Infosys Vegetation Segmentation Analysis

Pretrained Model Training:

Test 1 :

```
# Load segmentation model (nano or medium)
model = YOLO('yolov8n-seg.pt')

model.train(
    data="/content/vegetation/data.yaml",
    epochs=50,          # Max epochs
    imgsz=640,

    # Early Stopping
    patience=5,         # Stop if no improvement for 5 epochs

    # Augmentations
    hsv_h=0.03,
    hsv_s=0.6,
    hsv_v=0.5,
    degrees=10,
    translate=0.1,
    scale=0.5,
    shear=2,
    flipud=0.0,
    fliplr=0.5,
    mosaic=1.0,
    mixup=0.2,
    copy_paste=0.2
)
```

✅ **Results:**
mAP50: 0.5734
mAP50-95: 0.3126
Precision: 0.5949
Recall: 0.5000

Test 2:

```
from ultralytics import YOLO

# Load segmentation model (nano or medium)
model = YOLO("yolov8m-seg.pt")
model.train(
    data="/content/vegetation/data.yaml",
    epochs=50,              # max 50 epochs
    imgsz=1024,             # higher resolution for finer segmentation
    batch=8,                # adjust based on GPU memory

    # Early Stopping
    patience=10,            # stop if no improvement in 10 epochs

    # Learning Rate Scheduler (Cosine decay)
    lr0=0.01,               # initial learning rate
    lrf=0.001,              # final learning rate (cosine decay)
    cos_lr=True,            # enable cosine learning rate

    # Optimized Augmentations
    hsv_h=0.02,
    hsv_s=0.5,
    hsv_v=0.4,
    degrees=10,
    translate=0.05,
    scale=0.3,
    shear=2,
    flipud=0.0,
    fliplr=0.5,
    mosaic=0.5,
    mixup=0.1,
    copy_paste=0.0
)
```

```
✅ Results:
mAP50:      0.6083
mAP50-95:   0.3943
Precision: 0.6404
Recall:     0.5802
```

Test 3:

```
# 1. Load Pretrained Model (Large for best accuracy)
model = YOLO("yolov8l-seg.pt")

# 2. Train Model with Optimized Hyperparameters
model.train(
    data="/content/vegetation/data.yaml",  # dataset config
    epochs=50,                # max epochs (early stopping may stop
earlier)
    imgsz=1024,               # high resolution for fine segmentation
    batch=4,                  # smaller batch for Colab GPU
    patience=15,              # stop if no improvement after 15 epochs

    # ⚡ Learning Rate & Optimizer
    lr0=0.003,                # lower initial LR for stability
    lrf=0.0005,               # smaller final LR
    cos_lr=True,              # cosine decay schedule
    optimizer="AdamW",        # more stable than SGD
    weight_decay=0.001,       # regularization (avoid overfitting)

    # 🎨 Data Augmentation (strong but not too heavy)
    hsv_h=0.015,
    hsv_s=0.7,
    hsv_v=0.4,
    degrees=15,
    translate=0.1,
    scale=0.5,
    shear=3,
    flipud=0.2,
    fliplr=0.5,
    mosaic=0.8,
    mixup=0.2,
    copy_paste=0.1
)
```

```
Results:
✅ Final Evaluation Metrics:
mAP50:     0.5923
mAP50-95:  0.3417
Precision: 0.6692
Recall:    0.5076
```

Model from scratch Training:

```
def build_segmentation_model(img_size=128):
    inputs = layers.Input((img_size, img_size, 3))

    # Encoder
    c1 = layers.Conv2D(32, (3,3), activation="relu",
padding="same")(inputs)
    p1 = layers.MaxPooling2D((2,2))(c1)

    c2 = layers.Conv2D(64, (3,3), activation="relu",
padding="same")(p1)
    p2 = layers.MaxPooling2D((2,2))(c2)

    # Bottleneck
    b1 = layers.Conv2D(128, (3,3), activation="relu",
padding="same")(p2)

    # Decoder
    u1 = layers.UpSampling2D((2,2))(b1)
    c3 = layers.Conv2D(64, (3,3), activation="relu",
padding="same")(u1)

    u2 = layers.UpSampling2D((2,2))(c3)
    c4 = layers.Conv2D(32, (3,3), activation="relu",
padding="same")(u2)

    outputs = layers.Conv2D(1, (1,1), activation="sigmoid")(c4)

    return models.Model(inputs, outputs)

model = build_segmentation_model(IMG_SIZE)
optimizer=Adam(learning_rate=1e-3)

model.compile(optimizer=optimizer, loss="binary_crossentropy",
metrics=["accuracy"])
model.summary()
Results:
✅ Accuracy:  0.9471
✅ Precision: 0.9360
✅ Recall:    0.9540
✅ F1 Score:  0.9449
✅ IoU:       0.8955
```