```
!pip install -q kaggle
from google.colab import files

# Upload your kaggle.json (Kaggle API key)
files.upload()

# Set up Kaggle API credentials
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Choose files kaggle.json

- **kaggle.json**(application/json) - 72 bytes, last modified: 10/06/2025 - 100% done

```
!kaggle datasets download -d tapakah68/segmentation-full-body-mads-dataset
!unzip segmentation-full-body-mads-dataset.zip -d mads_dataset
```

```
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
inflating: mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img/
```

```
!pip uninstall -y keras
!pip uninstall -y tensorflow
!pip install tensorflow==2.18.0
!pip install keras
```

```
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from t
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /u
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (75.2
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (1.1
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow==
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (1
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2
Collecting keras>=3.5.0 (from tensorflow==2.18.0)
  Downloading keras-3.10.0-py3-none-any.whl.metadata (6.0 kB)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.18.0) (3.1
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0-
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow==2.18.0
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow==2.18
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow==2.18
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->ten
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->te
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->ric
Downloading tensorflow-2.18.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (615.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 615.4/615.4 MB 2.9 MB/s eta 0:00:00
Downloading keras-3.10.0-py3-none-any.whl (1.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 63.2 MB/s eta 0:00:00
Installing collected packages: keras, tensorflow
Successfully installed keras-3.10.0 tensorflow-2.18.0
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from keras) (2.0.2)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras) (0.1.0)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras) (3.13.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras) (0.16.0)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras) (0.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras) (24.2)
Requirement already satisfied: typing-extensions>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from optree->kera
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras) (3
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->ric
```

```
import pandas as pd
import os

# Load CSV
df = pd.read_csv('mads_dataset/df.csv')

# Define base directory path
base_path = 'mads_dataset/segmentation_full_body_mads_dataset_1192_img/segmentation_full_body_mads_dataset_1192_img'

# Correct column names
image_paths = [os.path.join(base_path, path) for path in df['images']]
mask_paths = [os.path.join(base_path, path) for path in df['masks']]

# Filter out missing or corrupted files
valid_image_paths = []
valid_mask_paths = []

from sklearn.model_selection import train_test_split

train_img, val_img, train_mask, val_mask = train_test_split(
    image_paths, mask_paths, test_size=0.1, random_state=42)

train_gen = DataGenerator(train_img, train_mask, batch_size=8)
val_gen = DataGenerator(val_img, val_mask, batch_size=8)


for img, mask in zip(df['image_path'], df['mask_path']):
    if os.path.exists(img) and os.path.exists(mask):
```

```
        valid_image_paths.append(img)
        valid_mask_paths.append(mask)
    else:
        print(f"[WARNING] Skipping corrupted or missing file: {img} or {mask}")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3804         try:
-> 3805             return self._engine.get_loc(casted_key)
   3806         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'image_path'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                  2 frames
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3810             ):
   3811                 raise InvalidIndexError(key)
-> 3812             raise KeyError(key) from err
   3813         except TypeError:
   3814             # If we have a listlike key, _check_indexing_error will raise

KeyError: 'image_path'
```

Next steps: Explain error

```
import numpy as np
import cv2
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.utils import Sequence
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split



class DataGenerator(Sequence):
    def __init__(self, image_paths, mask_paths, batch_size=8, image_size=(256, 256)):
        self.image_paths = image_paths
        self.mask_paths = mask_paths
        self.batch_size = batch_size
        self.image_size = image_size
        self.indexes = np.arange(len(self.image_paths))

    def __len__(self):
        return int(np.ceil(len(self.image_paths) / self.batch_size))

    def __getitem__(self, index):
        # Get batch indices
        idxs = self.indexes[index * self.batch_size : (index + 1) * self.batch_size]

        # Init arrays
        batch_images = []
        batch_masks = []

        for i in idxs:
            img_path = self.image_paths[i]
            mask_path = self.mask_paths[i]

            img = cv2.imread(img_path)
            mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)

            if img is None or mask is None:
                print(f"[WARNING] Skipping corrupted or missing file: {img_path} or {mask_path}")
                continue

            img = cv2.resize(img, self.image_size)
```

```
            img = cv2.resize(img, self.image_size)
            mask = cv2.resize(mask, self.image_size)

            # Normalize
            img = img / 255.0
            mask = (mask > 127).astype(np.float32)  # Binary mask

            batch_images.append(img)
            batch_masks.append(mask[..., np.newaxis])  # Add channel dimension

        return np.array(batch_images), np.array(batch_masks)



def build_unet(input_shape=(256, 256, 3)):
    inputs = layers.Input(input_shape)

    def conv_block(x, filters):
        x = layers.Conv2D(filters, 3, activation='relu', padding='same')(x)
        x = layers.Conv2D(filters, 3, activation='relu', padding='same')(x)
        return x

    def encoder_block(x, filters):
        f = conv_block(x, filters)
        p = layers.MaxPooling2D((2, 2))(f)
        return f, p

    def decoder_block(x, conv_output, filters):
        x = layers.UpSampling2D((2, 2))(x)
        x = layers.Concatenate()([x, conv_output])
        x = conv_block(x, filters)
        return x

    # Encoder
    f1, p1 = encoder_block(inputs, 64)
    f2, p2 = encoder_block(p1, 128)
    f3, p3 = encoder_block(p2, 256)
    f4, p4 = encoder_block(p3, 512)

    # Bottleneck
    bottleneck = conv_block(p4, 1024)

    # Decoder
    d1 = decoder_block(bottleneck, f4, 512)
    d2 = decoder_block(d1, f3, 256)
    d3 = decoder_block(d2, f2, 128)
    d4 = decoder_block(d3, f1, 64)

    outputs = layers.Conv2D(1, (1, 1), activation='sigmoid')(d4)

    model = models.Model(inputs, outputs)
    return model



# Split data
train_img, val_img, train_mask, val_mask = train_test_split(
    image_paths, mask_paths, test_size=0.1, random_state=42)

train_gen = DataGenerator(train_img, train_mask, batch_size=8)
val_gen = DataGenerator(val_img, val_mask, batch_size=8)

# Build & compile
model = build_unet()
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train
history = model.fit(train_gen, validation_data=val_gen, epochs=10)
model.save('unet_mads_tf.h5')
```

```
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `P
  self._warn_if_super_not_called()
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-3529133148> in <cell line: 0>()
     10
     11 # Train
---> 12 history = model.fit(train_gen, validation_data=val_gen, epochs=10)
     13 model.save('unet_mads_tf.h5')

 1 frames
/usr/local/lib/python3.11/dist-packages/keras/src/models/functional.py in _adjust_input_rank(self, flat_inputs)
    274                     adjusted.append(ops.expand_dims(x, axis=-1))
    275                     continue
--> 276             raise ValueError(
    277                 f"Invalid input shape for input {x}. Expected shape "
    278                 f"{ref_shape}, but input has incompatible shape {x.shape}"

ValueError: Exception encountered when calling Functional.call().

Invalid input shape for input Tensor("functional_1_1/Cast:0", shape=(None,), dtype=float32). Expected shape (None, 256,
256, 3), but input has incompatible shape (None,)

Arguments received by Functional.call():
  • inputs=tf.Tensor(shape=(None,), dtype=float64)
  • training=True
  • mask=None
  • kwargs=<class 'inspect._empty'>
```

Next steps: Explain error

```
def predict_and_show(model, image_path):
    img = cv2.imread(image_path)
    img_resized = cv2.resize(img, (256, 256)) / 255.0
    pred_mask = model.predict(np.expand_dims(img_resized, axis=0))[0]

    pred_mask = (pred_mask > 0.5).astype(np.uint8) * 255
    pred_mask = pred_mask.squeeze()

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Original')

    plt.subplot(1, 2, 2)
    plt.imshow(pred_mask, cmap='gray')
    plt.title('Predicted Mask')

    plt.show()

# Example:
predict_and_show(model, val_img[0])
```