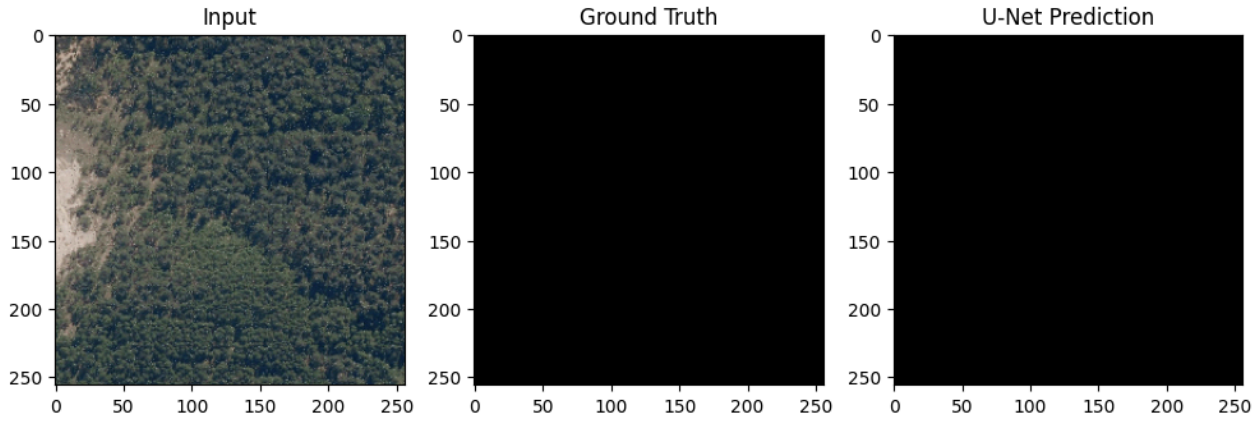
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import tensorflow as tf # Import tensorflow for tf.round and tf.reduce_sum

!pip install roboflow
from roboflow import Roboflow

print("Libraries successfully loaded!")
# ============================
# 2. DOWNLOAD ROBOFLOW DATASET
# ============================

rf = Roboflow(api_key="RhibbnwcsNrBZkRzGdQ3")
project = rf.workspace("blue-halo").project("vegetation-segmentation")
version = project.version(4)
dataset = version.download("yolov11")

root_path = dataset.location

# --- Removed problematic manual unzip block ---

root_path
# ============================
# 3. IMAGE NORMALIZATION + SAVE CLEAN DATASET
# ============================

norm_img_dir = "/content/processed_images/"
norm_mask_dir = "/content/processed_masks/"

os.makedirs(norm_img_dir, exist_ok=True)
os.makedirs(norm_mask_dir, exist_ok=True)

# --- Start of modified path detection logic ---
train_images_path = os.path.join(root_path, 'train/images')
train_labels_path = os.path.join(root_path, 'train/labels')

if os.path.exists(train_images_path) and os.path.exists(train_labels_path):
    img_src = train_images_path
    mask_src = train_labels_path
    print(f"Using dataset structure: {root_path}/train/images and {root_path}/train/labels")
else:
    # Try assuming images and labels are directly under root_path
    img_src = os.path.join(root_path, 'images')
    mask_src = os.path.join(root_path, 'labels')
    print(f"Using dataset structure: {root_path}/images and {root_path}/labels")
    print("If this is still incorrect, please manually inspect the dataset structure using `!ls -F {root_path}`")
# --- End of modified path detection logic ---

IMAGE_SIZE = 256

def yolo_mask_from_txt(txt_path, img_shape):
    """Convert YOLO polygon or bbox labels → binary mask."""
    mask = np.zeros(img_shape[:2], dtype=np.uint8)

    with open(txt_path, 'r') as f:
        lines = f.readlines()

    h, w = img_shape[:2]

    for line in lines:
        items = line.strip().split()
        cls = int(items[0])
        coords = list(map(float, items[1:]))

        if len(coords) == 4:
            # YOLO bounding box → mask
            cx, cy, bw, bh = coords
            x1 = int((cx - bw/2) * w)
            y1 = int((cy - bh/2) * h)
            x2 = int((cx + bw/2) * w)
            y2 = int((cy + bh/2) * h)
            mask[y1:y2, x1:x2] = 255
        else:
            # Optional: polygons (if dataset has them)
            pass
```

```
            pass

    return mask

# --- Added check for img_src existence before os.listdir ---
if not os.path.exists(img_src):
    print(f"Error: Image source directory '{img_src}' does not exist.")
    print("This indicates that the Roboflow dataset was not properly downloaded or extracted for the 'yolov11' format.")
    print("Please verify your dataset on Roboflow, or try downloading in a different format.")
    all_imgs = [] # Set to empty list to prevent FileNotFoundError
else:
    all_imgs = sorted(os.listdir(img_src))
# --- End of added check ---
saved = 0

for file in all_imgs:
    if not file.endswith(".jpg") and not file.endswith(".png"):
        continue

    img_path = os.path.join(img_src, file)
    label_path = os.path.join(mask_src, file.replace(".jpg", ".txt").replace(".png", ".txt"))

    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    if not os.path.exists(label_path):
        continue

    mask = yolo_mask_from_txt(label_path, img.shape)

    img_r = cv2.resize(img, (IMAGE_SIZE, IMAGE_SIZE))
    mask_r = cv2.resize(mask, (IMAGE_SIZE, IMAGE_SIZE))

    img_norm = img_r / 255.0
    mask_norm = (mask_r > 128).astype(np.float32)

    np.save(os.path.join(norm_img_dir, file.replace(".jpg", ".npy").replace(".png", ".npy")), img_norm)
    np.save(os.path.join(norm_mask_dir, file.replace(".jpg", ".npy").replace(".png", ".npy")), mask_norm)

    saved += 1

print(f"Normalization completed! {saved} samples stored.")
# ======================================================
# ✨ 4. LOAD NORMALIZED DATA
# ======================================================

X, Y = [], []

for f in os.listdir(norm_img_dir):
    if f.endswith(".npy"):
        X.append(np.load(os.path.join(norm_img_dir, f)))
        Y.append(np.load(os.path.join(norm_mask_dir, f)))

X = np.array(X)
Y = np.expand_dims(np.array(Y), axis=-1)

print("Images:", X.shape)
print("Masks :", Y.shape)

# ======================================================
# ✨ 5. TRAIN / TEST SPLIT
# ======================================================

X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.2, random_state=10
)
X_train.shape, X_test.shape
# ================================
# ✨ 6. BUILD A SIMPLE U-NET ARCHITECTURE
# ================================

def conv_block(inputs, filters):
    x = Conv2D(filters, 3, padding="same", activation="relu")(inputs)
    x = Conv2D(filters, 3, padding="same", activation="relu")(x)
    return x

def unet_model(input_shape=(256,256,3)):
    inp = Input(input_shape)

    # Encoder
    c1 = conv_block(inp, 32)
    p1 = MaxPooling2D()(c1)
```

```
    c2 = conv_block(p1, 64)
    p2 = MaxPooling2D()(c2)

    c3 = conv_block(p2, 128)
    p3 = MaxPooling2D()(c3)

    c4 = conv_block(p3, 256)
    p4 = MaxPooling2D()(c4)

    # Bottleneck
    bn = conv_block(p4, 512)

    # Decoder
    u1 = UpSampling2D()(bn)
    u1 = Concatenate()([u1, c4])
    c5 = conv_block(u1, 256)

    u2 = UpSampling2D()(c5)
    u2 = Concatenate()([u2, c3])
    c6 = conv_block(u2, 128)

    u3 = UpSampling2D()(c6)
    u3 = Concatenate()([u3, c2])
    c7 = conv_block(u3, 64)

    u4 = UpSampling2D()(c7)
    u4 = Concatenate()([u4, c1])
    c8 = conv_block(u4, 32)

    out = Conv2D(1, 1, activation="sigmoid")(c8)

    model = Model(inp, out)
    return model

model = unet_model()
# ================================
# ✨ 7. EVALUATION METRICS
# ================================

def iou_score(y_true, y_pred):
    y_pred = tf.round(y_pred)
    intersection = tf.reduce_sum(y_true * y_pred)
    union = tf.reduce_sum(y_true) + tf.reduce_sum(y_pred) - intersection
    return intersection / (union + 1e-7)

def dice_score(y_true, y_pred):
    y_pred = tf.round(y_pred)
    inter = tf.reduce_sum(y_true * y_pred)
    return (2 * inter) / (tf.reduce_sum(y_true) + tf.reduce_sum(y_pred) + 1e-7)


# ================================
# ✨ 8. COMPILE + TRAIN MODEL
# ================================

model.compile(
    optimizer=Adam(1e-4),
    loss="binary_crossentropy",
    metrics=[iou_score, dice_score]
)

history = model.fit(
    X_train, Y_train,
    validation_split=0.1,
    epochs=15,
    batch_size=8
)
# ✨ 9. TEST MODEL PREDICTIONS
# ================================================
#=====================

idx = np.random.randint(0, len(X_test))
sample = X_test[idx]
pred = model.predict(sample[np.newaxis,...])[0]

plt.figure(figsize=(12,4))
plt.subplot(1,3,1); plt.title("Input"); plt.imshow(sample)
plt.subplot(1,3,2); plt.title("Ground Truth"); plt.imshow(Y_test[idx].squeeze(), cmap="gray")
plt.subplot(1,3,3); plt.title("U-Net Prediction"); plt.imshow(pred.squeeze(), cmap="gray")
plt.show()
```

```
Requirement already satisfied: roboflow in /usr/local/lib/python3.12/dist-packages (1.2.11)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from roboflow) (2025.11.12)
Requirement already satisfied: idna==3.7 in /usr/local/lib/python3.12/dist-packages (from roboflow) (3.7)
Requirement already satisfied: cycler in /usr/local/lib/python3.12/dist-packages (from roboflow) (0.12.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from roboflow) (1.4.9)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from roboflow) (3.10.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from roboflow) (2.0.2)
Requirement already satisfied: opencv-python-headless==4.10.0.84 in /usr/local/lib/python3.12/dist-packages (from roboflow)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.12/dist-packages (from roboflow) (11.3.0)
Requirement already satisfied: pi-heif<2 in /usr/local/lib/python3.12/dist-packages (from roboflow) (1.1.1)
Requirement already satisfied: pillow-avif-plugin<2 in /usr/local/lib/python3.12/dist-packages (from roboflow) (1.5.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.12/dist-packages (from roboflow) (2.9.0.post0)
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.12/dist-packages (from roboflow) (1.2.1)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from roboflow) (2.32.4)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from roboflow) (1.17.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.12/dist-packages (from roboflow) (2.5.0)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.12/dist-packages (from roboflow) (4.67.1)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.12/dist-packages (from roboflow) (6.0.3)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.12/dist-packages (from roboflow) (1.0.0)
Requirement already satisfied: filetype in /usr/local/lib/python3.12/dist-packages (from roboflow) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->roboflow) (1.3.
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->roboflow) (4.6
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->roboflow) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->roboflow) (3.2.
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->roboflow)
Libraries successfully loaded!
loading Roboflow workspace...
loading Roboflow project...
Using dataset structure: /content/vegetation-segmentation-4/train/images and /content/vegetation-segmentation-4/train/labels
Normalization completed! 945 samples stored.
Images: (945, 256, 256, 3)
Masks : (945, 256, 256, 1)
Epoch 1/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1689s 20s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 0.3320 - val_dice_score:
Epoch 2/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1658s 20s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 1.4200e-09 - val_dice_scc
Epoch 3/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1641s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 3.8119e-09 - val_dice_scc
Epoch 4/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1599s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 7.3181e-09 - val_dice_scc
Epoch 5/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1632s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 1.2019e-08 - val_dice_scc
Epoch 6/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1642s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 5.6263e-09 - val_dice_scc
Epoch 7/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1600s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 2.1783e-09 - val_dice_scc
Epoch 8/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1640s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 2.4790e-09 - val_dice_scc
Epoch 9/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1648s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 3.0445e-09 - val_dice_scc
Epoch 10/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1638s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 3.5535e-09 - val_dice_scc
Epoch 11/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1631s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 2.6174e-09 - val_dice_scc
Epoch 12/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1642s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 7.3237e-09 - val_dice_scc
Epoch 13/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1641s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 5.3047e-09 - val_dice_scc
Epoch 14/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1626s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 1.5140e-09 - val_dice_scc
Epoch 15/15
85/85 ━━━━━━━━━━━━━━━━━━━━ 1627s 19s/step - dice_score: 0.0000e+00 - iou_score: 0.0000e+00 - loss: 2.4197e-09 - val_dice_scc
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
```




```
!ls -F /content/vegetation-segmentation-4
```

```
roboflow.zip
```