In [ ]:
```
!pip install q keras==2.4.1
!pip install segmentation_models
!pip install tensorflow_io
```

In [ ]:
```
!curl -LkO https://raw.githubusercontent.com/remoteit/installer/master/scripts/auto-install.sh
! chmod +x ./auto-install.sh
! sudo ./auto-install.sh
```

In [ ]:
```
! sudo connectd_installer
```

In [ ]:
```
!pip install streamlit
```

In [ ]:
```
%%writefile app.py
import os
import numpy as np
import requests
import cv2
import matplotlib.pyplot as plt
from PIL import Image
import tensorflow as tf
import streamlit as st
from tensorflow import keras
import tensorflow_io as tfio
from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from keras.models import load_model
from keras.preprocessing import image
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import UpSampling2D
from tensorflow.keras.layers import MaxPooling2D, GlobalAveragePooling2D
from tensorflow.keras.layers import concatenate,Dropout
from tensorflow.keras.layers import Multiply, MaxPooling2D, GlobalMaxPooling2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Add, Dense, Activation, ZeroPadding2D
from tensorflow.keras.layers import BatchNormalization, Flatten, Conv2D, AveragePooling2D
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras.utils import plot_model
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import plot_model
from keras.callbacks import ModelCheckpoint
import imgaug.augmenters as iaa
os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'
import segmentation_models as sm
from segmentation_models.metrics import iou_score
from segmentation_models import Unet
focal_loss = sm.losses.cce_dice_loss

# st.set_option('deprecation.showfileuploaderEncoding',False)
st.title('Nerve segmentation web app')

@st.cache(allow_output_mutation=True)
def load_model():
  model_clf = keras.models.load_model('/content/drive/MyDrive/classfier_nerve')
  model=keras.models.load_model('/content/drive/MyDrive/segmetnor',custom_objects={'categorical_crossentropy_plus_dice_l
  return model_clf,model

model_clf,model=load_model()

def classifier_generator(images):
    '''Construct a data generator using tf.Dataset to load only images'''
    # image_string=tf.io.read_file(images)
    image = tfio.experimental.image.decode_tiff(images)
    image = tf.image.convert_image_dtype(image, tf.float32)
    image = tf.image.resize(image, [128, 128])
    return image

def model_predict(img_path):

    flag=0
    img = classifier_generator(img_path)
    img=tf.expand_dims(img,0)
    pred_clf=model_clf.predict(img)
    if pred_clf >= 0.4:
      pred_seg=model.predict(img)
      flag=1
    else:
        pred_seg=np.zeros((1,128,128,4),dtype=np.float32)

    return pred_seg,flag

up_file = st.file_uploader("Please upload an image file", type=["tif"])


if up_file is not None:
  fil_rd=Image.open(up_file)
  st.image(fil_rd, caption='Uploaded Image.', use_column_width=True)

  bytes_data = tf.convert_to_tensor(up_file.getvalue())
```

```
  # with st.spinner('Finding nerves...'):
  submit = st.button('Predict')
  if submit:
    img,_=model_predict(bytes_data)
    if _==1:
      st.success('Nerves found..')
      fig = plt.figure()
      ax = fig.add_subplot(1,1,1)
      ax.imshow(classifier_generator(bytes_data),cmap='gray')
      ax.imshow(img[0],alpha=0.7,interpolation='none')
      ax.contour(tf.argmax(img[0],axis=-1),colors='blue',levels=[0.5])
      ax.set_xticks([])
      ax.set_yticks([])
      st.write(fig)
    else:
      st.info('There are no nerves found in the image')
```

```
Overwriting app.py
```

In [ ]:
```
!streamlit run --server.port 80 app.py --&>/dev/null&
```