

Start coding or [generate](#) with AI.

```
# Install necessary libraries
!pip install kaggle
# Set up Kaggle API credentials
import os
os.environ['KAGGLE_USERNAME'] = 'yamrajkhadka01'
os.environ['KAGGLE_KEY'] = '/content/kaggle (5).json'
# Download the dataset
!kaggle datasets download -d balraj98/deepglobe-land-cover-classification-dataset -p /content/
# Unzip the dataset
!unzip -q /content/deepglobe-land-cover-classification-dataset.zip -d /content/deepglobe_dataset/

➦ Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages (1.7.4.5)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2025.4.26)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.4.2)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from kaggle) (5.29.5)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.9.0.post0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.11/dist-packages (from kaggle) (75.2.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.4.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from kaggle) (0.5.1)
Dataset URL: https://www.kaggle.com/datasets/balraj98/deepglobe-land-cover-classification-dataset
License(s): other
Downloading deepglobe-land-cover-classification-dataset.zip to /content
 98% 2.67G/2.74G [00:24<00:01, 56.3MB/s]
100% 2.74G/2.74G [00:24<00:00, 121MB/s]
```

```
# Full Colab-ready Code for DeepGlobe Segmentation using U-Net with Improvements
```

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Dropout, Conv2DTranspose, concatenate, BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping, Callback
import albumentations as A

# 1. Load Class Mapping
CLASS_DICT_CSV = '/content/deepglobe_dataset/class_dict.csv'
assert os.path.exists(CLASS_DICT_CSV), "class_dict.csv not found!"

class_dict = pd.read_csv(CLASS_DICT_CSV)
color2class = {(row['r'], row['g'], row['b']): idx for idx, row in class_dict.iterrows()}
NUM_CLASSES = len(class_dict)
```

```

def rgb_to_class(mask_rgb):
    h, w, _ = mask_rgb.shape
    mask_class = np.zeros((h, w), dtype=np.uint8)
    for color, class_idx in color2class.items():
        matches = np.all(mask_rgb == color, axis=-1)
        mask_class[matches] = class_idx
    return mask_class

def class_to_rgb(mask_class):
    h, w = mask_class.shape
    mask_rgb = np.zeros((h, w, 3), dtype=np.uint8)
    for color, class_idx in color2class.items():
        mask_rgb[mask_class == class_idx] = color
    return mask_rgb

# 2. Load Dataset and Apply Augmentations
TRAIN_IMG_DIR = '/content/deepglobe_dataset/train'
assert os.path.exists(TRAIN_IMG_DIR), "Train directory not found!"

IMG_SIZE = (256, 256)
img_ids = [f[:-8] for f in os.listdir(TRAIN_IMG_DIR) if f.endswith('_sat.jpg')]
N = min(1000, len(img_ids))

transform = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.3),
    A.RandomBrightnessContrast(p=0.3),
    A.Affine(rotate=(-15, 15), scale=(0.9, 1.1), translate_percent=0.1, p=0.5),
])

X, Y = [], []
for img_id in img_ids[:N]:
    img_path = os.path.join(TRAIN_IMG_DIR, f"{img_id}_sat.jpg")
    mask_path = os.path.join(TRAIN_IMG_DIR, f"{img_id}_mask.png")

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

    augmented = transform(image=img, mask=mask)
    img = cv2.resize(augmented['image'], IMG_SIZE)
    mask = cv2.resize(augmented['mask'], IMG_SIZE, interpolation=cv2.INTER_NEAREST)

    img = img / 255.0
    mask_class = rgb_to_class(mask)
    mask_onehot = to_categorical(mask_class, num_classes=NUM_CLASSES)

    X.append(img)
    Y.append(mask_onehot)

X = np.array(X, dtype=np.float32)
Y = np.array(Y, dtype=np.float32)
print(f"X: {X.shape} Y: {Y.shape}")

```

```

# 3. Compute Class Weights
flat_mask = np.array(Y, dtype=np.float32)

```

```

flat_mask = np.argmax(r, axis=-1).reshape(-1)
class_counts = np.bincount(flat_mask, minlength=NUM_CLASSES)
class_weights = class_counts.max() / (class_counts + 1e-6)
class_weights = class_weights / class_weights.sum() * NUM_CLASSES
print("Class weights:", class_weights)

```

#### # 4. Custom Focal Loss

```

from tensorflow.keras import backend as K
def categorical_focal_loss(alpha, gamma=2.0):
    alpha = tf.constant(alpha, dtype=tf.float32)
    def loss_fn(y_true, y_pred):
        y_true = tf.convert_to_tensor(y_true, tf.float32)
        y_pred = tf.convert_to_tensor(y_pred, tf.float32)
        y_pred = tf.clip_by_value(y_pred, K.epsilon(), 1. - K.epsilon())
        cross_entropy = -y_true * tf.math.log(y_pred)
        weight = alpha * tf.pow(1 - y_pred, gamma)
        focal_loss = weight * cross_entropy
        return tf.reduce_mean(tf.reduce_sum(focal_loss, axis=-1))
    return loss_fn

```

```
loss_fn = categorical_focal_loss(class_weights)
```

#### # 5. Mean IoU Metric

```

class MeanIoU(tf.keras.metrics.MeanIoU):
    def __init__(self, num_classes, name='mean_iou', **kwargs):
        super().__init__(num_classes=num_classes, name=name, **kwargs)
    def update_state(self, y_true, y_pred, sample_weight=None):
        y_true = tf.argmax(y_true, axis=-1)
        y_pred = tf.argmax(y_pred, axis=-1)
        return super().update_state(y_true, y_pred, sample_weight)

```

#### # 6. Build U-Net Model

```

def build_unet(input_shape=(256, 256, 3), num_classes=NUM_CLASSES):
    def conv_block(x, filters):
        x = Conv2D(filters, 3, padding='same', activation='relu')(x)
        x = BatchNormalization()(x)
        x = Conv2D(filters, 3, padding='same', activation='relu')(x)
        x = BatchNormalization()(x)
        return x

    inputs = Input(input_shape)
    c1 = conv_block(inputs, 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)
    b = conv_block(p4, 512); b = Dropout(0.4)(b)
    u1 = Conv2DTranspose(256, 2, strides=2, padding='same')(b); u1 = concatenate([u1, c4]); c5 = conv_block(u1, 256)
    u2 = Conv2DTranspose(128, 2, strides=2, padding='same')(c5); u2 = concatenate([u2, c3]); c6 = conv_block(u2, 128)
    u3 = Conv2DTranspose(64, 2, strides=2, padding='same')(c6); u3 = concatenate([u3, c2]); c7 = conv_block(u3, 64)
    u4 = Conv2DTranspose(32, 2, strides=2, padding='same')(c7); u4 = concatenate([u4, c1]); c8 = conv_block(u4, 32)
    outputs = Conv2D(num_classes, 1, activation='softmax')(c8)
    return Model(inputs, outputs)

model = build_unet()
model.compile(optimizer='adam', loss=loss_fn, metrics=[MeanIoU(NUM_CLASSES)])
model.summary()

```


```
# 7. Training with Callbacks
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=8, min_lr=1e-5, verbose=1)
early_stop = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True, verbose=1)

history = model.fit(
    X, Y,
    validation_split=0.2,
    epochs=200,
    batch_size=8,
    shuffle=True,
    callbacks=[reduce_lr, early_stop],
    verbose=1
)

# 8. Plot Per-Class IoU
preds = model.predict(X)
preds_class = np.argmax(preds, axis=-1)
true_class = np.argmax(Y, axis=-1)

ious = []
for i in range(NUM_CLASSES):
    intersection = np.logical_and(preds_class == i, true_class == i).sum()
    union = np.logical_or(preds_class == i, true_class == i).sum()
    iou = intersection / (union + 1e-6)
    ious.append(iou)
    print(f"Class {i} IoU: {iou:.4f}")

print("Mean IoU:", np.mean(ious))
```

 X: (803, 256, 256, 3) Y: (803, 256, 256, 7)  
 Class weights: [0.76662306 0.14388482 0.97833991 0.75141309 2.50358822 0.99213277  
 0.86401812]  
 Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 256, 256, 3)	0	-
conv2d (Conv2D)	(None, 256, 256, 32)	896	input_layer[0][0]
batch_normalization (BatchNormalizatio...	(None, 256, 256, 32)	128	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 256, 256, 32)	9,248	batch_normalizat...
batch_normalizatio... (BatchNormalizatio...	(None, 256, 256, 32)	128	conv2d_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0	batch_normalizat...
conv2d_2 (Conv2D)	(None, 128, 128, 64)	18,496	max_pooling2d[0]...
batch_normalizatio... (BatchNormalizatio...	(None, 128, 128, 64)	256	conv2d_2[0][0]
conv2d_3 (Conv2D)	(None, 128, 128, 64)	36,928	batch_normalizat...
batch_normalizatio... (BatchNormalizatio...	(None, 128, 128, 64)	256	conv2d_3[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0	batch_normalizat...
conv2d_4 (Conv2D)	(None, 64, 64, 128)	73,856	max_pooling2d_1[...
batch_normalizatio... (BatchNormalizatio...	(None, 64, 64, 128)	512	conv2d_4[0][0]
conv2d_5 (Conv2D)	(None, 64, 64, 128)	147,584	batch_normalizat...
batch_normalizatio... (BatchNormalizatio...	(None, 64, 64, 128)	512	conv2d_5[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 128)	0	batch_normalizat...
conv2d_6 (Conv2D)	(None, 32, 32, 256)	295,168	max_pooling2d_2[...
batch_normalizatio... (BatchNormalizatio...	(None, 32, 32, 256)	1,024	conv2d_6[0][0]
conv2d_7 (Conv2D)	(None, 32, 32, 256)	590,080	batch_normalizat...

batch_normalizatio... (BatchNormalizatio...)	(None, 32, 32, 256)	1,024	conv2d_7[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 256)	0	batch_normalizat...
conv2d_8 (Conv2D)	(None, 16, 16, 512)	1,180,160	max_pooling2d_3[...]
batch_normalizatio... (BatchNormalizatio...)	(None, 16, 16, 512)	2,048	conv2d_8[0][0]
conv2d_9 (Conv2D)	(None, 16, 16, 512)	2,359,808	batch_normalizat...
batch_normalizatio... (BatchNormalizatio...)	(None, 16, 16, 512)	2,048	conv2d_9[0][0]
dropout (Dropout)	(None, 16, 16, 512)	0	batch_normalizat...
conv2d_transpose (Conv2DTranspose)	(None, 32, 32, 256)	524,544	dropout[0][0]
concatenate (Concatenate)	(None, 32, 32, 512)	0	conv2d_transpose... batch_normalizat...
conv2d_10 (Conv2D)	(None, 32, 32, 256)	1,179,904	concatenate[0][0]
batch_normalizatio... (BatchNormalizatio...)	(None, 32, 32, 256)	1,024	conv2d_10[0][0]
conv2d_11 (Conv2D)	(None, 32, 32, 256)	590,080	batch_normalizat...
batch_normalizatio... (BatchNormalizatio...)	(None, 32, 32, 256)	1,024	conv2d_11[0][0]
conv2d_transpose_1 (Conv2DTranspose)	(None, 64, 64, 128)	131,200	batch_normalizat...
concatenate_1 (Concatenate)	(None, 64, 64, 256)	0	conv2d_transpose... batch_normalizat...
conv2d_12 (Conv2D)	(None, 64, 64, 128)	295,040	concatenate_1[0]...
batch_normalizatio... (BatchNormalizatio...)	(None, 64, 64, 128)	512	conv2d_12[0][0]
conv2d_13 (Conv2D)	(None, 64, 64, 128)	147,584	batch_normalizat...
batch_normalizatio... (BatchNormalizatio...)	(None, 64, 64, 128)	512	conv2d_13[0][0]
conv2d_transpose_2 (Conv2DTranspose)	(None, 128, 128, 64)	32,832	batch_normalizat...
concatenate_2 (Concatenate)	(None, 128, 128, 128)	0	conv2d_transpose... batch_normalizat...

conv2d_14 (Conv2D)	(None, 128, 128, 64)	73,792	concatenate_2[0]...
batch_normalizatio... (BatchNormalizatio...	(None, 128, 128, 64)	256	conv2d_14[0][0]
conv2d_15 (Conv2D)	(None, 128, 128, 64)	36,928	batch_normalizat...
batch_normalizatio... (BatchNormalizatio...	(None, 128, 128, 64)	256	conv2d_15[0][0]
conv2d_transpose_3 (Conv2DTranspose)	(None, 256, 256, 32)	8,224	batch_normalizat...
concatenate_3 (Concatenate)	(None, 256, 256, 64)	0	conv2d_transpose... batch_normalizat...
conv2d_16 (Conv2D)	(None, 256, 256, 32)	18,464	concatenate_3[0]...
batch_normalizatio... (BatchNormalizatio...	(None, 256, 256, 32)	128	conv2d_16[0][0]
conv2d_17 (Conv2D)	(None, 256, 256, 32)	9,248	batch_normalizat...
batch_normalizatio... (BatchNormalizatio...	(None, 256, 256, 32)	128	conv2d_17[0][0]
conv2d_18 (Conv2D)	(None, 256, 256, 7)	231	batch_normalizat...

Total params: 7,772,071 (29.65 MB)

Trainable params: 7,766,183 (29.63 MB)

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

Epoch 1/200

81/81 ————— 85s 546ms/step - loss: 0.6633 - mean\_iou: 0.2250 - val\_loss: 1.1755 - val\_mean\_iou: 0.0250 - learning\_rate: 0.0010

Epoch 2/200

81/81 ————— 24s 180ms/step - loss: 0.4328 - mean\_iou: 0.3253 - val\_loss: 1.0129 - val\_mean\_iou: 0.0378 - learning\_rate: 0.0010

Epoch 3/200

81/81 ————— 20s 177ms/step - loss: 0.3956 - mean\_iou: 0.3428 - val\_loss: 1.2142 - val\_mean\_iou: 0.0109 - learning\_rate: 0.0010

Epoch 4/200

81/81 ————— 14s 178ms/step - loss: 0.3970 - mean\_iou: 0.3400 - val\_loss: 0.8209 - val\_mean\_iou: 0.0322 - learning\_rate: 0.0010

Epoch 5/200

81/81 ————— 15s 179ms/step - loss: 0.3431 - mean\_iou: 0.3704 - val\_loss: 1.0878 - val\_mean\_iou: 0.0420 - learning\_rate: 0.0010

Epoch 6/200

81/81 ————— 15s 179ms/step - loss: 0.3628 - mean\_iou: 0.3530 - val\_loss: 0.8466 - val\_mean\_iou: 0.0988 - learning\_rate: 0.0010

Epoch 7/200

81/81 ————— 21s 184ms/step - loss: 0.3541 - mean\_iou: 0.3555 - val\_loss: 0.6919 - val\_mean\_iou: 0.2226 - learning\_rate: 0.0010

Epoch 8/200

81/81 ————— 20s 183ms/step - loss: 0.3529 - mean\_iou: 0.3598 - val\_loss: 0.3126 - val\_mean\_iou: 0.3632 - learning\_rate: 0.0010

Epoch 9/200

81/81 ————— 20s 180ms/step - loss: 0.3613 - mean\_iou: 0.3724 - val\_loss: 0.4084 - val\_mean\_iou: 0.3115 - learning\_rate: 0.0010

Epoch 10/200

81/81 ————— 21s 183ms/step - loss: 0.3314 - mean\_iou: 0.3751 - val\_loss: 0.3149 - val\_mean\_iou: 0.3598 - learning\_rate: 0.0010

Epoch 11/200

81/81 ————— 20s 184ms/step - loss: 0.3110 - mean\_iou: 0.3874 - val\_loss: 0.5363 - val\_mean\_iou: 0.2834 - learning\_rate: 0.0010

Epoch 12/200

81/81 ————— 20s 183ms/step - loss: 0.3237 - mean\_iou: 0.3996 - val\_loss: 0.6791 - val\_mean\_iou: 0.1458 - learning\_rate: 0.0010

Epoch 13/200

81/81 ————— 15s 182ms/step - loss: 0.3320 - mean\_iou: 0.3812 - val\_loss: 0.3442 - val\_mean\_iou: 0.3629 - learning\_rate: 0.0010