u-net-v2

November 8, 2024

U-NET V2 ARCHITECTURE BUILDING

LOADING MODULES

[8]:
```python
import os
import time
import random
import pathlib
import itertools
from glob import glob
from tqdm import tqdm_notebook, tnrange

# import data handling tools
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt
%matplotlib inline
from skimage.color import rgb2gray
from skimage.morphology import label
from skimage.transform import resize
from sklearn.model_selection import train_test_split
from skimage.io import imread, imshow, concatenate_images

# import Deep learning Libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.models import Model, load_model, save_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.layers import Input, Activation, BatchNormalization,␣
↪Dropout, Lambda, Conv2D, Conv2DTranspose, MaxPooling2D, concatenate

# Ignore Warnings
```

```
import warnings
warnings.filterwarnings("ignore")

print ('modules loaded')
```

```
modules loaded
```

SPLITTING THE DATAFRAME

[9]:
```
# function to create dataframe
def create_df(data_dir):
    images_paths = []
    masks_paths = glob(f'{data_dir}/*/*_mask*')

    for i in masks_paths:
        images_paths.append(i.replace('_mask', ''))

    df = pd.DataFrame(data= {'images_paths': images_paths, 'masks_paths':␣
 ↪masks_paths})

    return df

# Function to split dataframe into train, valid, test
def split_df(df):
    # create train_df
    train_df, dummy_df = train_test_split(df, train_size= 0.8)

    # create valid_df and test_df
    valid_df, test_df = train_test_split(dummy_df, train_size= 0.5)

    return train_df, valid_df, test_df
```

[10]:
```
def create_gens(df, aug_dict):
    img_size = (256, 256)
    batch_size = 40


    img_gen = ImageDataGenerator(**aug_dict)
    msk_gen = ImageDataGenerator(**aug_dict)

    # Create general generator
    image_gen = img_gen.flow_from_dataframe(df, x_col='images_paths',␣
 ↪class_mode=None, color_mode='rgb', target_size=img_size,
                                        batch_size=batch_size,␣
 ↪save_to_dir=None, save_prefix='image', seed=1)

    mask_gen = msk_gen.flow_from_dataframe(df, x_col='masks_paths',␣
 ↪class_mode=None, color_mode='grayscale', target_size=img_size,
                                                    batch_size=batch_size,␣
↪save_to_dir=None, save_prefix= 'mask', seed=1)

    gen = zip(image_gen, mask_gen)

    for (img, msk) in gen:
        img = img / 255
        msk = msk / 255
        msk[msk > 0.5] = 1
        msk[msk <= 0.5] = 0

        yield (img, msk)
```

ARCHITECTURE BUILDING

[11]:
```
def ChannelAttention(x, ratio=8):
    channel = x.shape[-1]
    avg_pool = GlobalAveragePooling2D()(x)
    avg_pool = Dense(channel // ratio, activation='relu')(avg_pool)
    avg_pool = Dense(channel, activation='sigmoid')(avg_pool)

    return Multiply()([x, avg_pool])

# import tensorflow.keras.backend as K
# from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import Conv2D, Lambda

# Modified SpatialAttention function using Keras layers only
def SpatialAttention(x):
    # Calculate average and max pooling along the channel axis, using Keras␣
↪layers
    avg_pool = Lambda(lambda y: K.mean(y, axis=-1, keepdims=True))(x)
    max_pool = Lambda(lambda y: K.max(y, axis=-1, keepdims=True))(x)

    # Concatenate the pooled features along the channel axis
    concat = concatenate([avg_pool, max_pool], axis=-1)

    # Convolution layer to create the attention map
    attention = Conv2D(1, (7, 7), padding="same", activation="sigmoid")(concat)

    # Multiply attention map with the input feature map
    return tf.keras.layers.multiply([x, attention])



# SDI (Semantics and Detail Infusion) Module
```

```python
from tensorflow.keras.layers import UpSampling2D, MaxPooling2D, Conv2D

# Updated SDI (Semantics and Detail Infusion) Module with matching dimensions
def SDI_module(x):
    x_upsample = UpSampling2D()(x)  # Upsampling the feature maps
    x_downsample = MaxPooling2D()(x)  # Downsampling the feature maps
    identity_map = x  # Identity map (preserve original detail)

    # Resize upsampled and downsampled features to match identity_map dimensions
    x_upsample = Conv2D(x.shape[-1], (1, 1),␣
↪padding='same')(UpSampling2D(size=(2, 2))(x_downsample))  # Match dimensions
    x_downsample = Conv2D(x.shape[-1], (1, 1),␣
↪padding='same')(UpSampling2D(size=(2, 2))(x_downsample))  # Match dimensions

    # Combine all three maps with attention mechanisms
    fused = concatenate([x_upsample, x_downsample, identity_map], axis=-1)
    fused = ChannelAttention(fused)
    fused = SpatialAttention(fused)

    return fused


# SmoothConv: A smoothing convolutional layer
def SmoothConv(x):
    return Conv2D(filters=x.shape[-1], kernel_size=(3, 3), padding="same")(x)

# U-Net v2 Architecture
def unet_v2(input_size=(256, 256, 3)):
    inputs = Input(input_size)

    # First DownConvolution / Encoder Leg
    conv1 = Conv2D(64, (3, 3), padding="same")(inputs)
    bn1 = Activation("relu")(conv1)
    conv1 = Conv2D(64, (3, 3), padding="same")(bn1)
    bn1 = BatchNormalization(axis=3)(conv1)
    bn1 = Activation("relu")(bn1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(bn1)

    conv2 = Conv2D(128, (3, 3), padding="same")(pool1)
    bn2 = Activation("relu")(conv2)
    conv2 = Conv2D(128, (3, 3), padding="same")(bn2)
    bn2 = BatchNormalization(axis=3)(conv2)
    bn2 = Activation("relu")(bn2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(bn2)

    conv3 = Conv2D(256, (3, 3), padding="same")(pool2)
    bn3 = Activation("relu")(conv3)
    conv3 = Conv2D(256, (3, 3), padding="same")(bn3)
    bn3 = BatchNormalization(axis=3)(conv3)
    bn3 = Activation("relu")(bn3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(bn3)

    conv4 = Conv2D(512, (3, 3), padding="same")(pool3)
    bn4 = Activation("relu")(conv4)
    conv4 = Conv2D(512, (3, 3), padding="same")(bn4)
    bn4 = BatchNormalization(axis=3)(conv4)
    bn4 = Activation("relu")(bn4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(bn4)

    # Bottleneck
    conv5 = Conv2D(1024, (3, 3), padding="same")(pool4)
    bn5 = Activation("relu")(conv5)
    conv5 = Conv2D(1024, (3, 3), padding="same")(bn5)
    bn5 = BatchNormalization(axis=3)(conv5)
    bn5 = Activation("relu")(bn5)

    # Decoder Leg / UpConvolution with SDI and Attention
    up6 = concatenate([Conv2DTranspose(512, (2, 2), strides=(2, 2),␣
↪padding="same")(bn5), SDI_module(conv4)], axis=3)
    conv6 = SmoothConv(up6)
    bn6 = Activation("relu")(conv6)
    conv6 = Conv2D(512, (3, 3), padding="same")(bn6)
    bn6 = BatchNormalization(axis=3)(conv6)
    bn6 = Activation("relu")(bn6)

    up7 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2),␣
↪padding="same")(bn6), SDI_module(conv3)], axis=3)
    conv7 = SmoothConv(up7)
    bn7 = Activation("relu")(conv7)
    conv7 = Conv2D(256, (3, 3), padding="same")(bn7)
    bn7 = BatchNormalization(axis=3)(conv7)
    bn7 = Activation("relu")(bn7)

    up8 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2),␣
↪padding="same")(bn7), SDI_module(conv2)], axis=3)
    conv8 = SmoothConv(up8)
    bn8 = Activation("relu")(conv8)
    conv8 = Conv2D(128, (3, 3), padding="same")(bn8)
    bn8 = BatchNormalization(axis=3)(conv8)
    bn8 = Activation("relu")(bn8)

    up9 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2),␣
↪padding="same")(bn8), SDI_module(conv1)], axis=3)
    conv9 = SmoothConv(up9)
    bn9 = Activation("relu")(conv9)
    conv9 = Conv2D(64, (3, 3), padding="same")(bn9)
    bn9 = BatchNormalization(axis=3)(conv9)
    bn9 = Activation("relu")(bn9)

    # Final Layer
    conv10 = Conv2D(1, (1, 1), activation="sigmoid")(bn9)

    return Model(inputs=[inputs], outputs=[conv10])
```

DICE COEFFICIENT

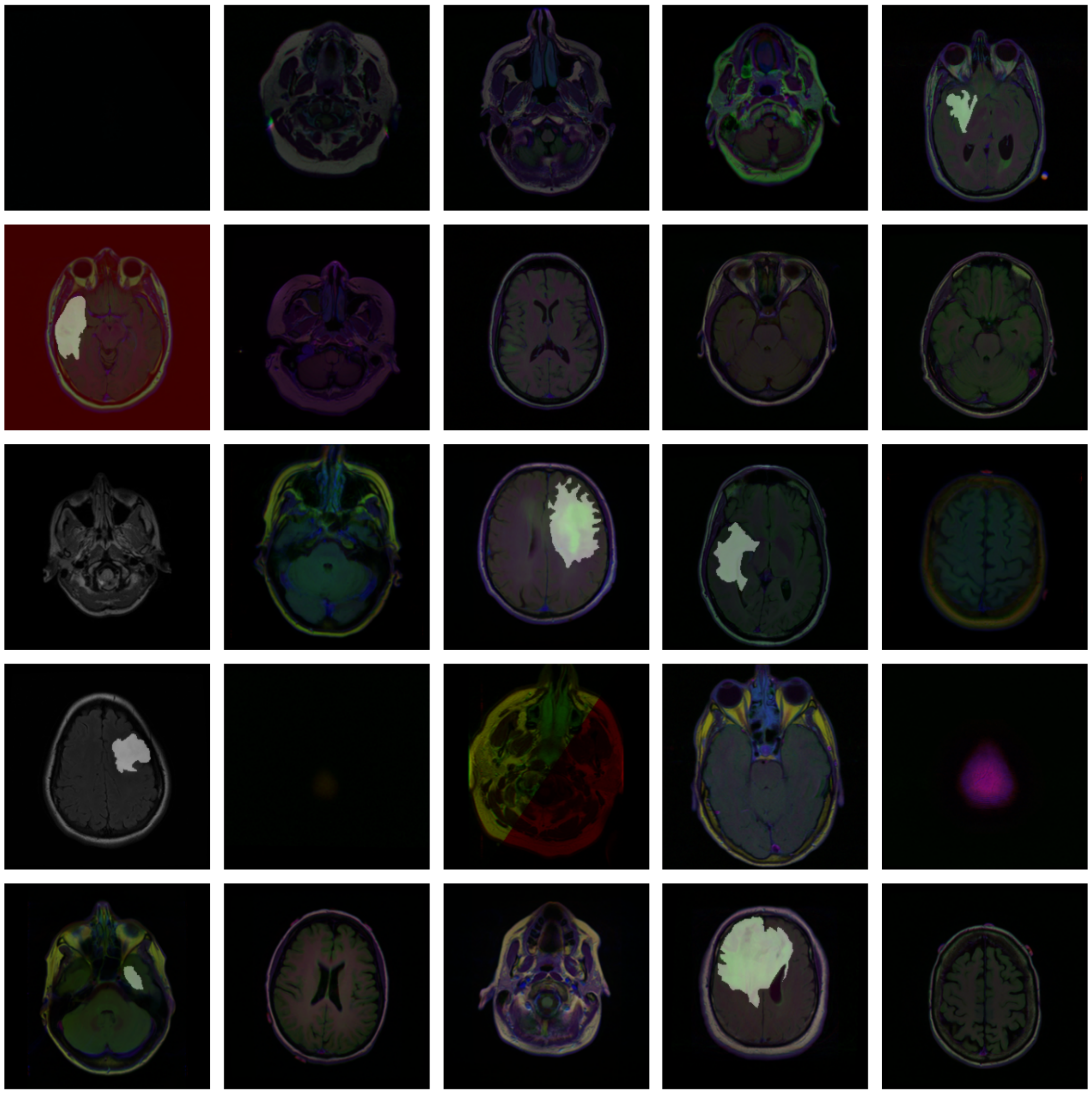
[12]:
```
# function to create dice coefficient
def dice_coef(y_true, y_pred, smooth=100):
    y_true_flatten = K.flatten(y_true)
    y_pred_flatten = K.flatten(y_pred)

    intersection = K.sum(y_true_flatten * y_pred_flatten)
    union = K.sum(y_true_flatten) + K.sum(y_pred_flatten)
    return (2 * intersection + smooth) / (union + smooth)

# function to create dice loss
def dice_loss(y_true, y_pred, smooth=100):
    return -dice_coef(y_true, y_pred, smooth)
```

PLOTTING

[13]:
```
def show_images(images, masks):
    plt.figure(figsize=(12, 12))
    for i in range(25):
        plt.subplot(5, 5, i+1)
        img_path = images[i]
        mask_path = masks[i]
        # read image and convert it to RGB scale
        image = cv2.imread(img_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        # read mask
        mask = cv2.imread(mask_path)
        # sho image and mask
        plt.imshow(image)
        plt.imshow(mask, alpha=0.4)

        plt.axis('off')

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

DATASET

[14]:
```
data_dir = '/kaggle/input/lgg-mri-segmentation/kaggle_3m'

df = create_df(data_dir)
train_df, valid_df, test_df = split_df(df)


tr_aug_dict = dict(rotation_range=0.2,
                            width_shift_range=0.05,
                            height_shift_range=0.05,
                            shear_range=0.05,
                            zoom_range=0.05,
                            horizontal_flip=True,
                            fill_mode='nearest')


train_gen = create_gens(train_df, aug_dict=tr_aug_dict)
valid_gen = create_gens(valid_df, aug_dict={})
test_gen = create_gens(test_df, aug_dict={})

show_images(list(train_df['images_paths']), list(train_df['masks_paths']))
```



MODEL SUMMARY

```
[15]: model = unet()
      model.compile(Adamax(learning_rate= 0.001), loss= dice_loss, metrics= [␣
       ↪dice_coef])

      model.summary()
```

```
Model: "functional_1"


 Layer (type)            Output Shape            Param #   Connected to
```

```
input_layer           (None, 256, 256,           0  -
(InputLayer)          3)

conv2d (Conv2D)       (None, 256, 256,       1,792  input_layer[0][0]
                      64)

activation            (None, 256, 256,           0  conv2d[0][0]
(Activation)          64)

conv2d_1 (Conv2D)     (None, 256, 256,      36,928  activation[0][0]
                      64)

batch_normalization   (None, 256, 256,         256  conv2d_1[0][0]
(BatchNormalizatio…   64)

activation_1          (None, 256, 256,           0  batch_normalizat…
(Activation)          64)

max_pooling2d         (None, 128, 128,           0  activation_1[0][…
(MaxPooling2D)        64)

conv2d_2 (Conv2D)     (None, 128, 128,      73,856  max_pooling2d[0]…
                      128)

activation_2          (None, 128, 128,           0  conv2d_2[0][0]
(Activation)          128)

conv2d_3 (Conv2D)     (None, 128, 128,     147,584  activation_2[0][…
                      128)

batch_normalizatio…   (None, 128, 128,         512  conv2d_3[0][0]
(BatchNormalizatio…   128)

activation_3          (None, 128, 128,           0  batch_normalizat…
(Activation)          128)

max_pooling2d_1       (None, 64, 64,             0  activation_3[0][…
(MaxPooling2D)        128)

conv2d_4 (Conv2D)     (None, 64, 64,       295,168  max_pooling2d_1[…
                      256)

activation_4          (None, 64, 64,             0  conv2d_4[0][0]
(Activation)          256)

conv2d_5 (Conv2D)     (None, 64, 64,       590,080  activation_4[0][…
                      256)
```

```
batch_normalizatio…    (None, 64, 64,          1,024  conv2d_5[0][0]
(BatchNormalizatio…    256)

activation_5           (None, 64, 64,              0  batch_normalizat…
(Activation)           256)

max_pooling2d_2        (None, 32, 32,              0  activation_5[0][…
(MaxPooling2D)         256)

conv2d_6 (Conv2D)      (None, 32, 32,      1,180,160  max_pooling2d_2[…
                       512)

activation_6           (None, 32, 32,              0  conv2d_6[0][0]
(Activation)           512)

conv2d_7 (Conv2D)      (None, 32, 32,      2,359,808  activation_6[0][…
                       512)

batch_normalizatio…    (None, 32, 32,          2,048  conv2d_7[0][0]
(BatchNormalizatio…    512)

activation_7           (None, 32, 32,              0  batch_normalizat…
(Activation)           512)

max_pooling2d_3        (None, 16, 16,              0  activation_7[0][…
(MaxPooling2D)         512)

conv2d_8 (Conv2D)      (None, 16, 16,      4,719,616  max_pooling2d_3[…
                       1024)

activation_8           (None, 16, 16,              0  conv2d_8[0][0]
(Activation)           1024)

conv2d_9 (Conv2D)      (None, 16, 16,      9,438,208  activation_8[0][…
                       1024)

batch_normalizatio…    (None, 16, 16,          4,096  conv2d_9[0][0]
(BatchNormalizatio…    1024)

activation_9           (None, 16, 16,              0  batch_normalizat…
(Activation)           1024)

conv2d_transpose       (None, 32, 32,      2,097,664  activation_9[0][…
(Conv2DTranspose)      512)

concatenate            (None, 32, 32,              0  conv2d_transpose…
(Concatenate)          1024)                          conv2d_7[0][0]
```

```
conv2d_10 (Conv2D)      (None, 32, 32,       4,719,104  concatenate[0][0]
                        512)

activation_10           (None, 32, 32,               0  conv2d_10[0][0]
(Activation)            512)

conv2d_11 (Conv2D)      (None, 32, 32,       2,359,808  activation_10[0]…
                        512)

batch_normalizatio…     (None, 32, 32,           2,048  conv2d_11[0][0]
(BatchNormalizatio…     512)

activation_11           (None, 32, 32,               0  batch_normalizat…
(Activation)            512)

conv2d_transpose_1      (None, 64, 64,         524,544  activation_11[0]…
(Conv2DTranspose)       256)

concatenate_1           (None, 64, 64,               0  conv2d_transpose…
(Concatenate)           512)                            conv2d_5[0][0]

conv2d_12 (Conv2D)      (None, 64, 64,       1,179,904  concatenate_1[0]…
                        256)

activation_12           (None, 64, 64,               0  conv2d_12[0][0]
(Activation)            256)

conv2d_13 (Conv2D)      (None, 64, 64,         590,080  activation_12[0]…
                        256)

batch_normalizatio…     (None, 64, 64,           1,024  conv2d_13[0][0]
(BatchNormalizatio…     256)

activation_13           (None, 64, 64,               0  batch_normalizat…
(Activation)            256)

conv2d_transpose_2      (None, 128, 128,       131,200  activation_13[0]…
(Conv2DTranspose)       128)

concatenate_2           (None, 128, 128,             0  conv2d_transpose…
(Concatenate)           256)                            conv2d_3[0][0]

conv2d_14 (Conv2D)      (None, 128, 128,       295,040  concatenate_2[0]…
                        128)

activation_14           (None, 128, 128,             0  conv2d_14[0][0]
(Activation)            128)
```

```
conv2d_15 (Conv2D)       (None, 128, 128,        147,584   activation_14[0]…
                         128)

batch_normalizatio…      (None, 128, 128,            512   conv2d_15[0][0]
(BatchNormalizatio…      128)

activation_15            (None, 128, 128,              0   batch_normalizat…
(Activation)             128)

conv2d_transpose_3       (None, 256, 256,         32,832   activation_15[0]…
(Conv2DTranspose)        64)

concatenate_3            (None, 256, 256,              0   conv2d_transpose…
(Concatenate)            128)                              conv2d_1[0][0]

conv2d_16 (Conv2D)       (None, 256, 256,         73,792   concatenate_3[0]…
                         64)

activation_16            (None, 256, 256,              0   conv2d_16[0][0]
(Activation)             64)

conv2d_17 (Conv2D)       (None, 256, 256,         36,928   activation_16[0]…
                         64)

batch_normalizatio…      (None, 256, 256,            256   conv2d_17[0][0]
(BatchNormalizatio…      64)

activation_17            (None, 256, 256,              0   batch_normalizat…
(Activation)             64)

conv2d_18 (Conv2D)       (None, 256, 256,             65   activation_17[0]…
                         1)


Total params: 31,043,521 (118.42 MB)

Trainable params: 31,037,633 (118.40 MB)

Non-trainable params: 5,888 (23.00 KB)
```

TRAINING

[16]:
```
import math
from keras.callbacks import ModelCheckpoint

epochs = 25
```

```
batch_size = 40
callbacks = [ModelCheckpoint('/kaggle/working/unet.keras', verbose=0,␣
 ↪save_best_only=True)]

history = model.fit(train_gen,
                    steps_per_epoch=math.ceil(len(train_df) / batch_size),
                    epochs=epochs,
                    verbose=1,
                    callbacks=callbacks,
                    validation_data=valid_gen,
                    validation_steps=math.ceil(len(valid_df) / batch_size))
```

```
Found 3143 validated image filenames.
Found 3143 validated image filenames.
Epoch 1/25
```

```
2024-03-27 03:26:47.480555: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng4{k11=2} for conv (f32[40,128,128,128]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,128,128]{3,2,1,0}, f32[128,128,3,3]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardInput", backend_config={"conv_result_sca
le":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:26:47.574930: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
1.094569919s
Trying algorithm eng4{k11=2} for conv (f32[40,128,128,128]{3,2,1,0}, u8[0]{0})
custom-call(f32[40,128,128,128]{3,2,1,0}, f32[128,128,3,3]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardInput", backend_config={"conv_result_sca
le":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:28:06.727656: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[64,64,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,64,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:28:12.031047: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
6.303566872s
Trying algorithm eng0{} for conv (f32[64,64,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,64,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
```

```
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:28:17.098433: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[64,64,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,64,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:28:22.444199: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
6.345852364s
Trying algorithm eng0{} for conv (f32[64,64,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,64,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:29:50.437931: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[128,64,2,2]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,64,256,256]{3,2,1,0}, f32[40,128,128,128]{3,2,1,0}),
window={size=2x2 stride=2x2}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:29:51.082708: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
1.644891291s
Trying algorithm eng0{} for conv (f32[128,64,2,2]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,64,256,256]{3,2,1,0}, f32[40,128,128,128]{3,2,1,0}),
window={size=2x2 stride=2x2}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:29:52.909779: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[128,64,2,2]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,64,256,256]{3,2,1,0}, f32[40,128,128,128]{3,2,1,0}),
window={size=2x2 stride=2x2}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:29:54.028149: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
2.118466171s
Trying algorithm eng0{} for conv (f32[128,64,2,2]{3,2,1,0}, u8[0]{0}) custom-
```

```
call(f32[40,64,256,256]{3,2,1,0}, f32[40,128,128,128]{3,2,1,0}),
window={size=2x2 stride=2x2}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:01.481301: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng20{k2=1,k3=0} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:01.486867: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
1.005676733s
Trying algorithm eng20{k2=1,k3=0} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0})
custom-call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:02.487052: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng1{k2=2,k3=0} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:02.550301: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
1.063361436s
Trying algorithm eng1{k2=2,k3=0} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0})
custom-call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:03.550495: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:15.297811: E
```

```
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
12.747427894s
Trying algorithm eng0{} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:22.232579: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng20{k2=1,k3=0} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:22.246161: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
1.0137615s
Trying algorithm eng20{k2=1,k3=0} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0})
custom-call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:23.246413: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng1{k2=2,k3=0} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:23.316298: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
1.070058532s
Trying algorithm eng1{k2=2,k3=0} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0})
custom-call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:24.316554: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
```

```
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:30:36.557383: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
13.241000245s
Trying algorithm eng0{} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[40,128,256,256]{3,2,1,0}, f32[40,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
W0000 00:00:1711510238.150314      89 hlo_rematerialization.cc:2946] Can't
reduce memory use below 10.84GiB (11641402856 bytes) by rematerialization; only
reduced to 11.76GiB (12623817276 bytes), down from 14.33GiB (15388767268 bytes)
originally
I0000 00:00:1711510243.181085      89 device_compiler.h:186] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.

78/79            1s 2s/step -
dice_coef: 0.1071 - loss: -0.1071

2024-03-27 03:35:35.161772: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[64,64,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[23,64,256,256]{3,2,1,0}, f32[23,64,256,256]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:35:37.510867: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
3.349190802s
Trying algorithm eng0{} for conv (f32[64,64,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[23,64,256,256]{3,2,1,0}, f32[23,64,256,256]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:35:40.969204: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[64,64,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[23,64,256,256]{3,2,1,0}, f32[23,64,256,256]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:35:43.314237: E
```

```
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
3.345188288s
Trying algorithm eng0{} for conv (f32[64,64,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[23,64,256,256]{3,2,1,0}, f32[23,64,256,256]{3,2,1,0}), window={size=3x3
pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:36:43.251431: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[23,128,256,256]{3,2,1,0}, f32[23,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:36:48.895652: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
6.644334534s
Trying algorithm eng0{} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[23,128,256,256]{3,2,1,0}, f32[23,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:36:54.420730: E
external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm
eng0{} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[23,128,256,256]{3,2,1,0}, f32[23,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
2024-03-27 03:37:00.058052: E
external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took
6.637428608s
Trying algorithm eng0{} for conv (f32[64,128,3,3]{3,2,1,0}, u8[0]{0}) custom-
call(f32[23,128,256,256]{3,2,1,0}, f32[23,64,256,256]{3,2,1,0}),
window={size=3x3 pad=1_1x1_1}, dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBackwardFilter", backend_config={"conv_result_sc
ale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0} is
taking a while…
```

```
79/79            0s 5s/step -
dice_coef: 0.1079 - loss: -0.1079Found 393 validated image filenames.
Found 393 validated image filenames.
79/79            886s 6s/step -
dice_coef: 0.1087 - loss: -0.1087 - val_dice_coef: 0.0207 - val_loss: -0.0207
```

```
Epoch 2/25
79/79            160s 2s/step -
dice_coef: 0.3575 - loss: -0.3575 - val_dice_coef: 0.0108 - val_loss: -0.0108
Epoch 3/25
79/79            162s 2s/step -
dice_coef: 0.5777 - loss: -0.5777 - val_dice_coef: 0.2745 - val_loss: -0.2745
Epoch 4/25
79/79            161s 2s/step -
dice_coef: 0.6652 - loss: -0.6652 - val_dice_coef: 0.4538 - val_loss: -0.4538
Epoch 5/25
79/79            161s 2s/step -
dice_coef: 0.7094 - loss: -0.7094 - val_dice_coef: 0.5693 - val_loss: -0.5693
Epoch 6/25
79/79            161s 2s/step -
dice_coef: 0.7283 - loss: -0.7283 - val_dice_coef: 0.6798 - val_loss: -0.6798
Epoch 7/25
79/79            161s 2s/step -
dice_coef: 0.7370 - loss: -0.7370 - val_dice_coef: 0.7237 - val_loss: -0.7237
Epoch 8/25
79/79            161s 2s/step -
dice_coef: 0.7572 - loss: -0.7572 - val_dice_coef: 0.7285 - val_loss: -0.7285
Epoch 9/25
79/79            160s 2s/step -
dice_coef: 0.7747 - loss: -0.7747 - val_dice_coef: 0.7063 - val_loss: -0.7063
Epoch 10/25
79/79            160s 2s/step -
dice_coef: 0.7748 - loss: -0.7748 - val_dice_coef: 0.6966 - val_loss: -0.6966
Epoch 11/25
79/79            161s 2s/step -
dice_coef: 0.8012 - loss: -0.8012 - val_dice_coef: 0.7816 - val_loss: -0.7816
Epoch 12/25
79/79            159s 2s/step -
dice_coef: 0.7868 - loss: -0.7868 - val_dice_coef: 0.7721 - val_loss: -0.7721
Epoch 13/25
79/79            160s 2s/step -
dice_coef: 0.7850 - loss: -0.7850 - val_dice_coef: 0.7731 - val_loss: -0.7731
Epoch 14/25
79/79            161s 2s/step -
dice_coef: 0.8147 - loss: -0.8147 - val_dice_coef: 0.7894 - val_loss: -0.7894
Epoch 15/25
79/79            160s 2s/step -
dice_coef: 0.8198 - loss: -0.8198 - val_dice_coef: 0.6664 - val_loss: -0.6664
Epoch 16/25
79/79            159s 2s/step -
dice_coef: 0.8315 - loss: -0.8315 - val_dice_coef: 0.7883 - val_loss: -0.7883
Epoch 17/25
79/79            160s 2s/step -
dice_coef: 0.8233 - loss: -0.8233 - val_dice_coef: 0.7739 - val_loss: -0.7739
```

```
Epoch 18/25
79/79              159s 2s/step -
dice_coef: 0.8205 - loss: -0.8205 - val_dice_coef: 0.7775 - val_loss: -0.7775
Epoch 19/25
79/79              159s 2s/step -
dice_coef: 0.8255 - loss: -0.8255 - val_dice_coef: 0.7525 - val_loss: -0.7525
Epoch 20/25
79/79              160s 2s/step -
dice_coef: 0.8218 - loss: -0.8218 - val_dice_coef: 0.7826 - val_loss: -0.7826
Epoch 21/25
79/79              161s 2s/step -
dice_coef: 0.8428 - loss: -0.8428 - val_dice_coef: 0.8089 - val_loss: -0.8089
Epoch 22/25
79/79              161s 2s/step -
dice_coef: 0.8481 - loss: -0.8481 - val_dice_coef: 0.8360 - val_loss: -0.8360
Epoch 23/25
79/79              159s 2s/step -
dice_coef: 0.8420 - loss: -0.8420 - val_dice_coef: 0.7910 - val_loss: -0.7910
Epoch 24/25
79/79              159s 2s/step -
dice_coef: 0.8432 - loss: -0.8432 - val_dice_coef: 0.8328 - val_loss: -0.8328
Epoch 25/25
79/79              161s 2s/step -
dice_coef: 0.8712 - loss: -0.8712 - val_dice_coef: 0.8585 - val_loss: -0.8585
```

[17]:
```
model.save("/kaggle/working/model.h5")
```

LOAD MODAL

[ ]:
```
from keras.models import load_model

# Assuming unet.keras is located in the current working directory
model = load_model('/kaggle/working/unet.keras')

# Now you can use the loaded model for inference or further processing
```

DICE SCORE COMPUTATION

[ ]:
```
ts_length = len(test_df)
test_batch_size = max(sorted([ts_length // n for n in range(1, ts_length + 1)␣
  ↪if ts_length%n == 0 and ts_length/n <= 80]))
test_steps = ts_length // test_batch_size

train_score = model.evaluate(train_gen, steps= test_steps, verbose= 1)
valid_score = model.evaluate(valid_gen, steps= test_steps, verbose= 1)
test_score = model.evaluate(test_gen, steps= test_steps, verbose= 1)
```

DICE VALUE

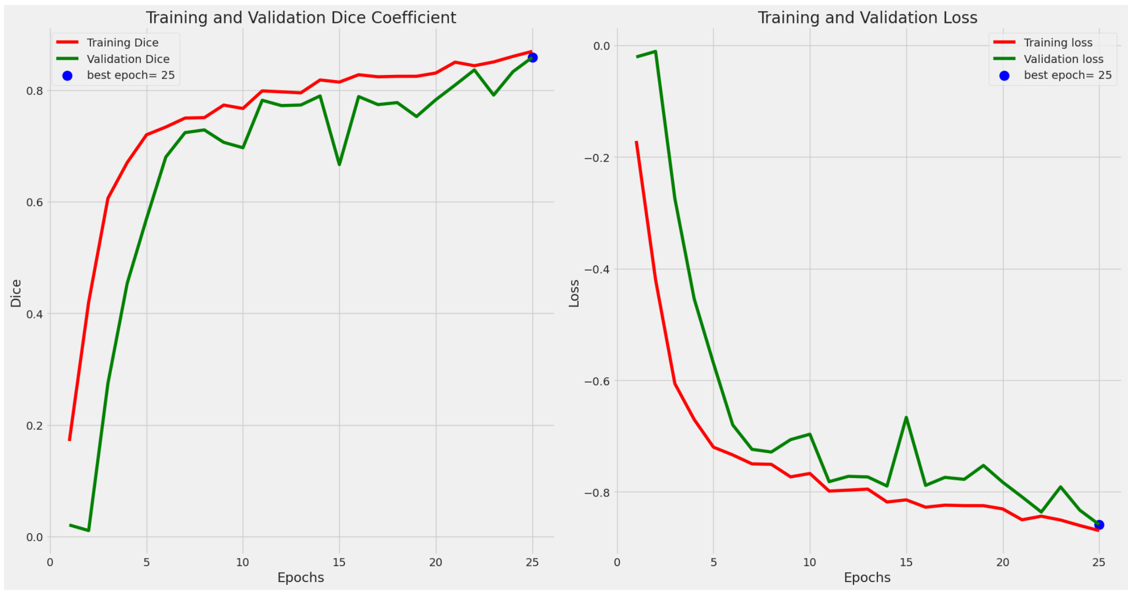
[1]:
```python
print("Train Loss: ", train_score[0])
print("Train Dice: ", train_score[1])
print('-' * 20)

print("Valid Loss: ", valid_score[0])
print("Valid Dice: ", valid_score[1])
print('-' * 20)

print("Test Loss: ", test_score[0])
print("Test Dice: ", test_score[1])
```

```
Training Loss:  0.1224716271181162
Training Dice:  0.8721450567245488
--------------------
Validation Loss: 0.1395732432585654
Validation Dice: 0.8610987663269046
--------------------
Test Loss: 0.130535134252623
Test Dice: 0.870535135269167
```

PLOTTING

[27]:
```python
import numpy as np
import matplotlib.pyplot as plt

def plot_training(hist):
    '''
    This function takes a training model and plots the history of accuracy and␣
 ↪losses with the best epoch in both of them.
    '''

    # Define needed variables
    tr_dice = hist.history['dice_coef']
    tr_loss = hist.history['loss']

    val_dice = hist.history['val_dice_coef']
    val_loss = hist.history['val_loss']

    index_dice = np.argmax(val_dice)  # Corrected line
    dice_highest = val_dice[index_dice]
    index_loss = np.argmin(val_loss)
    val_lowest = val_loss[index_loss]

    Epochs = [i+1 for i in range(len(val_loss))]

    dice_label = f'best epoch= {str(index_dice + 1)}'
    loss_label = f'best epoch= {str(index_loss + 1)}'
    # Plot training history
    plt.figure(figsize=(20, 20))
    plt.style.use('fivethirtyeight')


    # Training Dice
    plt.subplot(2, 2, 3)
    plt.plot(Epochs, tr_dice, 'r', label='Training Dice')
    plt.plot(Epochs, val_dice, 'g', label='Validation Dice')
    plt.scatter(index_dice + 1 , dice_highest, s=150, c='blue',␣
↪label=dice_label)
    plt.title('Training and Validation Dice Coefficient')
    plt.xlabel('Epochs')
    plt.ylabel('Dice')
    plt.legend()

    # Training Loss
    plt.subplot(2, 2, 4)
    plt.plot(Epochs, tr_loss, 'r', label='Training loss')
    plt.plot(Epochs, val_loss, 'g', label='Validation loss')
    plt.scatter(index_loss + 1, val_lowest, s=150, c='blue', label=loss_label)
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

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

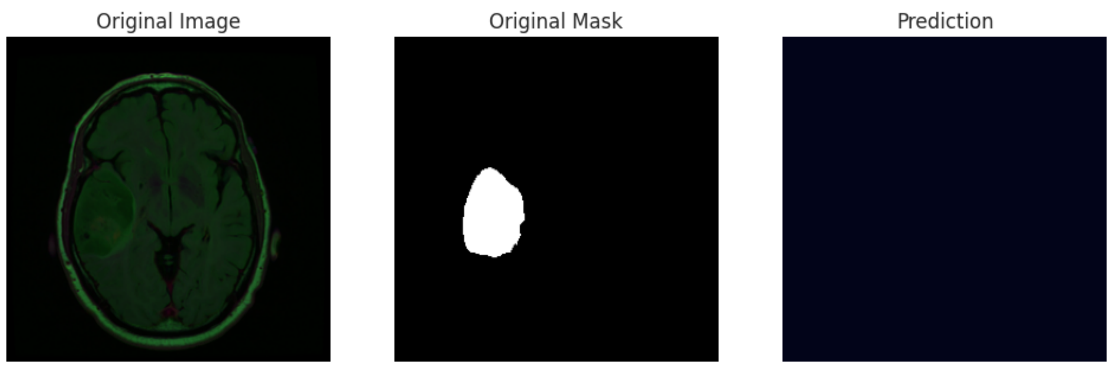
[28]:
```
plot_training(history)
```



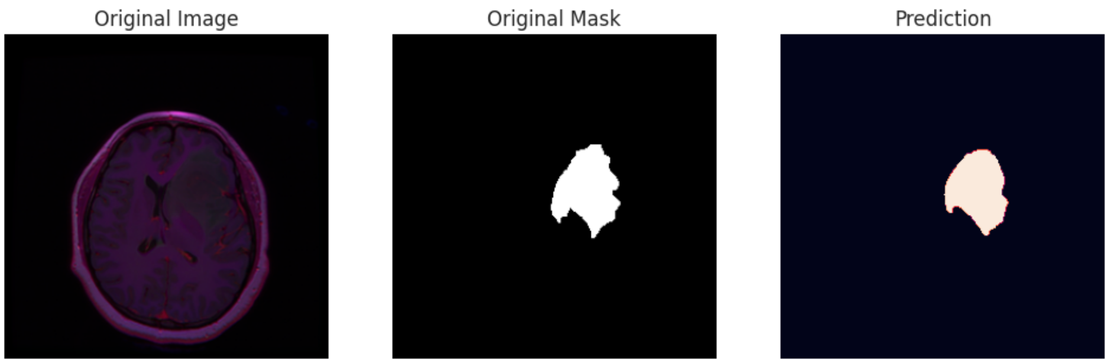
OUTPUT

```
[20]: for _ in range(20):
          index = np.random.randint(1, len(test_df.index))
          img = cv2.imread(test_df['images_paths'].iloc[index])
          img = cv2.resize(img, (256, 256))
          img = img/255
          img = img[np.newaxis, :, :, : ]

          predicted_img = model.predict(img)

          plt.figure(figsize=(12, 12))

          plt.subplot(1, 3, 1)
          plt.imshow(np.squeeze(img))
          plt.axis('off')
          plt.title('Original Image')

          plt.subplot(1, 3, 2)
          plt.imshow(np.squeeze(cv2.imread(test_df['masks_paths'].iloc[index])))
          plt.axis('off')
          plt.title('Original Mask')

          plt.subplot(1, 3, 3)
          plt.imshow(np.squeeze(predicted_img) > 0.5 )
          plt.title('Prediction')
          plt.axis('off')

          plt.show()
```

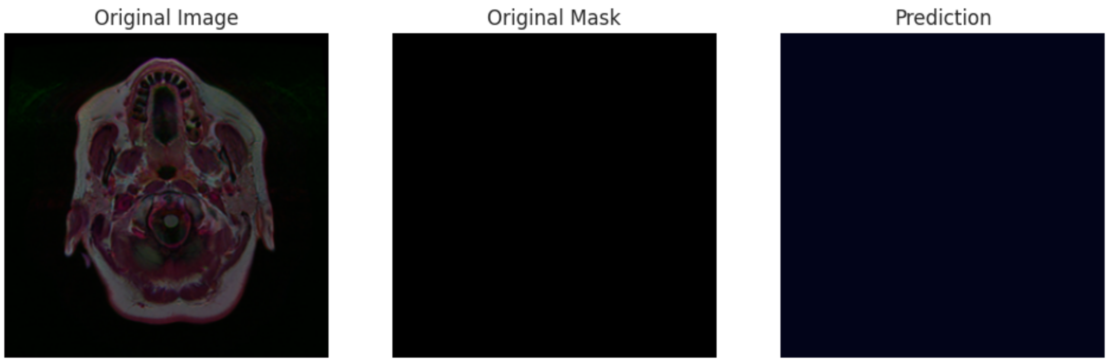
```
1/1                9s 9s/step
```



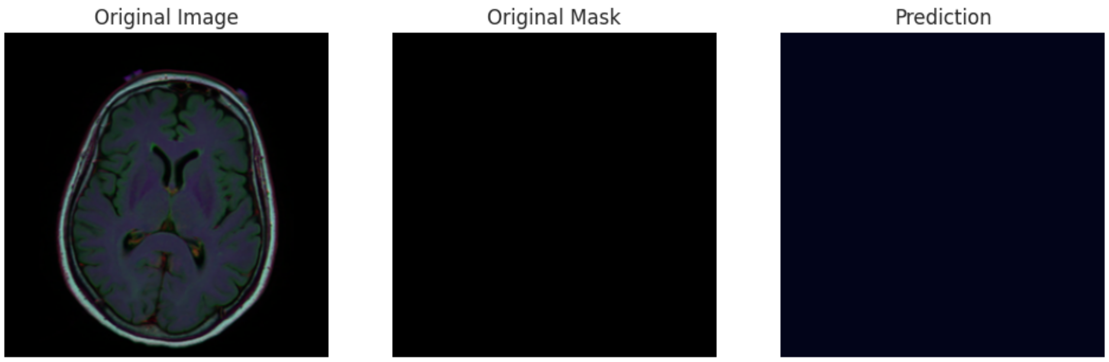
```
1/1                0s 19ms/step
```



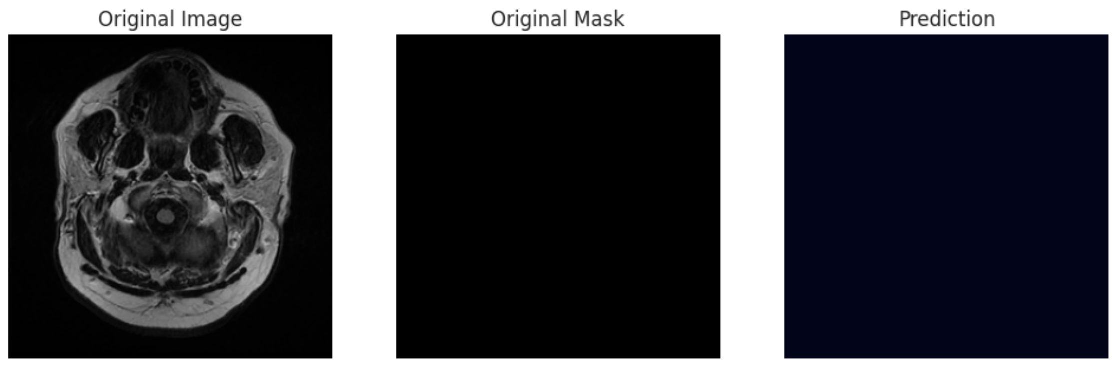
```
1/1                0s 19ms/step
```



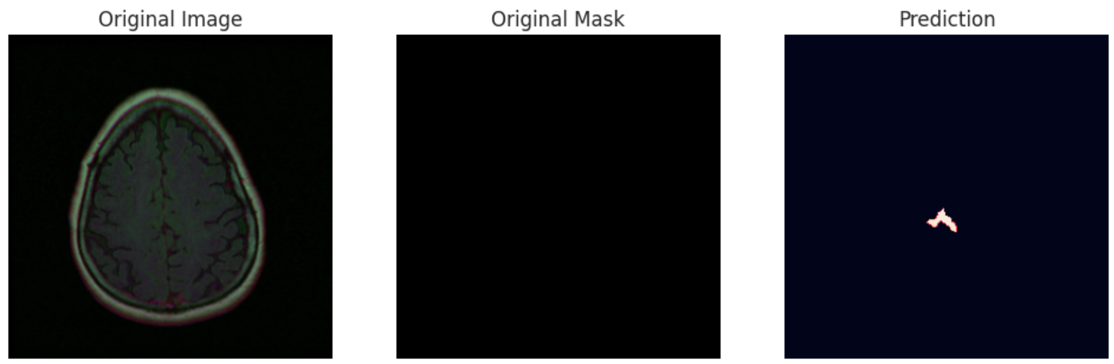
```
1/1                0s 19ms/step
```



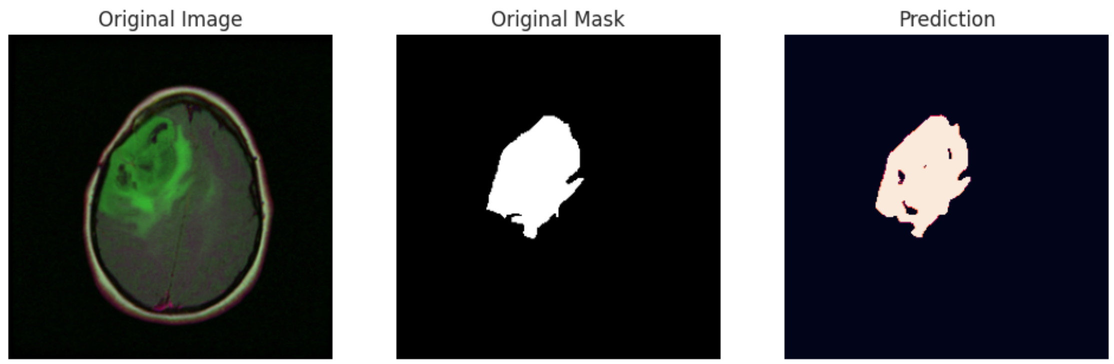
```
1/1                0s 19ms/step
```



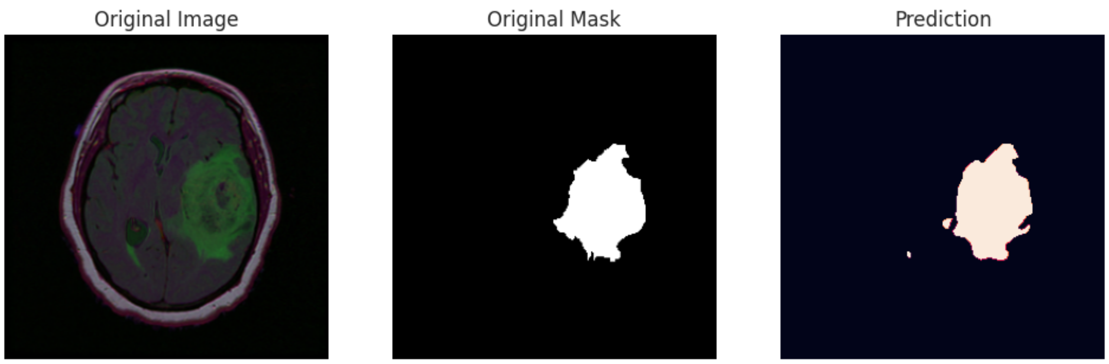
```
1/1                0s 22ms/step
```



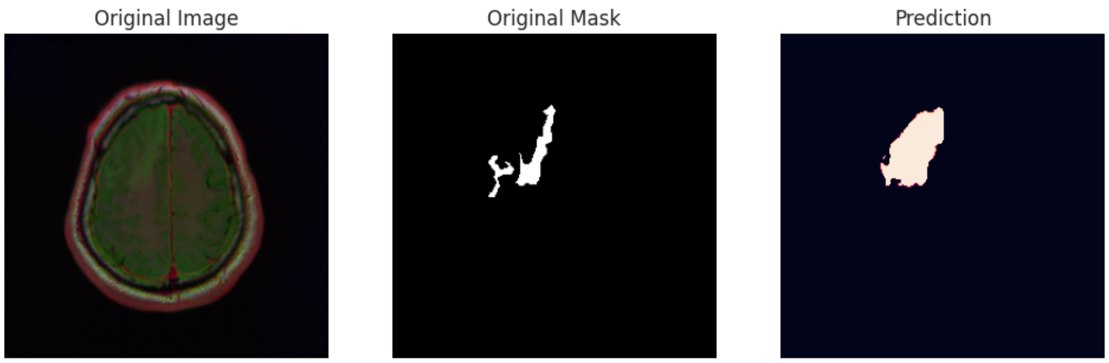
```
1/1                0s 22ms/step
```



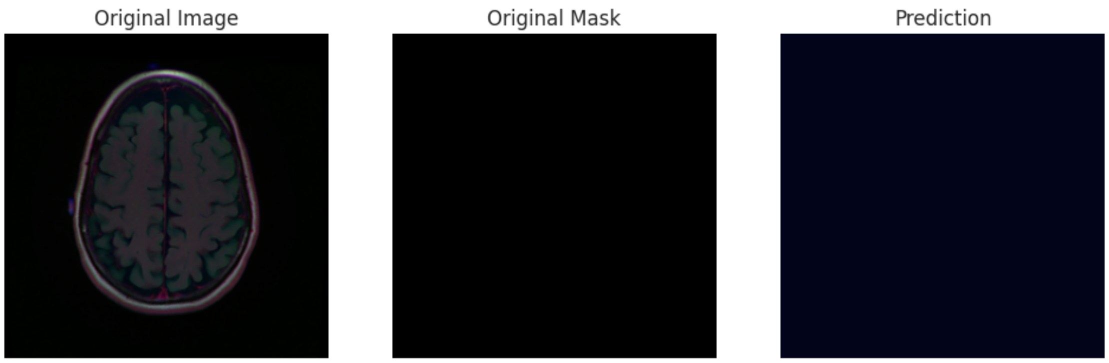
```
1/1                0s 20ms/step
```



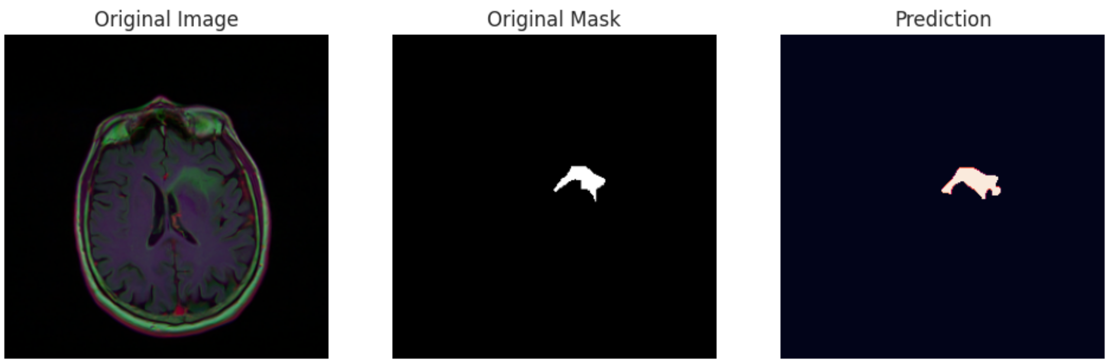
```
1/1                0s 19ms/step
```



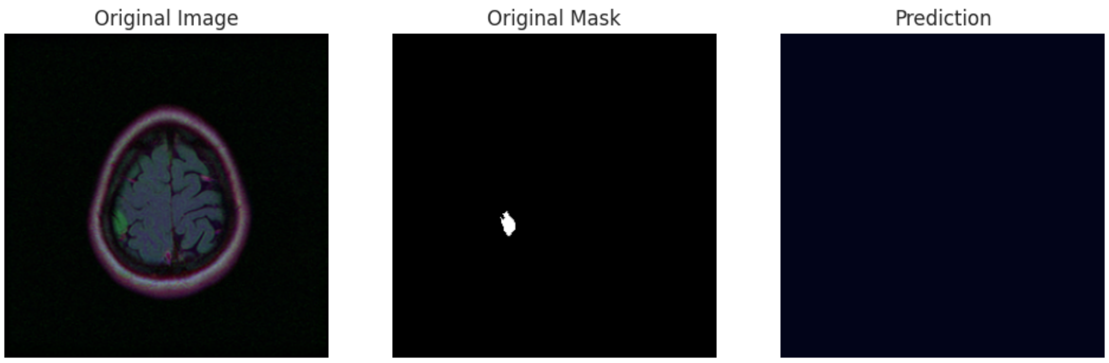
```
1/1                0s 18ms/step
```



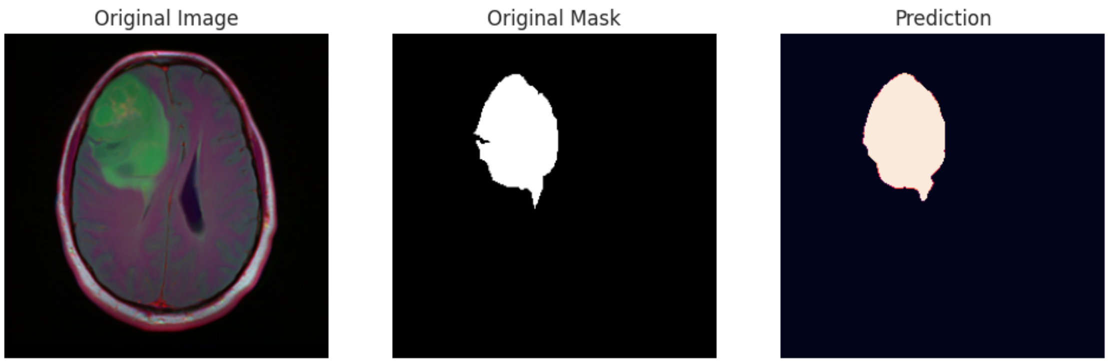
```
1/1                0s 20ms/step
```



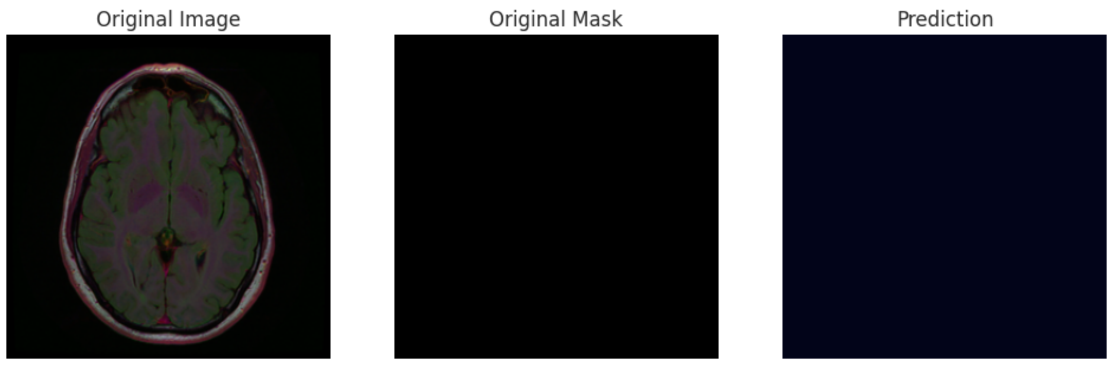
```
1/1                0s 20ms/step
```



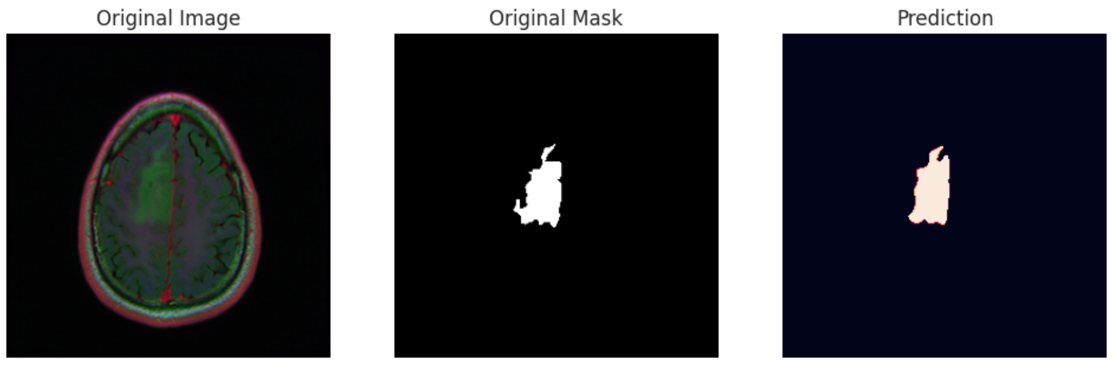
```
1/1                0s 19ms/step
```



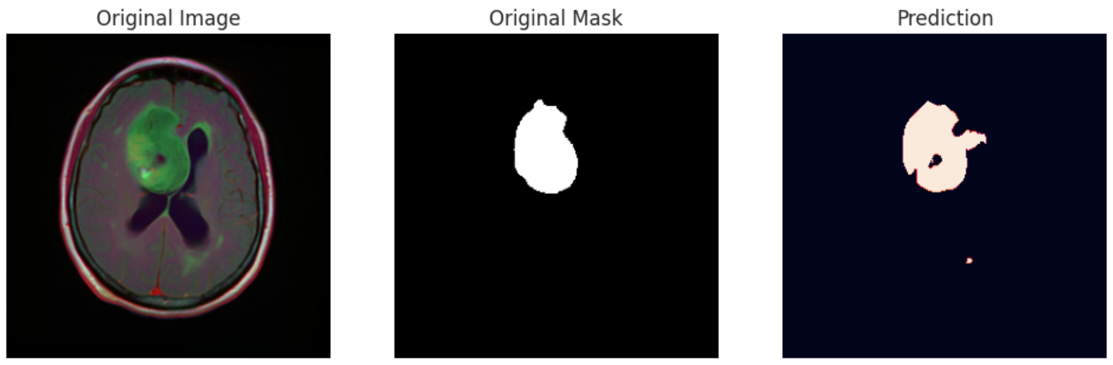
```
1/1                0s 18ms/step
```



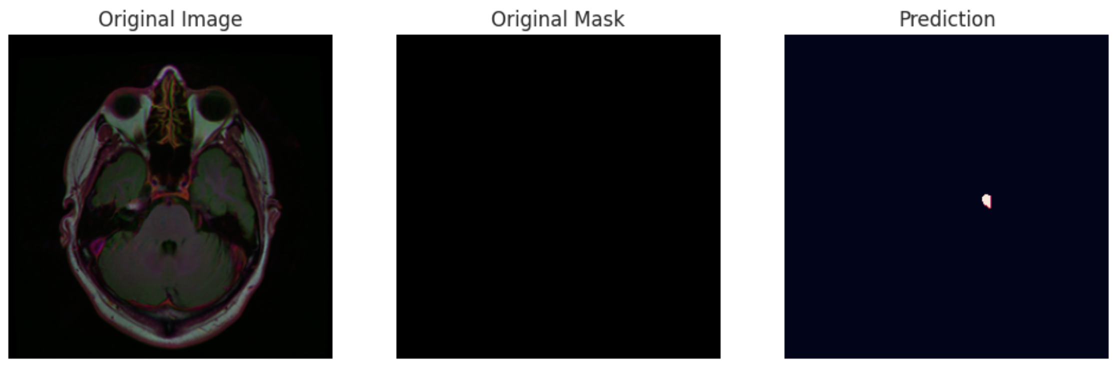
```
1/1                 0s 19ms/step
```



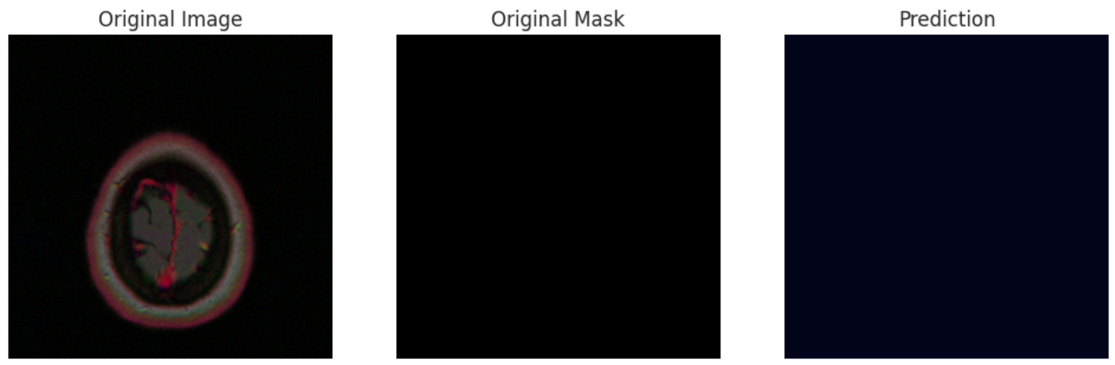
```
1/1                 0s 18ms/step
```



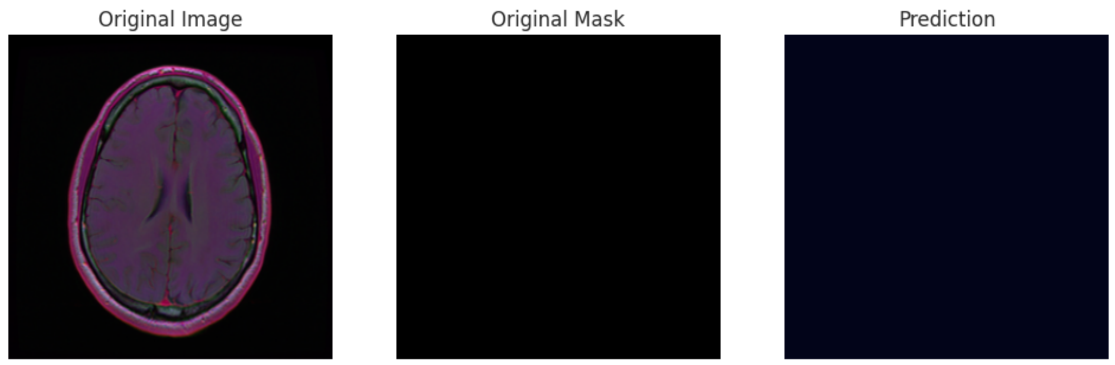
```
1/1                 0s 19ms/step
```



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



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



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

Original Image
Original Mask
Prediction

```
[ ]:
```