16]:

```
start_time = time.time()

# reduce_lr = ReduceLROnPlateau(monitor= "loss", factor=0.1,
#                               patience= 1, min_lr= 1e-6)
# early_stop = EarlyStopping(patience= 2)

reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,
                              patience=10, min_lr=1e-6)
early_stop = EarlyStopping(monitor='val_accuracy', patience=5, restore_
best_weights=True)

fcn_result1 = fcn_model1.fit(train_dataset,
                    validation_data=val_dataset,
                    epochs=EPOCHS,
                    callbacks=[reduce_lr,early_stop],
                    batch_size=BATCH_SIZE)
end_time = time.time()
training_time = end_time - start_time
```

```
Epoch 1/15


WARNING: All log messages before absl::InitializeLog() is called are wr
itten to STDERR
I0000 00:00:1713213812.688247      68 device_compiler.h:186] Compiled c
luster using XLA!  This line is logged at most once for the lifetime of
the process.


110/110 ━━━━━━━━━━━━━━━━━━━━ 220s 1s/step - accuracy: 0.6441 - loss:
1.5717 - val_accuracy: 0.3936 - val_loss: 2.2648 - learning_rate: 0.001
0
Epoch 2/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 94s 778ms/step - accuracy: 0.8108 - loss
: 1.0322 - val_accuracy: 0.3834 - val_loss: 2.4630 - learning_rate: 0.0
010
```

```
Epoch 3/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 94s 780ms/step - accuracy: 0.8382 - loss
: 0.8588 - val_accuracy: 0.5838 - val_loss: 1.6576 - learning_rate: 0.0
010
Epoch 4/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 94s 777ms/step - accuracy: 0.8538 - loss
: 0.7404 - val_accuracy: 0.7593 - val_loss: 1.0123 - learning_rate: 0.0
010
Epoch 5/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 93s 778ms/step - accuracy: 0.8637 - loss
: 0.6558 - val_accuracy: 0.8455 - val_loss: 0.6868 - learning_rate: 0.0
010
Epoch 6/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 93s 778ms/step - accuracy: 0.8738 - loss
: 0.5846 - val_accuracy: 0.8668 - val_loss: 0.5881 - learning_rate: 0.0
010
Epoch 7/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 93s 778ms/step - accuracy: 0.8802 - loss
: 0.5330 - val_accuracy: 0.8791 - val_loss: 0.5168 - learning_rate: 0.0
010
Epoch 8/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 93s 779ms/step - accuracy: 0.8858 - loss
: 0.4913 - val_accuracy: 0.8831 - val_loss: 0.4882 - learning_rate: 0.0
010
Epoch 9/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 93s 777ms/step - accuracy: 0.8907 - loss
: 0.4567 - val_accuracy: 0.8751 - val_loss: 0.4884 - learning_rate: 0.0
010
Epoch 10/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 93s 778ms/step - accuracy: 0.8946 - loss
: 0.4286 - val_accuracy: 0.8782 - val_loss: 0.4606 - learning_rate: 0.0
010
Epoch 11/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 94s 786ms/step - accuracy: 0.8982 - loss
: 0.4043 - val_accuracy: 0.8862 - val_loss: 0.4266 - learning_rate: 0.0
010
Epoch 12/15
```

```
110/110 ━━━━━━━━━━━━━━━━━━━━ 94s 782ms/step - accuracy: 0.9011 - loss
: 0.3847 - val_accuracy: 0.8853 - val_loss: 0.4205 - learning_rate: 0.0
010
Epoch 13/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 94s 779ms/step - accuracy: 0.9036 - loss
: 0.3678 - val_accuracy: 0.8865 - val_loss: 0.4068 - learning_rate: 0.0
010
Epoch 14/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 93s 778ms/step - accuracy: 0.9061 - loss
: 0.3526 - val_accuracy: 0.8967 - val_loss: 0.3737 - learning_rate: 0.0
010
Epoch 15/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 93s 772ms/step - accuracy: 0.9084 - loss
: 0.3390 - val_accuracy: 0.9058 - val_loss: 0.3440 - learning_rate: 0.0
010
```

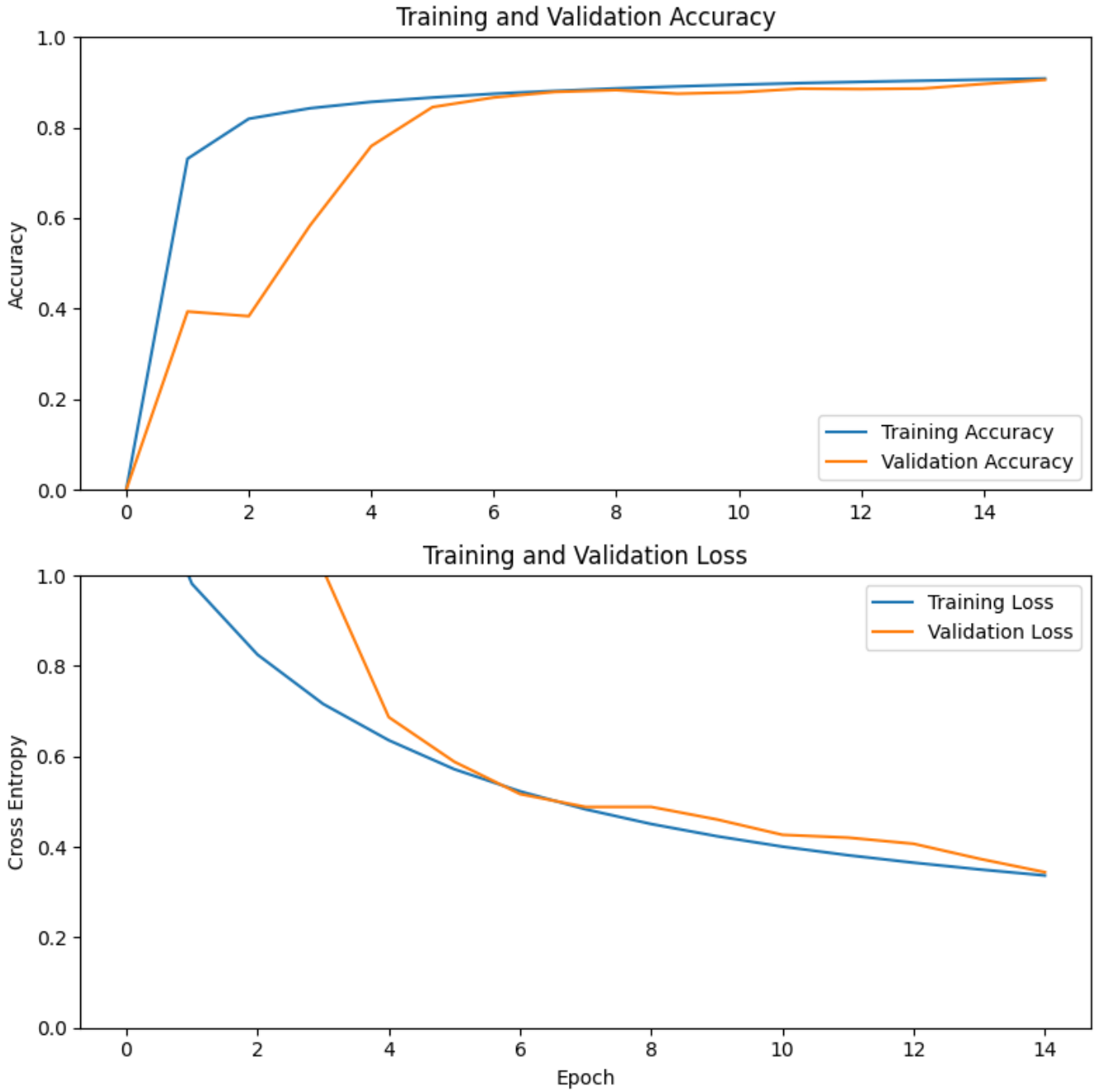
In [17]:

```
print("-----------------------------------------------------------------
")
print("Training time for FCN:", training_time, "seconds")
```

```
----------------------------------------------------------------
Training time for FCN: 1530.48957157135 seconds
```

In [18]:

```
plot_training_history(fcn_result1)
```



- ### Training U-Net

In [19]:

```
# U-net model
unet_model1 = unet_model()
unet_model1.summary()
```

Model: "functional_3"

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 256, 256, 3)	0	–
conv2d_9 (Conv2D)	(None, 256, 256, 32)	896	input_layer_1[0]
batch_normalizatio… (BatchNormalizatio…	(None, 256, 256, 32)	128	conv2d_9[0][0]
re_lu (ReLU)	(None, 256, 256, 32)	0	batch_normalizat
conv2d_10 (Conv2D)	(None, 256, 256, 32)	9,248	re_lu[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 256, 256, 32)	128	conv2d_10[0][0]
re_lu_1 (ReLU)	(None, 256, 256, 32)	0	batch_normalizat
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0	re_lu_1[0][0]
conv2d_11 (Conv2D)	(None, 128, 128, 64)	18,496	max_pooling2d[0]
batch_normalizatio… (BatchNormalizatio…	(None, 128, 128, 64)	256	conv2d_11[0][0]
re_lu_2 (ReLU)	(None, 128, 128, 64)	0	batch_normalizat
conv2d_12 (Conv2D)	(None, 128, 128, 64)	36,928	re_lu_2[0][0]

batch_normalizatio… (BatchNormalizatio…	(None, 128, 128, 64)	256	conv2d_12[0][0]
re_lu_3 (ReLU)	(None, 128, 128, 64)	0	batch_normalizat
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0	re_lu_3[0][0]
conv2d_13 (Conv2D)	(None, 64, 64, 128)	73,856	max_pooling2d_1
batch_normalizatio… (BatchNormalizatio…	(None, 64, 64, 128)	512	conv2d_13[0][0]
re_lu_4 (ReLU)	(None, 64, 64, 128)	0	batch_normalizat
conv2d_14 (Conv2D)	(None, 64, 64, 128)	147,584	re_lu_4[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 64, 64, 128)	512	conv2d_14[0][0]
re_lu_5 (ReLU)	(None, 64, 64, 128)	0	batch_normalizat
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 128)	0	re_lu_5[0][0]
conv2d_15 (Conv2D)	(None, 32, 32, 256)	295,168	max_pooling2d_2
batch_normalizatio… (BatchNormalizatio…	(None, 32, 32, 256)	1,024	conv2d_15[0][0]
re_lu_6 (ReLU)	(None, 32, 32, 256)	0	batch_normalizat
conv2d_16 (Conv2D)	(None, 32, 32, 256)	590,080	re_lu_6[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 32, 32, 256)	1,024	conv2d_16[0][0]
re_lu_7 (ReLU)	(None, 32, 32, 256)	0	batch_normalizat
dropout (Dropout)	(None, 32, 32, 256)	0	re_lu_7[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 256)	0	dropout[0][0]

conv2d_17 (Conv2D)	(None, 16, 16, 512)	1,180,160	max_pooling2d_3
batch_normalizatio… (BatchNormalizatio…	(None, 16, 16, 512)	2,048	conv2d_17[0][0]
re_lu_8 (ReLU)	(None, 16, 16, 512)	0	batch_normalizat
conv2d_18 (Conv2D)	(None, 16, 16, 512)	2,359,808	re_lu_8[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 16, 16, 512)	2,048	conv2d_18[0][0]
re_lu_9 (ReLU)	(None, 16, 16, 512)	0	batch_normalizat
dropout_1 (Dropout)	(None, 16, 16, 512)	0	re_lu_9[0][0]
conv2d_transpose_2 (Conv2DTranspose)	(None, 32, 32, 256)	1,179,904	dropout_1[0][0]
concatenate_2 (Concatenate)	(None, 32, 32, 512)	0	conv2d_transpose dropout[0][0]
conv2d_19 (Conv2D)	(None, 32, 32, 256)	1,179,904	concatenate_2[0]
batch_normalizatio… (BatchNormalizatio…	(None, 32, 32, 256)	1,024	conv2d_19[0][0]
re_lu_10 (ReLU)	(None, 32, 32, 256)	0	batch_normalizat
conv2d_20 (Conv2D)	(None, 32, 32, 256)	590,080	re_lu_10[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 32, 32, 256)	1,024	conv2d_20[0][0]
re_lu_11 (ReLU)	(None, 32, 32, 256)	0	batch_normalizat
conv2d_transpose_3 (Conv2DTranspose)	(None, 64, 64, 128)	295,040	re_lu_11[0][0]
concatenate_3 (Concatenate)	(None, 64, 64, 256)	0	conv2d_transpose re_lu_5[0][0]
conv2d_21 (Conv2D)	(None, 64, 64,	295,040	concatenate_3[0]

	128)		
batch_normalizatio… (BatchNormalizatio…	(None, 64, 64, 128)	512	conv2d_21[0][0]
re_lu_12 (ReLU)	(None, 64, 64, 128)	0	batch_normalizat
conv2d_22 (Conv2D)	(None, 64, 64, 128)	147,584	re_lu_12[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 64, 64, 128)	512	conv2d_22[0][0]
re_lu_13 (ReLU)	(None, 64, 64, 128)	0	batch_normalizat
conv2d_transpose_4 (Conv2DTranspose)	(None, 128, 128, 64)	73,792	re_lu_13[0][0]
concatenate_4 (Concatenate)	(None, 128, 128, 128)	0	conv2d_transpose re_lu_3[0][0]
conv2d_23 (Conv2D)	(None, 128, 128, 64)	73,792	concatenate_4[0]
batch_normalizatio… (BatchNormalizatio…	(None, 128, 128, 64)	256	conv2d_23[0][0]
re_lu_14 (ReLU)	(None, 128, 128, 64)	0	batch_normalizat
conv2d_24 (Conv2D)	(None, 128, 128, 64)	36,928	re_lu_14[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 128, 128, 64)	256	conv2d_24[0][0]
re_lu_15 (ReLU)	(None, 128, 128, 64)	0	batch_normalizat
conv2d_transpose_5 (Conv2DTranspose)	(None, 256, 256, 32)	18,464	re_lu_15[0][0]
concatenate_5 (Concatenate)	(None, 256, 256, 64)	0	conv2d_transpose re_lu_1[0][0]
conv2d_25 (Conv2D)	(None, 256, 256, 32)	18,464	concatenate_5[0]
batch_normalizatio… (BatchNormalizatio…	(None, 256, 256, 32)	128	conv2d_25[0][0]

re_lu_16 (ReLU)	(None, 256, 256, 32)	0	batch_normalizat
conv2d_26 (Conv2D)	(None, 256, 256, 32)	9,248	re_lu_16[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 256, 256, 32)	128	conv2d_26[0][0]
re_lu_17 (ReLU)	(None, 256, 256, 32)	0	batch_normalizat
conv2d_27 (Conv2D)	(None, 256, 256, 32)	9,248	re_lu_17[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 256, 256, 32)	128	conv2d_27[0][0]
re_lu_18 (ReLU)	(None, 256, 256, 32)	0	batch_normalizat
conv2d_28 (Conv2D)	(None, 256, 256, 13)	429	re_lu_18[0][0]

Total params: 8,652,045 (33.00 MB)

Trainable params: 8,646,093 (32.98 MB)

Non-trainable params: 5,952 (23.25 KB)

In [20]:

```
start_time = time.time()

unet_model1 = unet_model()
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,
                              patience=10, min_lr=1e-6)
early_stop = EarlyStopping(monitor='val_accuracy', patience=5, restore_
best_weights=True)

unet_result1 = unet_model1.fit(train_dataset,
                    validation_data=val_dataset,
                    epochs=EPOCHS,
                    callbacks=[reduce_lr,early_stop],
                    batch_size=BATCH_SIZE)
end_time = time.time()
```

```
Epoch 1/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 180s 1s/step - accuracy: 0.7155 - loss:
1.1250 - val_accuracy: 0.8096 - val_loss: 0.7900 - learning_rate: 0.001
0
Epoch 2/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 86s 701ms/step - accuracy: 0.9188 - loss
: 0.3165 - val_accuracy: 0.8040 - val_loss: 0.8075 - learning_rate: 0.0
010
Epoch 3/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 86s 707ms/step - accuracy: 0.9408 - loss
: 0.2140 - val_accuracy: 0.8134 - val_loss: 0.7329 - learning_rate: 0.0
010
Epoch 4/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 85s 702ms/step - accuracy: 0.9514 - loss
: 0.1691 - val_accuracy: 0.9216 - val_loss: 0.2815 - learning_rate: 0.0
010
Epoch 5/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 86s 702ms/step - accuracy: 0.9589 - loss
: 0.1393 - val_accuracy: 0.9363 - val_loss: 0.2229 - learning_rate: 0.0
010
```

```
Epoch 6/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 85s 704ms/step - accuracy: 0.9643 - loss
: 0.1176 - val_accuracy: 0.9401 - val_loss: 0.2012 - learning_rate: 0.0
010
Epoch 7/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 85s 702ms/step - accuracy: 0.9682 - loss
: 0.1021 - val_accuracy: 0.9607 - val_loss: 0.1281 - learning_rate: 0.0
010
Epoch 8/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 85s 700ms/step - accuracy: 0.9698 - loss
: 0.0948 - val_accuracy: 0.9672 - val_loss: 0.1028 - learning_rate: 0.0
010
Epoch 9/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 86s 708ms/step - accuracy: 0.9719 - loss
: 0.0873 - val_accuracy: 0.9697 - val_loss: 0.0944 - learning_rate: 0.0
010
Epoch 10/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 85s 702ms/step - accuracy: 0.9744 - loss
: 0.0781 - val_accuracy: 0.9706 - val_loss: 0.0904 - learning_rate: 0.0
010
Epoch 11/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 85s 702ms/step - accuracy: 0.9764 - loss
: 0.0709 - val_accuracy: 0.9723 - val_loss: 0.0842 - learning_rate: 0.0
010
Epoch 12/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 85s 705ms/step - accuracy: 0.9778 - loss
: 0.0660 - val_accuracy: 0.9745 - val_loss: 0.0774 - learning_rate: 0.0
010
Epoch 13/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 85s 699ms/step - accuracy: 0.9788 - loss
: 0.0623 - val_accuracy: 0.9649 - val_loss: 0.1127 - learning_rate: 0.0
010
Epoch 14/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 143s 706ms/step - accuracy: 0.9679 - los
s: 0.0989 - val_accuracy: 0.9740 - val_loss: 0.0789 - learning_rate: 0.
0010
Epoch 15/15
```

```
110/110 ━━━━━━━━━━━━━━━━━━━━ 86s 709ms/step - accuracy: 0.9786 - loss
: 0.0631 - val_accuracy: 0.9773 - val_loss: 0.0682 - learning_rate: 0.0
010
```

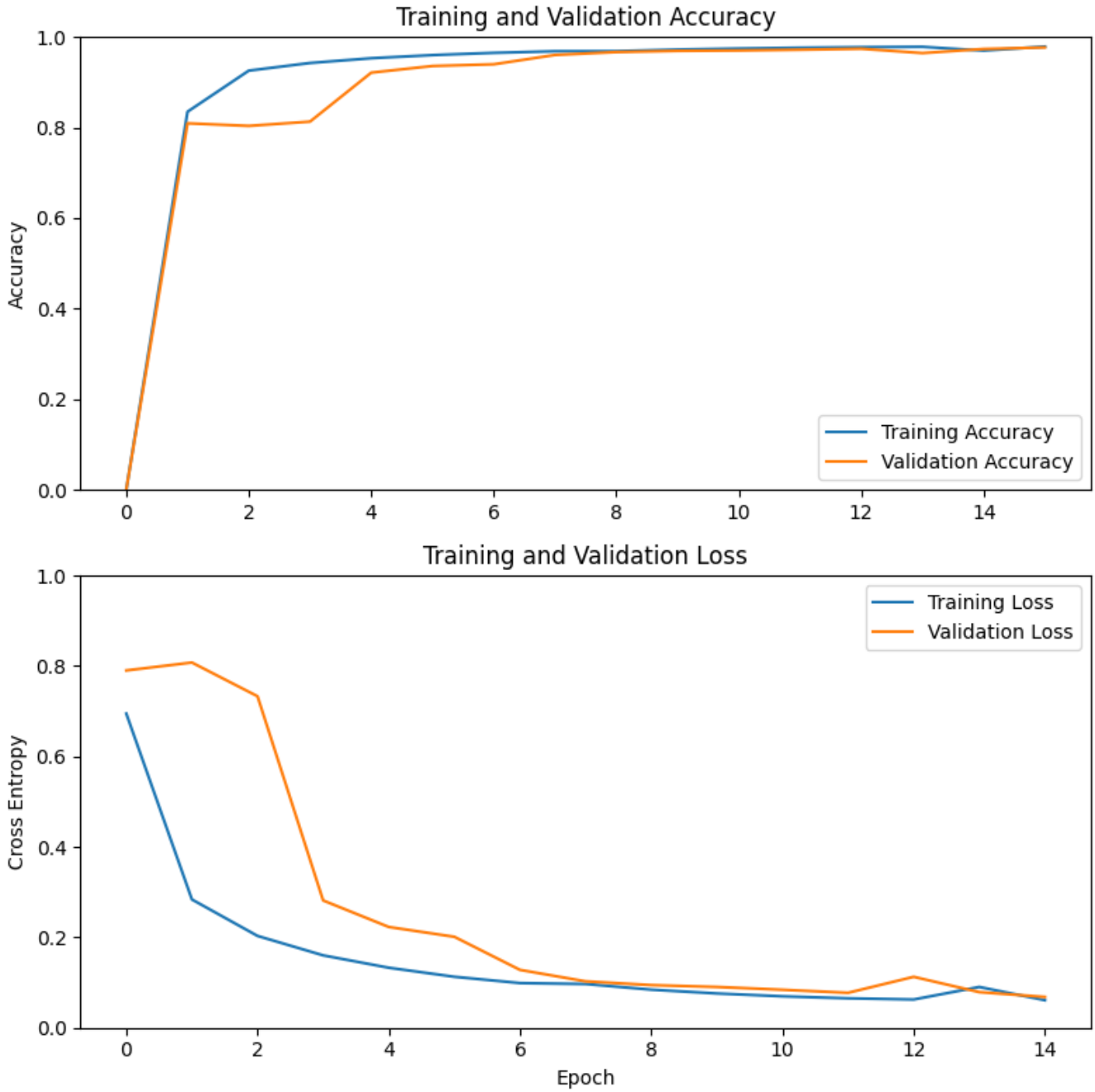
In [21]:

```
training_time = end_time - start_time
print("----------------------------------------------------------------
")
print("Training time for U-Net:", training_time, "seconds")
```

```
----------------------------------------------------------------
Training time for U-Net: 1433.0051529407501 seconds
```

In [22]:

```
plot_training_history(unet_result1)
```



- ### Training DeepLabV3

In [23]:
```
# Deeplab model
deeplab_model1 = DeeplabV3()
deeplab_model1.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-a
pplications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

Model: "functional_7"

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, 256, 256, 3)	0	–
conv1_pad (ZeroPadding2D)	(None, 262, 262, 3)	0	input_layer_3[0]
conv1_conv (Conv2D)	(None, 128, 128, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio…	(None, 128, 128, 64)	256	conv1_conv[0][0]
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 (Conv2D)	(None, 64, 64, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalizatio…	(None, 64, 64, 64)	256	conv2_block1_1_c
conv2_block1_1_relu (Activation)	(None, 64, 64, 64)	0	conv2_block1_1_b

conv2_block1_2_conv (Conv2D)	(None, 64, 64, 64)	36,928	conv2_block1_1_r
conv2_block1_2_bn (BatchNormalizatio…	(None, 64, 64, 64)	256	conv2_block1_2_c
conv2_block1_2_relu (Activation)	(None, 64, 64, 64)	0	conv2_block1_2_b
conv2_block1_0_conv (Conv2D)	(None, 64, 64, 256)	16,640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 64, 64, 256)	16,640	conv2_block1_2_r
conv2_block1_0_bn (BatchNormalizatio…	(None, 64, 64, 256)	1,024	conv2_block1_0_c
conv2_block1_3_bn (BatchNormalizatio…	(None, 64, 64, 256)	1,024	conv2_block1_3_c
conv2_block1_add (Add)	(None, 64, 64, 256)	0	conv2_block1_0_b conv2_block1_3_b
conv2_block1_out (Activation)	(None, 64, 64, 256)	0	conv2_block1_add
conv2_block2_1_conv (Conv2D)	(None, 64, 64, 64)	16,448	conv2_block1_out
conv2_block2_1_bn (BatchNormalizatio…	(None, 64, 64, 64)	256	conv2_block2_1_c
conv2_block2_1_relu (Activation)	(None, 64, 64, 64)	0	conv2_block2_1_b
conv2_block2_2_conv (Conv2D)	(None, 64, 64, 64)	36,928	conv2_block2_1_r
conv2_block2_2_bn (BatchNormalizatio…	(None, 64, 64, 64)	256	conv2_block2_2_c
conv2_block2_2_relu (Activation)	(None, 64, 64, 64)	0	conv2_block2_2_b
conv2_block2_3_conv (Conv2D)	(None, 64, 64, 256)	16,640	conv2_block2_2_r
conv2_block2_3_bn (BatchNormalizatio…	(None, 64, 64, 256)	1,024	conv2_block2_3_c
conv2_block2_add (Add)	(None, 64, 64, 256)	0	conv2_block1_out conv2_block2_3_b

conv2_block2_out (Activation)	(None, 64, 64, 256)	0	conv2_block2_add
conv2_block3_1_conv (Conv2D)	(None, 64, 64, 64)	16,448	conv2_block2_out
conv2_block3_1_bn (BatchNormalizatio…	(None, 64, 64, 64)	256	conv2_block3_1_c
conv2_block3_1_relu (Activation)	(None, 64, 64, 64)	0	conv2_block3_1_b
conv2_block3_2_conv (Conv2D)	(None, 64, 64, 64)	36,928	conv2_block3_1_r
conv2_block3_2_bn (BatchNormalizatio…	(None, 64, 64, 64)	256	conv2_block3_2_c
conv2_block3_2_relu (Activation)	(None, 64, 64, 64)	0	conv2_block3_2_b
conv2_block3_3_conv (Conv2D)	(None, 64, 64, 256)	16,640	conv2_block3_2_r
conv2_block3_3_bn (BatchNormalizatio…	(None, 64, 64, 256)	1,024	conv2_block3_3_c
conv2_block3_add (Add)	(None, 64, 64, 256)	0	conv2_block2_out conv2_block3_3_b
conv2_block3_out (Activation)	(None, 64, 64, 256)	0	conv2_block3_add
conv3_block1_1_conv (Conv2D)	(None, 32, 32, 128)	32,896	conv2_block3_out
conv3_block1_1_bn (BatchNormalizatio…	(None, 32, 32, 128)	512	conv3_block1_1_c
conv3_block1_1_relu (Activation)	(None, 32, 32, 128)	0	conv3_block1_1_b
conv3_block1_2_conv (Conv2D)	(None, 32, 32, 128)	147,584	conv3_block1_1_r
conv3_block1_2_bn (BatchNormalizatio…	(None, 32, 32, 128)	512	conv3_block1_2_c
conv3_block1_2_relu (Activation)	(None, 32, 32, 128)	0	conv3_block1_2_b
conv3_block1_0_conv	(None, 32, 32,	131,584	conv2_block3_out

(Conv2D)	512)		
conv3_block1_3_conv (Conv2D)	(None, 32, 32, 512)	66,048	conv3_block1_2_r
conv3_block1_0_bn (BatchNormalizatio…	(None, 32, 32, 512)	2,048	conv3_block1_0_c
conv3_block1_3_bn (BatchNormalizatio…	(None, 32, 32, 512)	2,048	conv3_block1_3_c
conv3_block1_add (Add)	(None, 32, 32, 512)	0	conv3_block1_0_b conv3_block1_3_b
conv3_block1_out (Activation)	(None, 32, 32, 512)	0	conv3_block1_add
conv3_block2_1_conv (Conv2D)	(None, 32, 32, 128)	65,664	conv3_block1_out
conv3_block2_1_bn (BatchNormalizatio…	(None, 32, 32, 128)	512	conv3_block2_1_c
conv3_block2_1_relu (Activation)	(None, 32, 32, 128)	0	conv3_block2_1_b
conv3_block2_2_conv (Conv2D)	(None, 32, 32, 128)	147,584	conv3_block2_1_r
conv3_block2_2_bn (BatchNormalizatio…	(None, 32, 32, 128)	512	conv3_block2_2_c
conv3_block2_2_relu (Activation)	(None, 32, 32, 128)	0	conv3_block2_2_b
conv3_block2_3_conv (Conv2D)	(None, 32, 32, 512)	66,048	conv3_block2_2_r
conv3_block2_3_bn (BatchNormalizatio…	(None, 32, 32, 512)	2,048	conv3_block2_3_c
conv3_block2_add (Add)	(None, 32, 32, 512)	0	conv3_block1_out conv3_block2_3_b
conv3_block2_out (Activation)	(None, 32, 32, 512)	0	conv3_block2_add
conv3_block3_1_conv (Conv2D)	(None, 32, 32, 128)	65,664	conv3_block2_out
conv3_block3_1_bn (BatchNormalizatio…	(None, 32, 32, 128)	512	conv3_block3_1_c

conv3_block3_1_relu (Activation)	(None, 32, 32, 128)	0	conv3_block3_1_b
conv3_block3_2_conv (Conv2D)	(None, 32, 32, 128)	147,584	conv3_block3_1_r
conv3_block3_2_bn (BatchNormalizatio…	(None, 32, 32, 128)	512	conv3_block3_2_c
conv3_block3_2_relu (Activation)	(None, 32, 32, 128)	0	conv3_block3_2_b
conv3_block3_3_conv (Conv2D)	(None, 32, 32, 512)	66,048	conv3_block3_2_r
conv3_block3_3_bn (BatchNormalizatio…	(None, 32, 32, 512)	2,048	conv3_block3_3_c
conv3_block3_add (Add)	(None, 32, 32, 512)	0	conv3_block2_out conv3_block3_3_b
conv3_block3_out (Activation)	(None, 32, 32, 512)	0	conv3_block3_add
conv3_block4_1_conv (Conv2D)	(None, 32, 32, 128)	65,664	conv3_block3_out
conv3_block4_1_bn (BatchNormalizatio…	(None, 32, 32, 128)	512	conv3_block4_1_c
conv3_block4_1_relu (Activation)	(None, 32, 32, 128)	0	conv3_block4_1_b
conv3_block4_2_conv (Conv2D)	(None, 32, 32, 128)	147,584	conv3_block4_1_r
conv3_block4_2_bn (BatchNormalizatio…	(None, 32, 32, 128)	512	conv3_block4_2_c
conv3_block4_2_relu (Activation)	(None, 32, 32, 128)	0	conv3_block4_2_b
conv3_block4_3_conv (Conv2D)	(None, 32, 32, 512)	66,048	conv3_block4_2_r
conv3_block4_3_bn (BatchNormalizatio…	(None, 32, 32, 512)	2,048	conv3_block4_3_c
conv3_block4_add (Add)	(None, 32, 32, 512)	0	conv3_block3_out conv3_block4_3_b
conv3_block4_out (Activation)	(None, 32, 32, 512)	0	conv3_block4_add

conv4_block1_1_conv (Conv2D)	(None, 16, 16, 256)	131,328	conv3_block4_out
conv4_block1_1_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block1_1_c
conv4_block1_1_relu (Activation)	(None, 16, 16, 256)	0	conv4_block1_1_b
conv4_block1_2_conv (Conv2D)	(None, 16, 16, 256)	590,080	conv4_block1_1_r
conv4_block1_2_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block1_2_c
conv4_block1_2_relu (Activation)	(None, 16, 16, 256)	0	conv4_block1_2_b
conv4_block1_0_conv (Conv2D)	(None, 16, 16, 1024)	525,312	conv3_block4_out
conv4_block1_3_conv (Conv2D)	(None, 16, 16, 1024)	263,168	conv4_block1_2_r
conv4_block1_0_bn (BatchNormalizatio…	(None, 16, 16, 1024)	4,096	conv4_block1_0_c
conv4_block1_3_bn (BatchNormalizatio…	(None, 16, 16, 1024)	4,096	conv4_block1_3_c
conv4_block1_add (Add)	(None, 16, 16, 1024)	0	conv4_block1_0_b conv4_block1_3_b
conv4_block1_out (Activation)	(None, 16, 16, 1024)	0	conv4_block1_add
conv4_block2_1_conv (Conv2D)	(None, 16, 16, 256)	262,400	conv4_block1_out
conv4_block2_1_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block2_1_c
conv4_block2_1_relu (Activation)	(None, 16, 16, 256)	0	conv4_block2_1_b
conv4_block2_2_conv (Conv2D)	(None, 16, 16, 256)	590,080	conv4_block2_1_r
conv4_block2_2_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block2_2_c
conv4_block2_2_relu	(None, 16, 16,	0	conv4_block2_2_b

(Activation)	256)		
conv4_block2_3_conv (Conv2D)	(None, 16, 16, 1024)	263,168	conv4_block2_2_r
conv4_block2_3_bn (BatchNormalizatio…	(None, 16, 16, 1024)	4,096	conv4_block2_3_c
conv4_block2_add (Add)	(None, 16, 16, 1024)	0	conv4_block1_out conv4_block2_3_b
conv4_block2_out (Activation)	(None, 16, 16, 1024)	0	conv4_block2_add
conv4_block3_1_conv (Conv2D)	(None, 16, 16, 256)	262,400	conv4_block2_out
conv4_block3_1_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block3_1_c
conv4_block3_1_relu (Activation)	(None, 16, 16, 256)	0	conv4_block3_1_b
conv4_block3_2_conv (Conv2D)	(None, 16, 16, 256)	590,080	conv4_block3_1_r
conv4_block3_2_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block3_2_c
conv4_block3_2_relu (Activation)	(None, 16, 16, 256)	0	conv4_block3_2_b
conv4_block3_3_conv (Conv2D)	(None, 16, 16, 1024)	263,168	conv4_block3_2_r
conv4_block3_3_bn (BatchNormalizatio…	(None, 16, 16, 1024)	4,096	conv4_block3_3_c
conv4_block3_add (Add)	(None, 16, 16, 1024)	0	conv4_block2_out conv4_block3_3_b
conv4_block3_out (Activation)	(None, 16, 16, 1024)	0	conv4_block3_add
conv4_block4_1_conv (Conv2D)	(None, 16, 16, 256)	262,400	conv4_block3_out
conv4_block4_1_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block4_1_c
conv4_block4_1_relu (Activation)	(None, 16, 16, 256)	0	conv4_block4_1_b

conv4_block4_2_conv (Conv2D)	(None, 16, 16, 256)	590,080	conv4_block4_1_r
conv4_block4_2_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block4_2_c
conv4_block4_2_relu (Activation)	(None, 16, 16, 256)	0	conv4_block4_2_b
conv4_block4_3_conv (Conv2D)	(None, 16, 16, 1024)	263,168	conv4_block4_2_r
conv4_block4_3_bn (BatchNormalizatio…	(None, 16, 16, 1024)	4,096	conv4_block4_3_c
conv4_block4_add (Add)	(None, 16, 16, 1024)	0	conv4_block3_out conv4_block4_3_b
conv4_block4_out (Activation)	(None, 16, 16, 1024)	0	conv4_block4_add
conv4_block5_1_conv (Conv2D)	(None, 16, 16, 256)	262,400	conv4_block4_out
conv4_block5_1_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block5_1_c
conv4_block5_1_relu (Activation)	(None, 16, 16, 256)	0	conv4_block5_1_b
conv4_block5_2_conv (Conv2D)	(None, 16, 16, 256)	590,080	conv4_block5_1_r
conv4_block5_2_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block5_2_c
conv4_block5_2_relu (Activation)	(None, 16, 16, 256)	0	conv4_block5_2_b
conv4_block5_3_conv (Conv2D)	(None, 16, 16, 1024)	263,168	conv4_block5_2_r
conv4_block5_3_bn (BatchNormalizatio…	(None, 16, 16, 1024)	4,096	conv4_block5_3_c
conv4_block5_add (Add)	(None, 16, 16, 1024)	0	conv4_block4_out conv4_block5_3_b
conv4_block5_out (Activation)	(None, 16, 16, 1024)	0	conv4_block5_add
conv4_block6_1_conv (Conv2D)	(None, 16, 16, 256)	262,400	conv4_block5_out

conv4_block6_1_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block6_1_c
conv4_block6_1_relu (Activation)	(None, 16, 16, 256)	0	conv4_block6_1_b
conv4_block6_2_conv (Conv2D)	(None, 16, 16, 256)	590,080	conv4_block6_1_r
conv4_block6_2_bn (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv4_block6_2_c
conv4_block6_2_relu (Activation)	(None, 16, 16, 256)	0	conv4_block6_2_b
average_pooling2d (AveragePooling2D)	(None, 1, 1, 256)	0	conv4_block6_2_r
conv2d_49 (Conv2D)	(None, 1, 1, 256)	65,536	average_pooling2
batch_normalizatio… (BatchNormalizatio…	(None, 1, 1, 256)	1,024	conv2d_49[0][0]
conv2d_50 (Conv2D)	(None, 16, 16, 256)	65,536	conv4_block6_2_r
conv2d_51 (Conv2D)	(None, 16, 16, 256)	65,536	conv4_block6_2_r
conv2d_52 (Conv2D)	(None, 16, 16, 256)	65,536	conv4_block6_2_r
conv2d_53 (Conv2D)	(None, 16, 16, 256)	65,536	conv4_block6_2_r
activation_8 (Activation)	(None, 1, 1, 256)	0	batch_normalizat
batch_normalizatio… (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv2d_50[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv2d_51[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv2d_52[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv2d_53[0][0]
up_sampling2d (UpSampling2D)	(None, 16, 16, 256)	0	activation_8[0][

activation_9 (Activation)	(None, 16, 16, 256)	0	batch_normalizat
activation_10 (Activation)	(None, 16, 16, 256)	0	batch_normalizat
activation_11 (Activation)	(None, 16, 16, 256)	0	batch_normalizat
activation_12 (Activation)	(None, 16, 16, 256)	0	batch_normalizat
concatenate_10 (Concatenate)	(None, 16, 16, 1280)	0	up_sampling2d[0] activation_9[0] activation_10[0] activation_11[0] activation_12[0]
conv2d_54 (Conv2D)	(None, 16, 16, 256)	327,680	concatenate_10[0
batch_normalizatio… (BatchNormalizatio…	(None, 16, 16, 256)	1,024	conv2d_54[0][0]
conv2d_55 (Conv2D)	(None, 64, 64, 48)	3,072	conv2_block3_2_r
activation_13 (Activation)	(None, 16, 16, 256)	0	batch_normalizat
batch_normalizatio… (BatchNormalizatio…	(None, 64, 64, 48)	192	conv2d_55[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 64, 64, 256)	0	activation_13[0]
activation_14 (Activation)	(None, 64, 64, 48)	0	batch_normalizat
concatenate_11 (Concatenate)	(None, 64, 64, 304)	0	up_sampling2d_1 activation_14[0]
conv2d_56 (Conv2D)	(None, 64, 64, 256)	700,416	concatenate_11[0
batch_normalizatio… (BatchNormalizatio…	(None, 64, 64, 256)	1,024	conv2d_56[0][0]
activation_15 (Activation)	(None, 64, 64, 256)	0	batch_normalizat
conv2d_57 (Conv2D)	(None, 64, 64,	589,824	activation_15[0]

	256)		
batch_normalizatio… (BatchNormalizatio…	(None, 64, 64, 256)	1,024	conv2d_57[0][0]
activation_16 (Activation)	(None, 64, 64, 256)	0	batch_normalizat
up_sampling2d_2 (UpSampling2D)	(None, 256, 256, 256)	0	activation_16[0]
output_layer (Conv2D)	(None, 256, 256, 13)	3,341	up_sampling2d_2
activation_17 (Activation)	(None, 256, 256, 13)	0	output_layer[0]

Total params: 10,282,317 (39.22 MB)

Trainable params: 10,249,581 (39.10 MB)

Non-trainable params: 32,736 (127.88 KB)

In [24]:

```
start_time = time.time()

early_stop = EarlyStopping(monitor='val_accuracy', patience=5, restore_
best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy',factor=1e-1, patie
nce=10, verbose=1, min_lr = 1e-6)

deeplab_result1 = deeplab_model1.fit(train_dataset,
                    validation_data = val_dataset,
                    epochs = EPOCHS,
                    callbacks=[early_stop, reduce_lr],
                    batch_size = BATCH_SIZE)
end_time = time.time()
```

```
Epoch 1/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 248s 1s/step - accuracy: 0.8340 - loss:
0.5719 - val_accuracy: 0.3233 - val_loss: 2.0047 - learning_rate: 0.001
0
Epoch 2/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 101s 842ms/step - accuracy: 0.9427 - los
s: 0.1571 - val_accuracy: 0.4221 - val_loss: 2.1206 - learning_rate: 0.
0010
Epoch 3/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 838ms/step - accuracy: 0.9516 - los
s: 0.1269 - val_accuracy: 0.4205 - val_loss: 2.4646 - learning_rate: 0.
0010
Epoch 4/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 837ms/step - accuracy: 0.9558 - los
s: 0.1135 - val_accuracy: 0.4405 - val_loss: 2.1882 - learning_rate: 0.
0010
Epoch 5/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 840ms/step - accuracy: 0.9576 - los
s: 0.1077 - val_accuracy: 0.5647 - val_loss: 1.4782 - learning_rate: 0.
0010
Epoch 6/15
```

```
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 840ms/step - accuracy: 0.9595 - los
s: 0.1018 - val_accuracy: 0.6184 - val_loss: 1.4269 - learning_rate: 0.
0010
Epoch 7/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 101s 840ms/step - accuracy: 0.9611 - los
s: 0.0972 - val_accuracy: 0.7973 - val_loss: 0.7535 - learning_rate: 0.
0010
Epoch 8/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 101s 843ms/step - accuracy: 0.9622 - los
s: 0.0937 - val_accuracy: 0.8655 - val_loss: 0.4462 - learning_rate: 0.
0010
Epoch 9/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 841ms/step - accuracy: 0.9624 - los
s: 0.0928 - val_accuracy: 0.9152 - val_loss: 0.2511 - learning_rate: 0.
0010
Epoch 10/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 101s 842ms/step - accuracy: 0.9631 - los
s: 0.0906 - val_accuracy: 0.9358 - val_loss: 0.1827 - learning_rate: 0.
0010
Epoch 11/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 839ms/step - accuracy: 0.9636 - los
s: 0.0892 - val_accuracy: 0.9519 - val_loss: 0.1265 - learning_rate: 0.
0010
Epoch 12/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 839ms/step - accuracy: 0.9640 - los
s: 0.0878 - val_accuracy: 0.9587 - val_loss: 0.1035 - learning_rate: 0.
0010
Epoch 13/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 836ms/step - accuracy: 0.9644 - los
s: 0.0866 - val_accuracy: 0.9578 - val_loss: 0.1059 - learning_rate: 0.
0010
Epoch 14/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 100s 839ms/step - accuracy: 0.9645 - los
s: 0.0861 - val_accuracy: 0.9407 - val_loss: 0.2083 - learning_rate: 0.
0010
Epoch 15/15
110/110 ━━━━━━━━━━━━━━━━━━━━ 101s 842ms/step - accuracy: 0.9607 - los
```

```
s: 0.0993 - val_accuracy: 0.8211 - val_loss: 0.7244 - learning_rate: 0.
0010
```

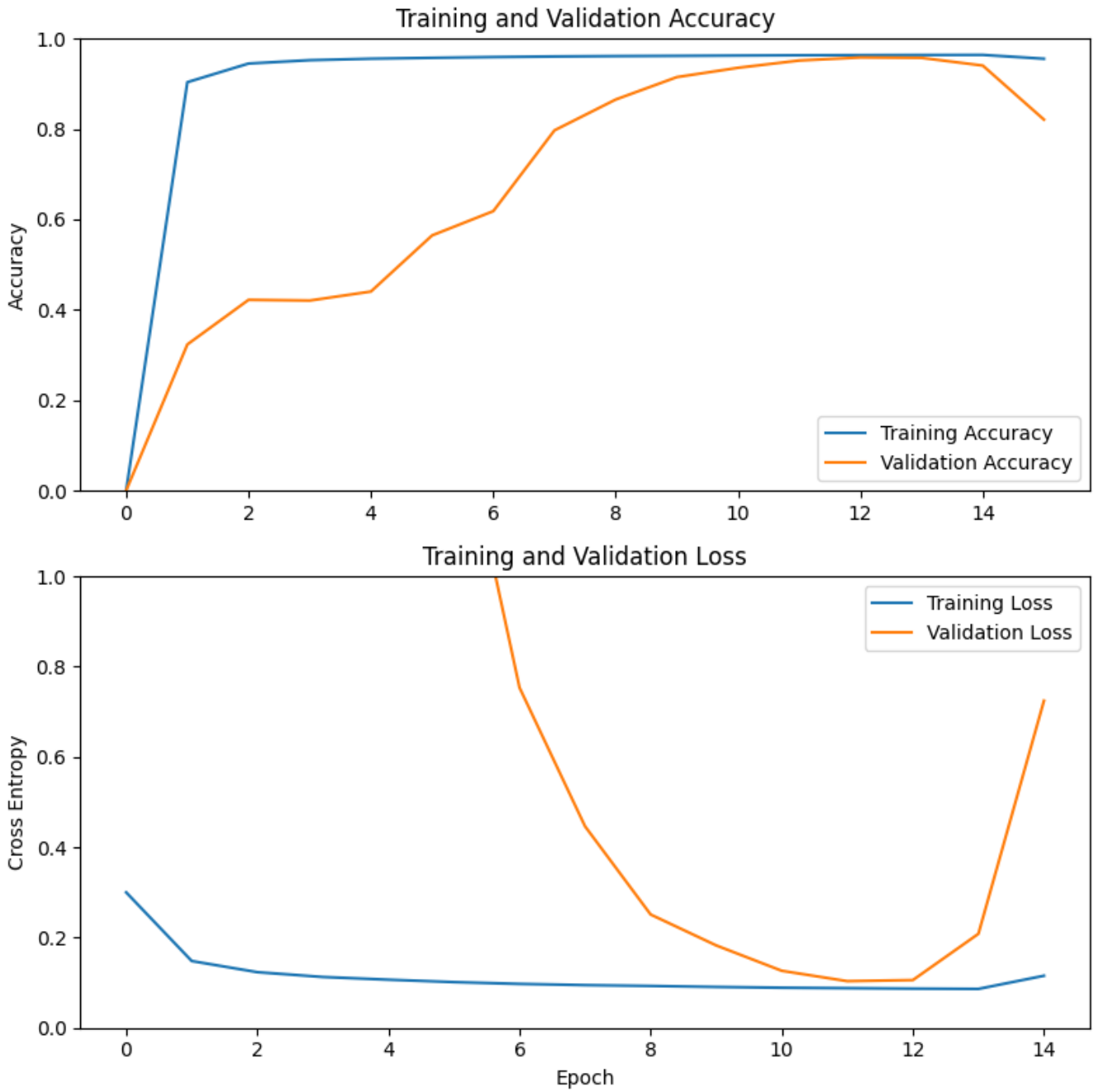
In [25]:

```
training_time = end_time - start_time
print("----------------------------------------------------------------
")
print("Training time for Deeplab:", training_time, "seconds")
```

```
----------------------------------------------------------------
Training time for Deeplab: 1653.4657363891602 seconds
```

In [26]:

```
plot_training_history(deeplab_result1)
```



4. Evaluation

- ### Accuracy

In [27]:
```
def model_accuracy(model):
    train_loss, train_accuracy = model.evaluate(train_dataset, batch_si
ze = BATCH_SIZE)
    validation_loss, validation_accuracy = model.evaluate(val_dataset,
batch_size = BATCH_SIZE)
    test_loss, test_accuracy = model.evaluate(test_dataset, batch_size
= BATCH_SIZE)
    print(f'Model Accuracy on the Training Dataset: {round(train_accura
cy * 100, 2)}%')
    print(f'Model Accuracy on the Validation Dataset: {round(validation
_accuracy * 100, 2)}%')
    print(f'Model Accuracy on the Test Dataset: {round(test_accuracy *
100, 2)}%')
```

In [28]:
```
model_accuracy(fcn_model1)
```

```
110/110 ━━━━━━━━━━━━━━━━━━━━ 39s 286ms/step - accuracy: 0.9089 - los
s: 0.3347
24/24 ━━━━━━━━━━━━━━━━━━━━ 41s 220ms/step - accuracy: 0.9044 - loss:
0.3474
24/24 ━━━━━━━━━━━━━━━━━━━━ 43s 118ms/step - accuracy: 0.9088 - loss:
0.3353
Model Accuracy on the Training Dataset: 90.92%
Model Accuracy on the Validation Dataset: 90.58%
Model Accuracy on the Test Dataset: 90.85%
```

In [29]:
```
model_accuracy(unet_model1)
```

```
110/110 ━━━━━━━━━━━━━━━━━━━━ 40s 288ms/step - accuracy: 0.9794 - los
s: 0.0610
24/24 ━━━━━━━━━━━━━━━━━━━━ 41s 213ms/step - accuracy: 0.9777 - loss:
0.0673
24/24 ━━━━━━━━━━━━━━━━━━━━ 42s 104ms/step - accuracy: 0.9783 - loss:
0.0656
Model Accuracy on the Training Dataset: 97.95%
Model Accuracy on the Validation Dataset: 97.73%
Model Accuracy on the Test Dataset: 97.83%
```

In [30]:
```
model_accuracy(deeplab_model1)
```

```
110/110 ━━━━━━━━━━━━━━━━━━━━ 43s 322ms/step - accuracy: 0.9617 - los
s: 0.0934
24/24 ━━━━━━━━━━━━━━━━━━━━ 42s 263ms/step - accuracy: 0.9594 - loss:
0.1015
24/24 ━━━━━━━━━━━━━━━━━━━━ 44s 202ms/step - accuracy: 0.9602 - loss:
0.0991
Model Accuracy on the Training Dataset: 96.16%
Model Accuracy on the Validation Dataset: 95.87%
Model Accuracy on the Test Dataset: 96.0%
```

- ### Intersection-over-Union (IoU)

In [31]:
```
import numpy as np
import tensorflow as tf
import pandas as pd
import numpy as np
```

```
def evaluate_metrics(dataset, model, n_classes=13):
    # create mask
    true_masks, predicted_masks = [], []
    for images, masks in dataset:
        pred_masks = model.predict(images)
        pred_masks = tf.expand_dims(tf.argmax(pred_masks, axis=-1), axi
s=-1)
        true_masks.extend(masks)
        predicted_masks.extend(pred_masks)
    true_masks = np.array(true_masks)
    predicted_masks = np.array(predicted_masks)

    # initialize
    class_wise_metrics = {
        "TP": np.zeros(n_classes),  # True Positives
        "FP": np.zeros(n_classes),  # False Positives
        "FN": np.zeros(n_classes)   # False Negatives
    }

    # calculate metrics per class
    for c_id in range(n_classes):
        true_positive = (predicted_masks == c_id) & (true_masks == c_id
)
        false_positive = (predicted_masks == c_id) & (true_masks != c_i
d)
        false_negative = (predicted_masks != c_id) & (true_masks == c_i
d)

        class_wise_metrics["TP"][c_id] += np.sum(true_positive)
        class_wise_metrics["FP"][c_id] += np.sum(false_positive)
        class_wise_metrics["FN"][c_id] += np.sum(false_negative)

    # class-wise metrics
    recall = np.round(class_wise_metrics["TP"] / (class_wise_metrics["T
P"] + class_wise_metrics["FN"] + 1e-6), 2)
    precision = np.round(class_wise_metrics["TP"] / (class_wise_metrics
```

```
["TP"] + class_wise_metrics["FP"] + 1e-6), 2)
    iou = np.round(class_wise_metrics["TP"] / (class_wise_metrics["TP"]
+ class_wise_metrics["FP"] + class_wise_metrics["FN"] + 1e-6), 2)

    # overall metrics
    overall_recall = np.mean(recall)
    overall_precision = np.mean(precision)
    overall_iou = np.mean(iou)

    # package evaluations
    evaluations = {
        "class_wise": {
            "Recall": recall.tolist(),
            "Precision": precision.tolist(),
            "IoU": iou.tolist()
        },
        "overall": {
            "Recall": round(overall_recall, 2),
            "Precision": round(overall_precision, 2),
            "IoU": round(overall_iou, 2)
        }
    }

    return evaluations


def show_evaluations(evaluations):
    class_wise_data = evaluations['class_wise']
    class_wise_df = pd.DataFrame(class_wise_data)
    class_wise_df.index = [f"Class {i+1}" for i in range(len(class_wis
e_data['Recall']))]

    overall_data = evaluations['overall']
    overall_df = pd.DataFrame(overall_data, index=['All Classes'])

    # Combine
    combined_df = pd.concat([overall_df, class_wise_df]).reset_index()
```

```
    combined_df.rename(columns={'index': 'Class'}, inplace=True)

    return combined_df
```

fcn

In [32]:

```
evaluations = evaluate_metrics(train_dataset, fcn_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 709ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 499ms/step
```

Out[32]:

	Class	Recall	Precision	IoU
0	All Classes	0.61	0.75	0.55
1	Class 1	0.97	0.98	0.96
2	Class 2	0.83	0.77	0.66
3	Class 3	0.46	0.73	0.40
4	Class 4	0.29	0.59	0.24
5	Class 5	0.00	0.00	0.00
6	Class 6	0.33	0.60	0.27
7	Class 7	0.89	0.98	0.88
8	Class 8	0.93	0.91	0.86
9	Class 9	0.79	0.82	0.68
10	Class 10	0.92	0.87	0.81
11	Class 11	0.96	0.94	0.90
12	Class 12	0.37	0.72	0.32
13	Class 13	0.21	0.78	0.20

In [33]:

```
evaluations = evaluate_metrics(val_dataset, fcn_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 276ms/step
```

Out[33]:

	Class	Recall	Precision	IoU
0	All Classes	0.61	0.74	0.54
1	Class 1	0.97	0.98	0.95
2	Class 2	0.81	0.76	0.65
3	Class 3	0.48	0.74	0.41
4	Class 4	0.26	0.56	0.22
5	Class 5	0.00	0.00	0.00
6	Class 6	0.33	0.60	0.27
7	Class 7	0.88	0.98	0.87
8	Class 8	0.93	0.91	0.85
9	Class 9	0.79	0.81	0.66
10	Class 10	0.92	0.87	0.81
11	Class 11	0.96	0.94	0.90
12	Class 12	0.36	0.72	0.31
13	Class 13	0.19	0.73	0.18

In [34]:

```
evaluations = evaluate_metrics(test_dataset, fcn_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```

Out[34]:

	Class	Recall	Precision	IoU
0	All Classes	0.61	0.74	0.55
1	Class 1	0.97	0.98	0.96
2	Class 2	0.82	0.78	0.67
3	Class 3	0.47	0.74	0.40
4	Class 4	0.29	0.59	0.24
5	Class 5	0.00	0.00	0.00
6	Class 6	0.33	0.59	0.27
7	Class 7	0.90	0.98	0.88
8	Class 8	0.93	0.91	0.85
9	Class 9	0.79	0.82	0.68
10	Class 10	0.92	0.87	0.81
11	Class 11	0.97	0.93	0.90
12	Class 12	0.35	0.71	0.31
13	Class 13	0.22	0.71	0.20

unet

In [35]:

```
evaluations = evaluate_metrics(train_dataset, unet_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
```

Out[35]:

	Class	Recall	Precision	IoU
0	All Classes	0.82	0.87	0.78
1	Class 1	0.99	0.99	0.98
2	Class 2	0.98	0.95	0.94
3	Class 3	0.76	0.84	0.66
4	Class 4	0.74	0.91	0.69
5	Class 5	0.00	0.00	0.00
6	Class 6	0.66	0.90	0.61
7	Class 7	0.97	0.98	0.95
8	Class 8	1.00	1.00	0.99
9	Class 9	0.98	0.97	0.95
10	Class 10	0.96	0.96	0.92
11	Class 11	0.99	0.99	0.99
12	Class 12	0.87	0.92	0.80
13	Class 13	0.72	0.86	0.65

In [36]:

```
evaluations = evaluate_metrics(val_dataset, unet_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 749ms/step
```

Out[36]:

	Class	Recall	Precision	IoU
0	All Classes	0.80	0.86	0.76
1	Class 1	0.99	0.99	0.98
2	Class 2	0.98	0.94	0.92
3	Class 3	0.76	0.83	0.66
4	Class 4	0.68	0.89	0.62
5	Class 5	0.00	0.00	0.00
6	Class 6	0.62	0.89	0.57
7	Class 7	0.97	0.97	0.94
8	Class 8	0.99	0.99	0.99
9	Class 9	0.98	0.96	0.94
10	Class 10	0.96	0.96	0.92
11	Class 11	0.99	0.99	0.99
12	Class 12	0.85	0.90	0.78
13	Class 13	0.68	0.85	0.60

In [37]:

```
evaluations = evaluate_metrics(test_dataset, unet_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```

Out[37]:

	Class	Recall	Precision	IoU
0	All Classes	0.81	0.86	0.77
1	Class 1	0.99	0.99	0.98
2	Class 2	0.98	0.95	0.93
3	Class 3	0.76	0.83	0.66
4	Class 4	0.72	0.90	0.67
5	Class 5	0.00	0.00	0.00
6	Class 6	0.62	0.89	0.58
7	Class 7	0.97	0.98	0.95
8	Class 8	0.99	1.00	0.99
9	Class 9	0.98	0.97	0.94
10	Class 10	0.96	0.96	0.92
11	Class 11	0.99	0.99	0.99
12	Class 12	0.85	0.90	0.78
13	Class 13	0.67	0.83	0.58

deeplab

In [38]:
```
evaluations = evaluate_metrics(train_dataset, deeplab_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
```

Out[38]:

	Class	Recall	Precision	IoU
0	All Classes	0.86	0.84	0.76
1	Class 1	0.97	0.98	0.95
2	Class 2	0.96	0.97	0.93
3	Class 3	0.74	0.63	0.52
4	Class 4	0.85	0.81	0.71
5	Class 5	0.62	0.56	0.41
6	Class 6	0.73	0.68	0.54
7	Class 7	0.79	0.75	0.63
8	Class 8	0.98	0.99	0.97
9	Class 9	0.96	0.97	0.93
10	Class 10	0.93	0.90	0.84
11	Class 11	0.99	0.99	0.99
12	Class 12	0.93	0.85	0.80
13	Class 13	0.72	0.88	0.66

In [39]:

```
evaluations = evaluate_metrics(val_dataset, deeplab_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```

Out[39]:

	Class	Recall	Precision	IoU
0	All Classes	0.84	0.83	0.74
1	Class 1	0.97	0.98	0.95
2	Class 2	0.95	0.96	0.92
3	Class 3	0.74	0.62	0.51
4	Class 4	0.80	0.78	0.65
5	Class 5	0.53	0.56	0.37
6	Class 6	0.67	0.67	0.50
7	Class 7	0.79	0.75	0.62
8	Class 8	0.98	0.99	0.97
9	Class 9	0.95	0.97	0.92
10	Class 10	0.93	0.89	0.83
11	Class 11	0.99	0.99	0.99
12	Class 12	0.91	0.83	0.76
13	Class 13	0.66	0.86	0.60

In [40]:

```
evaluations = evaluate_metrics(test_dataset, deeplab_model1)
show_evaluations(evaluations)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```

Out[40]:

	Class	Recall	Precision	IoU
0	All Classes	0.81	0.84	0.73
1	Class 1	0.97	0.98	0.95
2	Class 2	0.96	0.97	0.93
3	Class 3	0.74	0.62	0.51
4	Class 4	0.81	0.80	0.68
5	Class 5	0.19	0.58	0.17
6	Class 6	0.69	0.67	0.51
7	Class 7	0.79	0.75	0.63
8	Class 8	0.98	0.99	0.97
9	Class 9	0.95	0.97	0.93
10	Class 10	0.93	0.90	0.84
11	Class 11	0.99	0.99	0.99
12	Class 12	0.90	0.84	0.77
13	Class 13	0.63	0.86	0.57

5. Prediction

- ### Predict on Train, Val, Test Dataset

In [41]:

```
def create_mask(pred_mask):
    pred_mask = tf.argmax(pred_mask, axis=-1)
    pred_mask = pred_mask[..., tf.newaxis]
    return pred_mask[0]
```

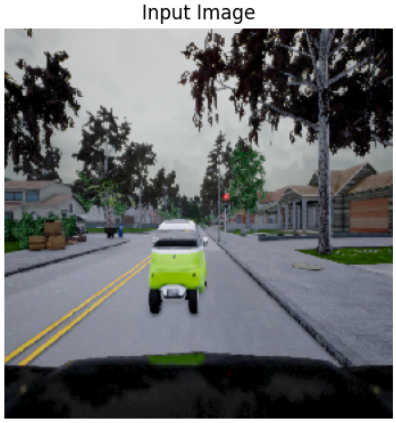
In [42]:
```
def display(display_list):
    plt.figure(figsize=(15, 15))

    title = ['Input Image', 'True Mask', 'Predicted Mask']

    for i in range(len(display_list)):
        plt.subplot(1, len(display_list), i+1)
        plt.title(title[i])
        plt.imshow(tf.keras.preprocessing.image.array_to_img(display_li
st[i]))
        plt.axis('off')
    plt.show()
```

In [43]:
```
def show_predictions(dataset, num, model):
    """
    Displays the first image of each of the num batches
    """
    if dataset:
        for image, mask in dataset.take(num):
            pred_mask = model.predict(image)
            display([image[0], mask[0], create_mask(pred_mask)])
    else:
        display([sample_image, sample_mask,
             create_mask(model.predict(sample_image[tf.newaxis, ...]))]
)
```

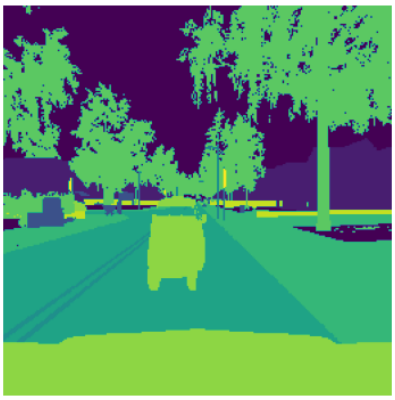
In [44]:
```
# fcn
show_predictions(train_dataset, 3, fcn_model1)
show_predictions(val_dataset, 3, fcn_model1)
show_predictions(test_dataset, 3, fcn_model1)
```

1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 53ms/step

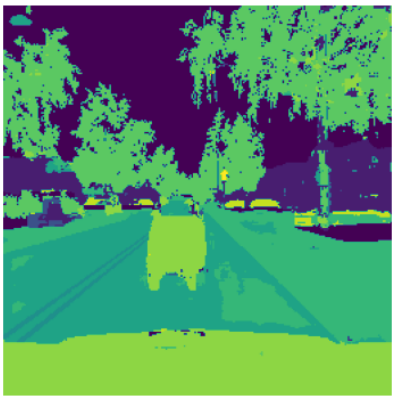
Input Image



True Mask

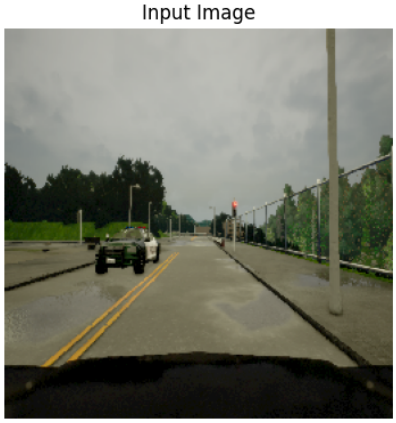


Predicted Mask



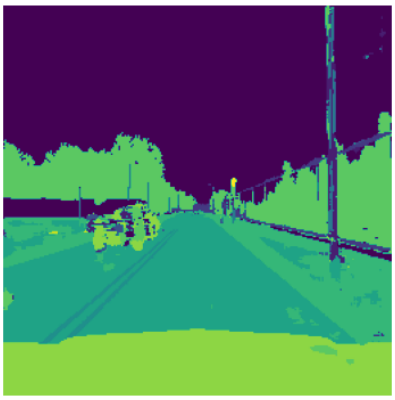
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 36ms/step

Input Image



True Mask

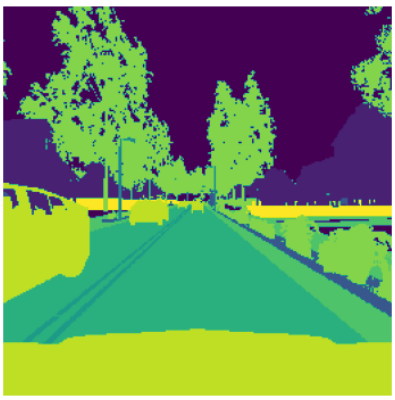
Predicted Mask



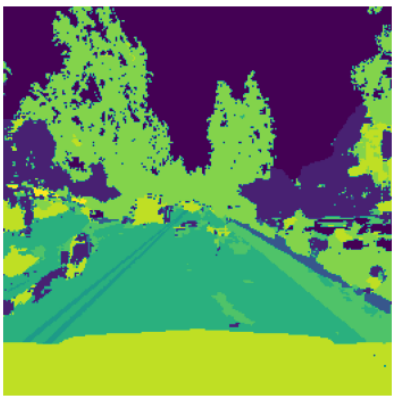
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 29ms/step

Input Image

True Mask

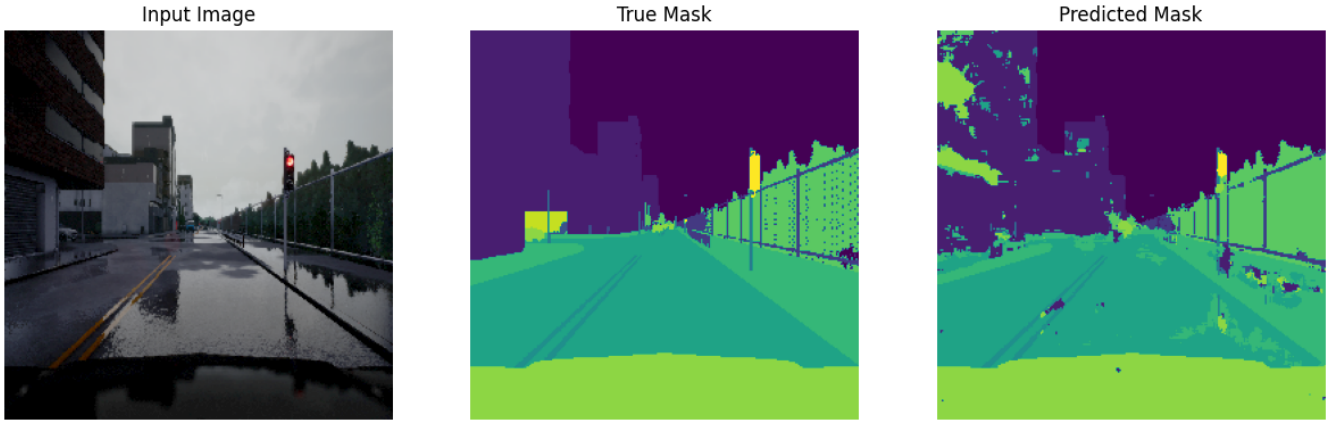


Predicted Mask



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
```

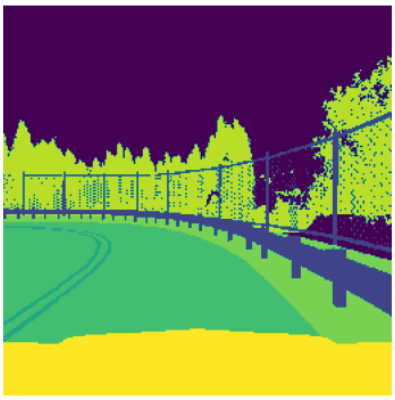

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
```



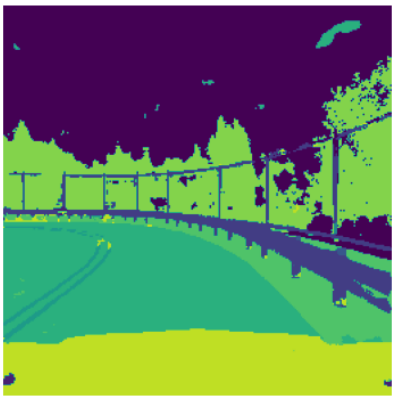

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
```

Input Image

True Mask

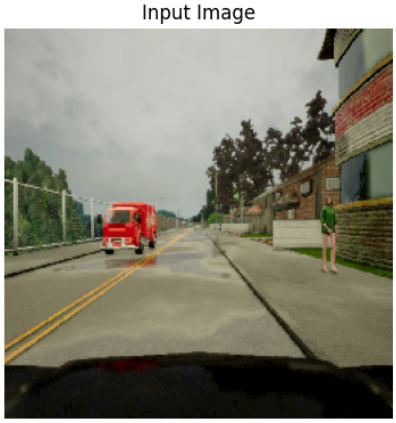


Predicted Mask



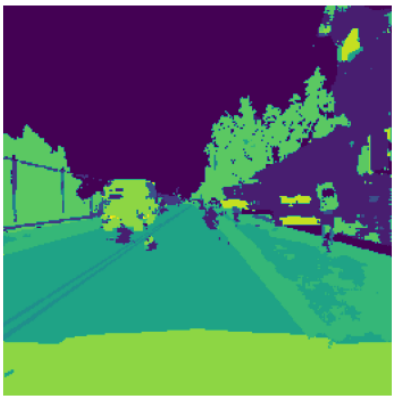
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 29ms/step

Input Image



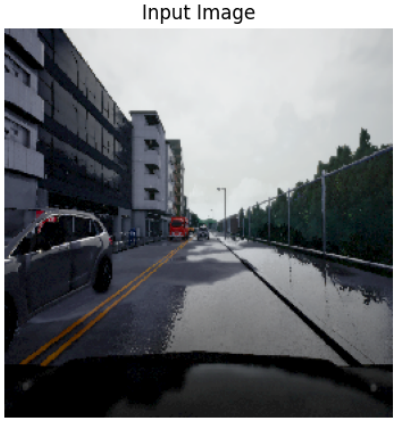
True Mask

Predicted Mask



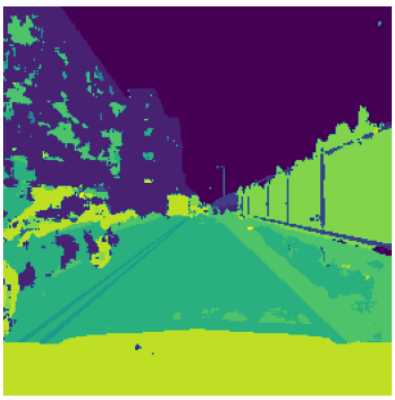
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 30ms/step

Input Image



True Mask

Predicted Mask



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
```

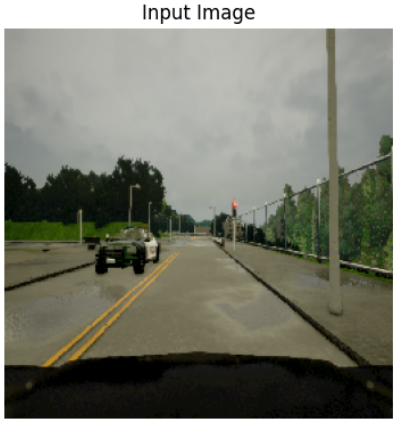
In [45]:

```
# unet
show_predictions(train_dataset, 3, unet_model1)
show_predictions(val_dataset, 3, unet_model1)
show_predictions(test_dataset, 3, unet_model1)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
```

Input Image



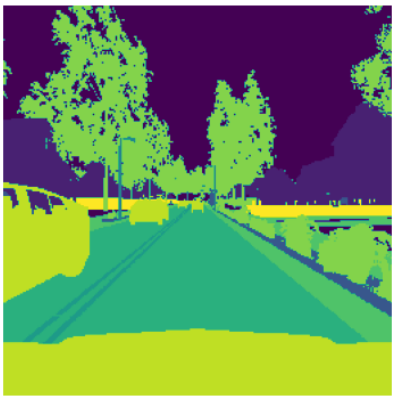
True Mask

Predicted Mask

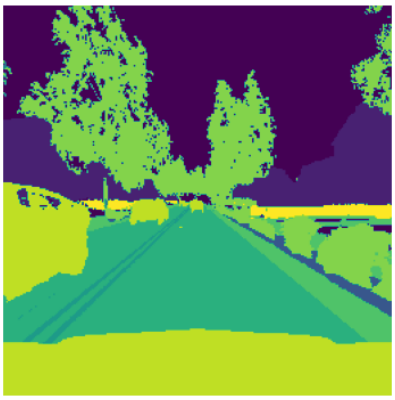
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 30ms/step

Input Image

True Mask



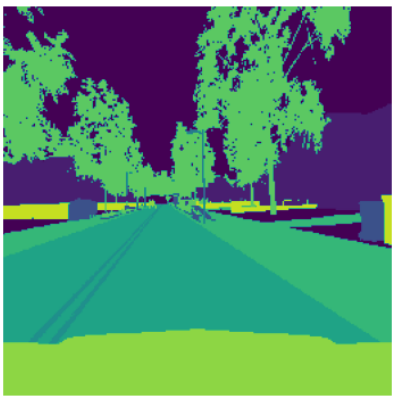
Predicted Mask



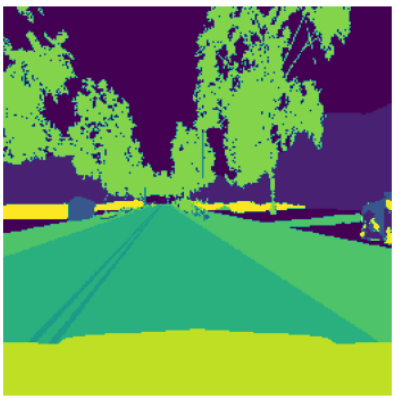
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 38ms/step

Input Image

True Mask

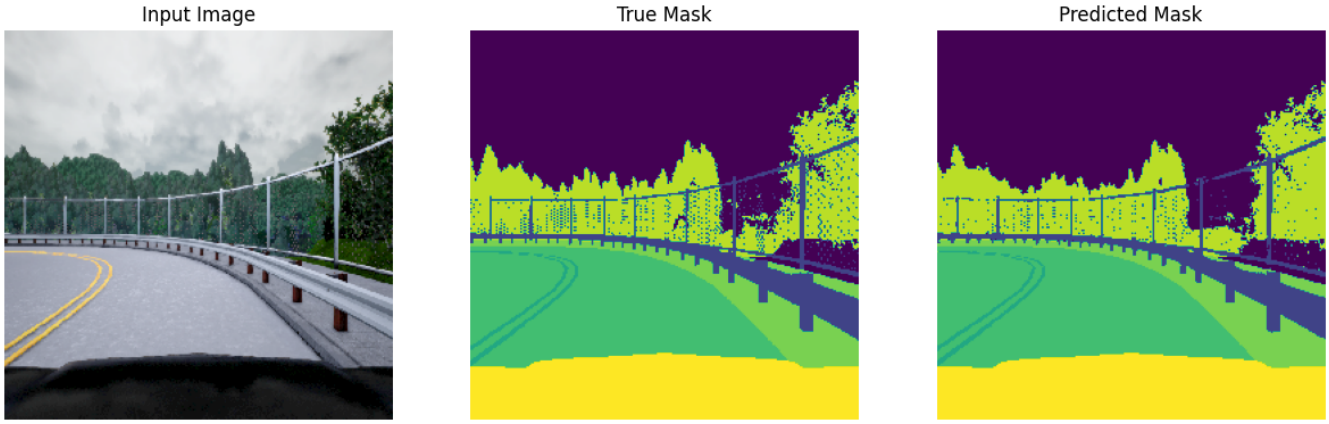


Predicted Mask



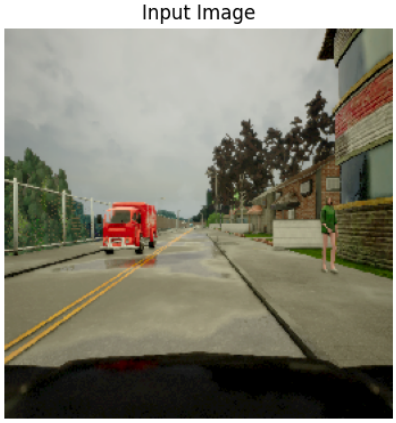
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
```

Input Image

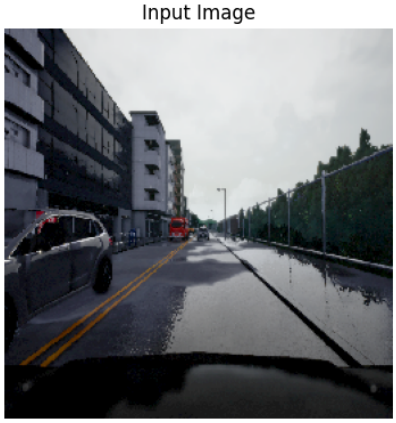


True Mask

Predicted Mask

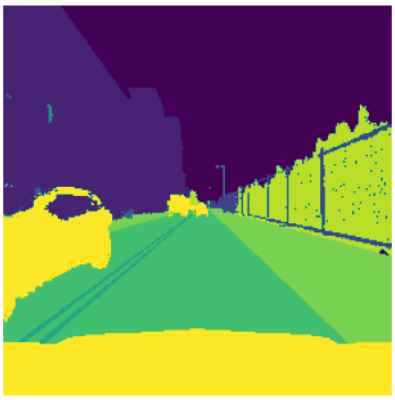
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 42ms/step

Input Image



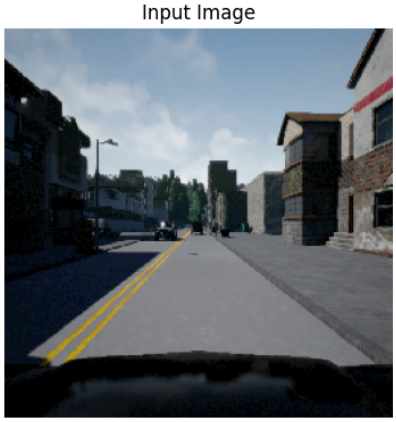
True Mask

Predicted Mask



1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 30ms/step

Input Image



True Mask

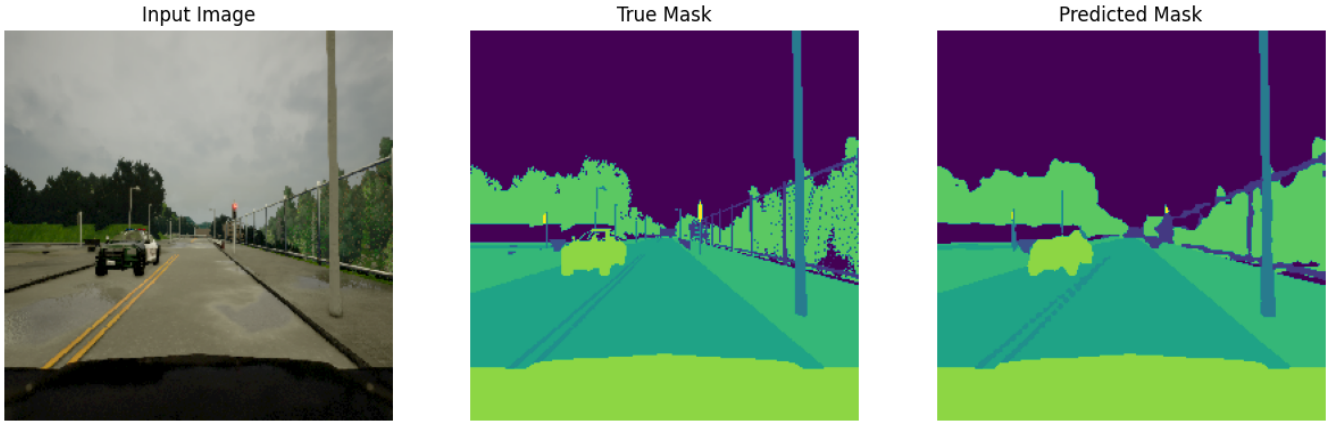
Predicted Mask

In [46]:

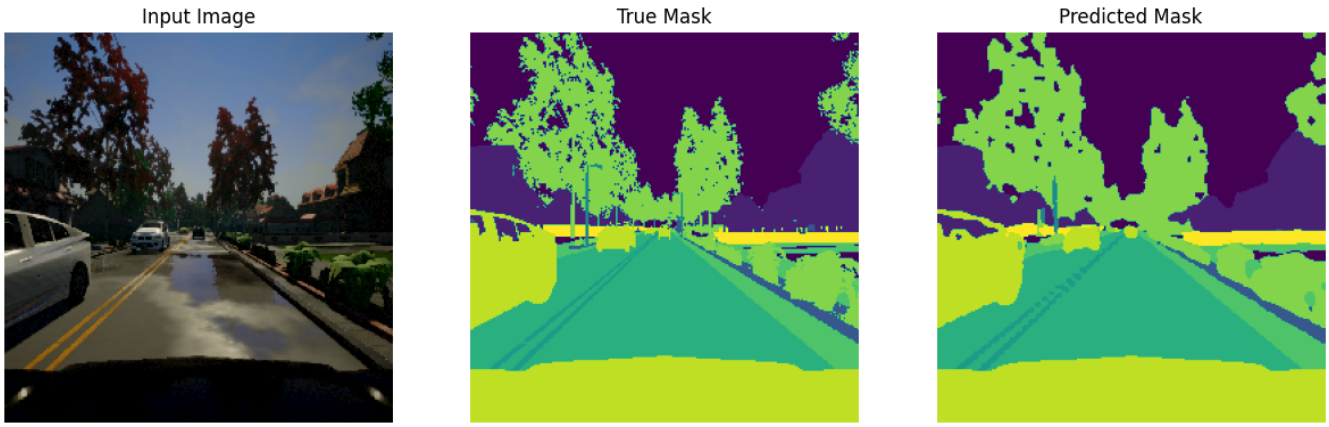
```
# deeplab
show_predictions(train_dataset, 3, deeplab_model1)
show_predictions(val_dataset, 3, deeplab_model1)
show_predictions(test_dataset, 3, deeplab_model1)
```

1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 54ms/step

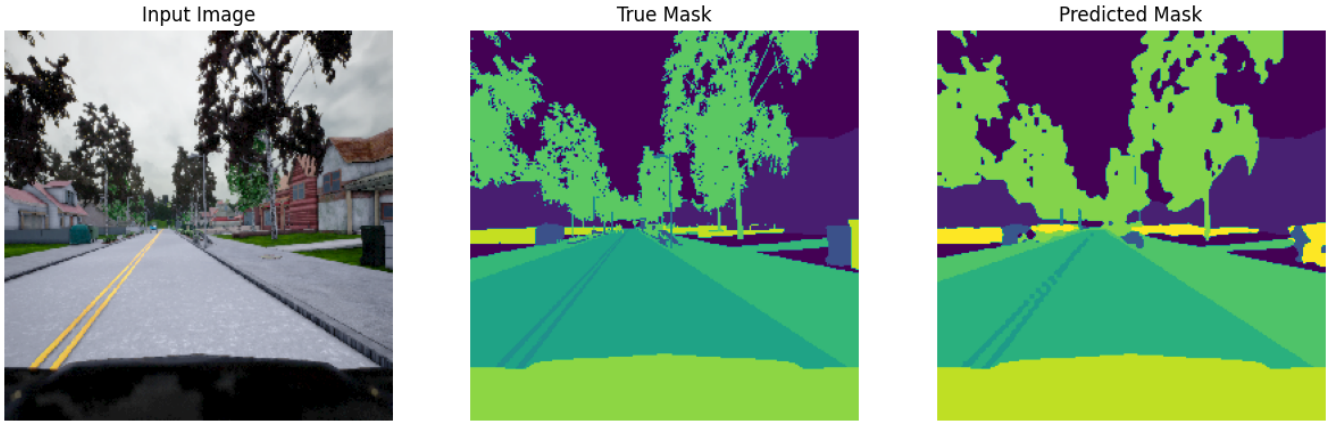
1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 60ms/step



1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 34ms/step

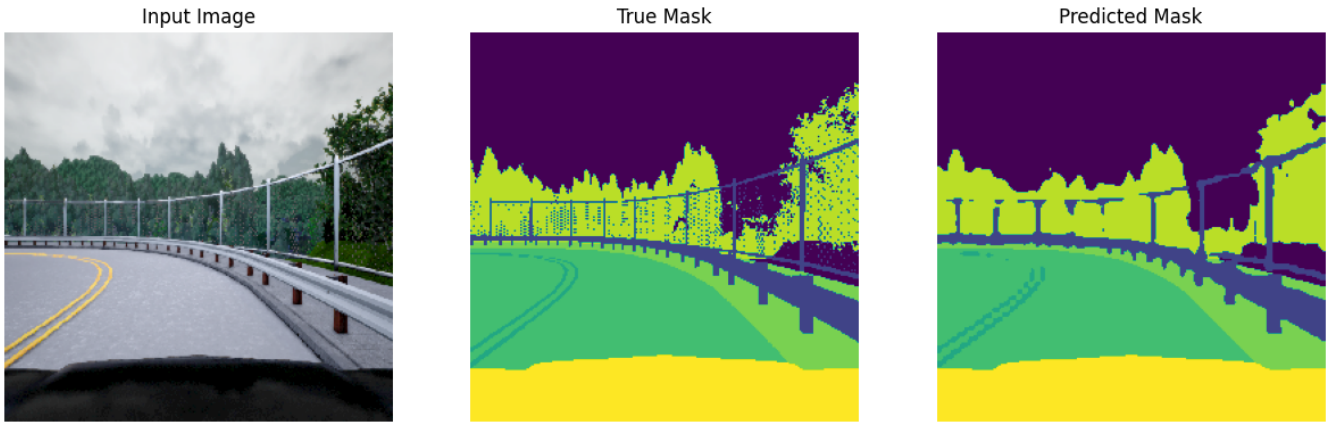



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
```

- ### Predict on New Dataset

In [47]:
```
from pathlib import Path

def load_and_preprocess_image(image_path, mask_path):
    image = tf.io.read_file(image_path)
    image = tf.image.decode_png(image, channels=3)
    image = tf.image.resize(image, [256, 256], method='nearest')
    image = tf.image.convert_image_dtype(image, tf.float32)

    mask = tf.io.read_file(mask_path)
    mask = tf.image.decode_png(mask, channels=3)
    mask = tf.image.resize(mask, [256, 256], method='nearest')
    mask = tf.math.reduce_max(mask, axis=-1, keepdims=True)

    return image, mask

# Paths to the new dataset
IMAGE_PATH = Path("/kaggle/input/semantic-segmentation-car-driving/x-rgb")
MASK_PATH = Path("/kaggle/input/semantic-segmentation-car-driving/y-mask")
new_images_paths = sorted([str(x) for x in IMAGE_PATH.glob("*.png")])
new_masks_paths = sorted([str(x) for x in MASK_PATH.glob("*.png")])

# Creating the new dataset
new_dataset = tf.data.Dataset.from_tensor_slices((new_images_paths, new_masks_paths))
new_dataset = new_dataset.map(load_and_preprocess_image)
new_dataset_batched = new_dataset.batch(1)
```

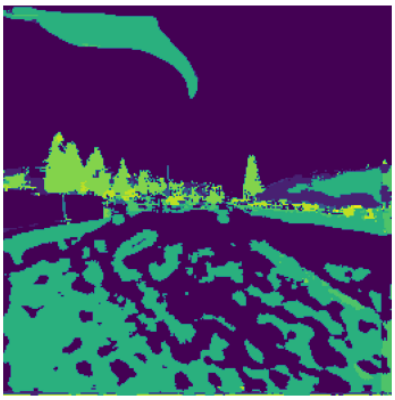
In [48]:
```
# fcn
show_predictions(new_dataset_batched, 5, fcn_model1)
```

1/1 **1s** 1s/step

Input Image

True Mask

Predicted Mask

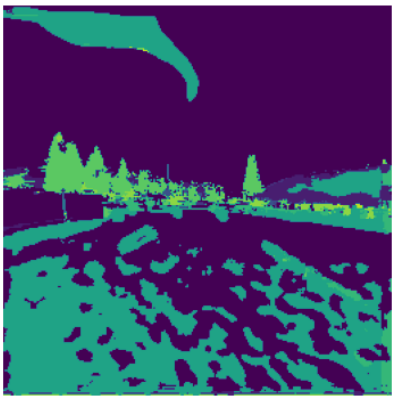


1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 18ms/step

Input Image

True Mask

Predicted Mask

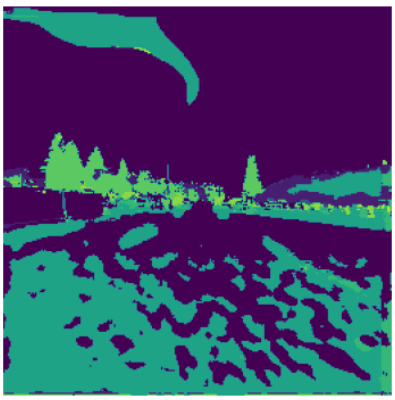


1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 18ms/step

Input Image

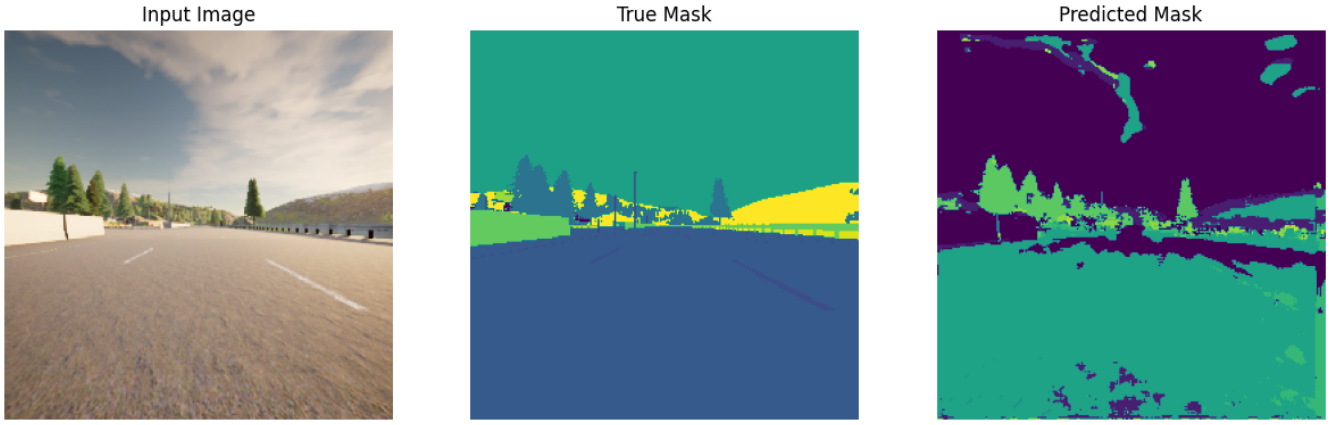
True Mask

Predicted Mask



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
```



In [49]:

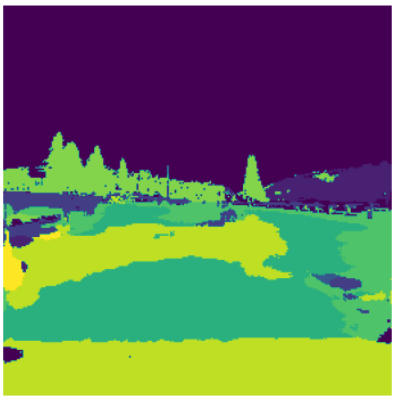
```
# unet
show_predictions(new_dataset_batched, 5, unet_model1)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```

Input Image

True Mask

Predicted Mask

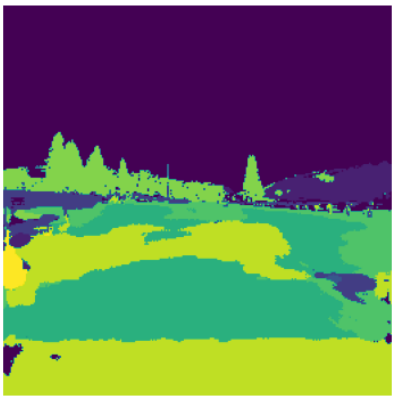


1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 19ms/step

Input Image

True Mask

Predicted Mask

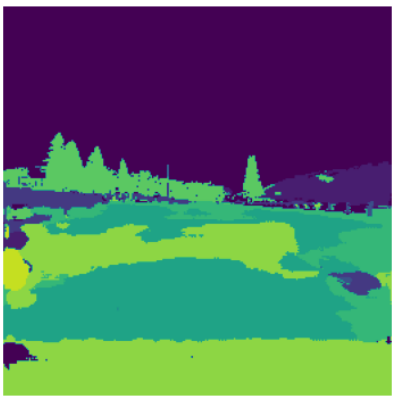


1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 19ms/step

Input Image

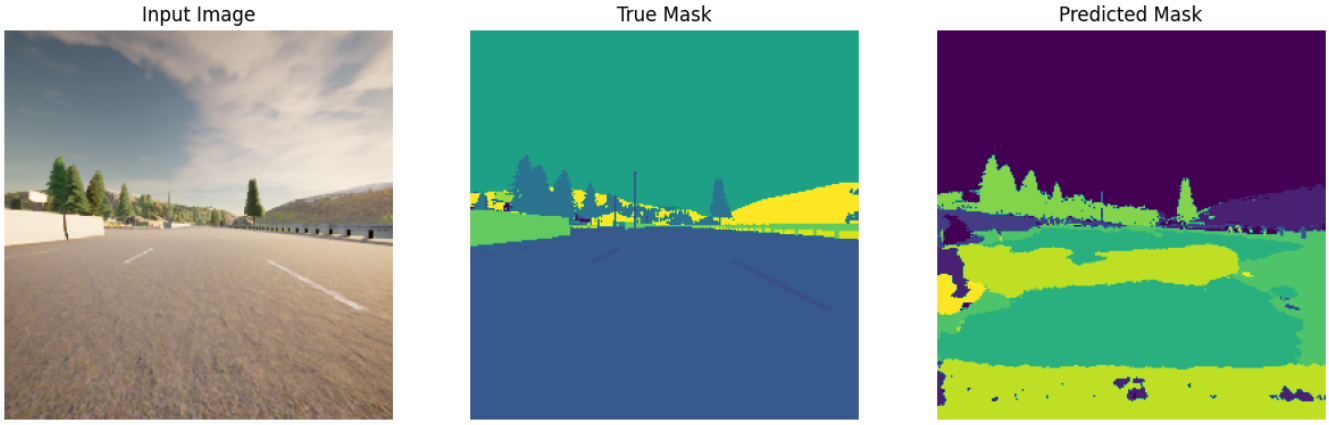
True Mask

Predicted Mask



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
```

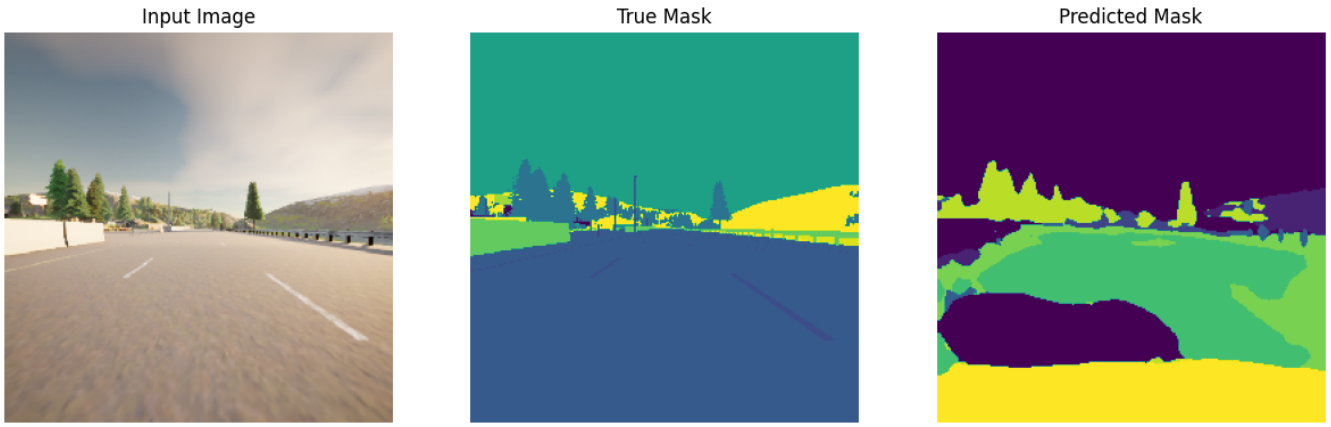

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
```



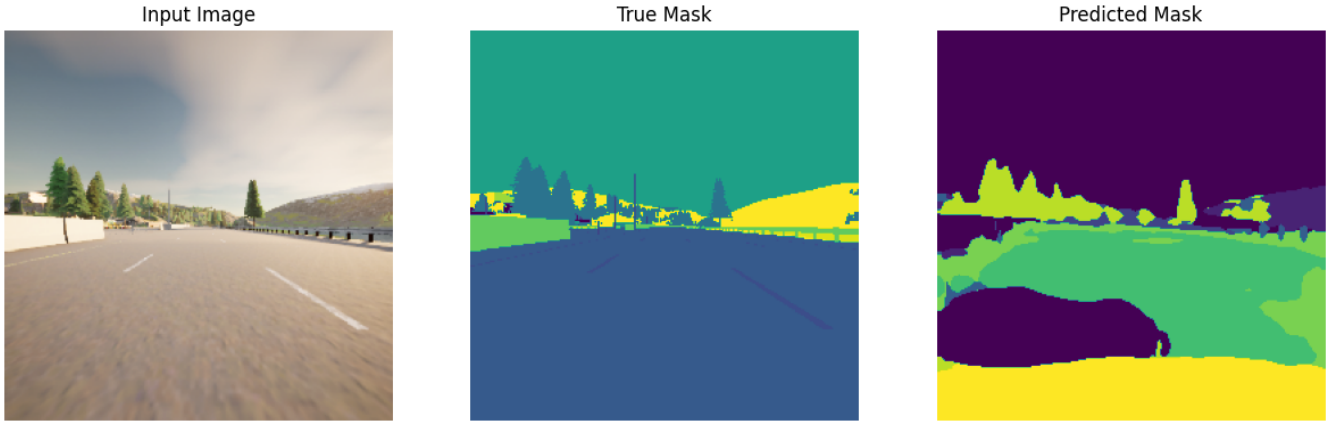
In [50]:

```
#deeplab
show_predictions(new_dataset_batched, 5, deeplab_model1)
```

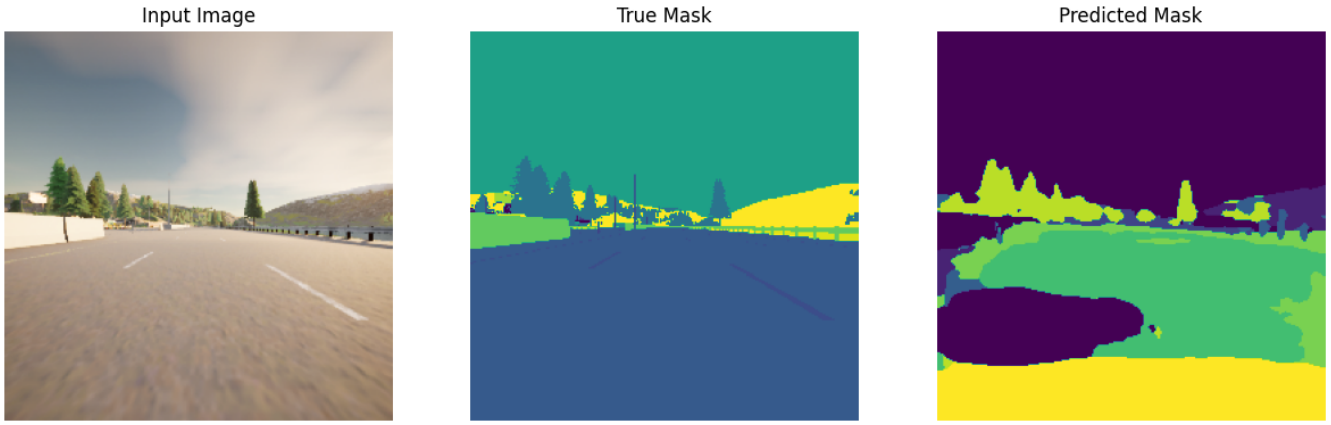
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
```



1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 22ms/step

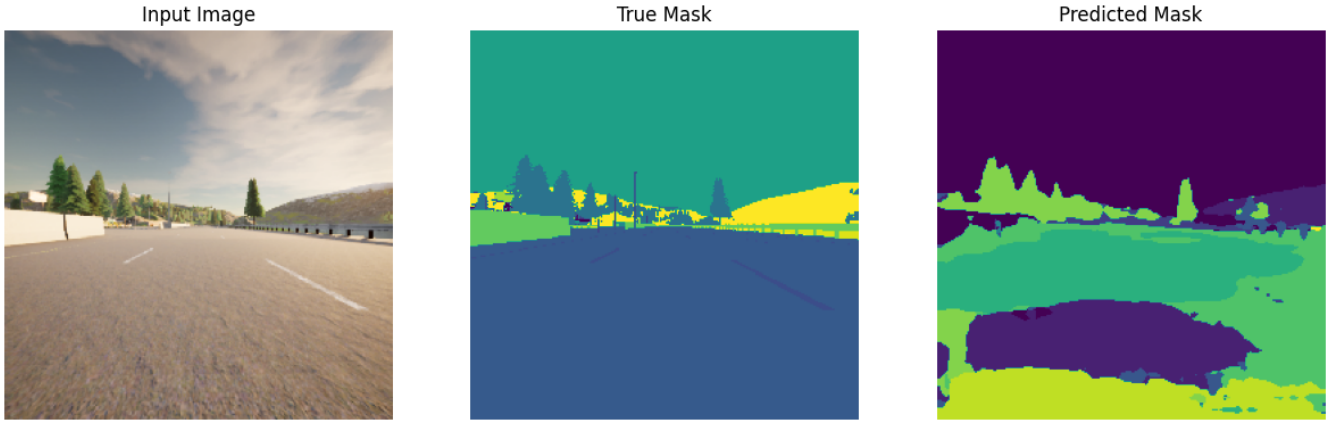


1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 22ms/step



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```



6. Save Model

In [51]:

```
from tensorflow.keras.models import load_model
fcn_model1.save('fcn1-image-segmentation-model.h5')

unet_model1.save('unet1-image-segmentation-model.h5')

deeplab_model1.save('deeplab1-image-segmentation-model.h5')
```