test.py

```python
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"]="2"

import numpy as np
import cv2
import pandas as pd
from glob import glob
from tqdm import tqdm
import tensorflow as tf
# from tensorflow.kera.utils import customObjectScope
from sklearn.metrics import f1_score, jaccard_score, recall_score
from sklearn.model_selection import train_test_split
# from metrics import dice_loss, dice_coef
from train import load_dataset
from unet import build_unet

"""Global parameters"""
H=256
W=256

"""Creating a directory"""
def create_dir(path):
    if not os.path.exists(path):
        os.makedirs(path)

def save_results(image, mask, Y_pred, save_image_path):
   mask = np.expand_dims(mask, axis=-1)
   mask = np.concatenate([mask, mask, mask], axis=-1)

   Y_pred = np.expand_dims(mask, axis=-1)
   Y_pred = np.concatenate([mask, mask, mask], axis=-1)
   Y_pred = Y_pred * 255

   line = np.ones((H, 10, 3)) * 255


   cat_images = np.concatenate([image, line, mask, line, Y_pred], axis=1)
   cv2.imwrite(save_image_path, cat_images)


   print(image.shape, mask.shape, y_pred.shape)

if __name__ == " __main__ ":
   np.random.seed(42)
   tf.random.set_seed(42)

# """directory for storing files"""
create_dir("result")

# """load the model"""
with customObjectScope({"dice_coef": dice_coef,"dice_loss":dice_loss}):
   model= tf.keras.model(os.path.join("files" , "model.h5"))

```

```
54
55  """dataset"""
56  dataset_path = "C:\pythonu\data"
57  (train_x, train_y),(valid_x, valid_y), (test_x, test_y) = load_dataset(dataset_path)
58
59  #  """Prediction and Evalution"""
60  SCORE = []
61  for x, y in tqdm(zip(test_x, test_y), total=len(test_y)):
62      """extracting the name"""
63      name = x.split("/") [-1]
64
65
66      """Reading the image"""
67      image = cv2.imread (x, cv2.IMREAD_COLOR) ## [H, w, 3]
68      image = cv2.resize(image, (W, H))        ## [H, w, 3]
69      x = image/255.0                          ## [H, w, 3]
70      x = np. expand_dims(x,axis=0)            ## [1, H, w, 3]
71
72
73      # """ Reading the mask """
74      mask = cv2.imread (y, cv2. IMREAD_GRAYSCALE)
75      mask = cv2.resize(mask, (W, H))
76      # """ Prediction"""
77      Y_pred = model.predict (x, verbose=0) [0]
78      Y_pred = np.squeeze(Y_pred, axis=-1)
79      Y_pred = y_pred >= 0.5
80      Y_pred = y_pred.astype(np. int32)
81
82      """ Saving the prediction """
83      save_image_path = os.path.join("results", name)
84      save_results(image, mask, Y_pred, save_image_path)
85
86      """ Flatten the array """
87      mask = mask/255.0
88  mask = (mask > 0.5).astype(np.int32).flatten()
89  Y_pred = Y_pred.flatten()
90
91  """ calculate the metrics values """
92  f1_value = f1_score(mask, Y_pred, labels=[0, 1], average="binary")
93  jac_value = jaccard_score(mask, Y_pred, labels=[0, 1], average="binary")
94  recall_value = recall_score(mask, Y_pred, labels=[0, 1], average="binary")
95  precision_value = precision_score(mask, Y_pred, labels=[0, 1], average="binary")
96  SCORE.append([name, f1_value, jac_value, recall_value, precision_value])
97
98  """ Metrics values """
99  score =[s[1:]for s in SCORE]
100 score = np.mean(score, axis=0)
101 print(f"F1: {score[0]:0.5f}")
102 print(f"Jaccard: {score[1]:0.5f}")
103 print(f"Recall: {score[2]:0.5f}")
104 print(f"Precision: {score[3]:0.5f}")
105
106 df = pd.DataFrame(SCORE, columns=["Image", "F1", "Jaccard", "Recall", "precision"])
107 df.to_csv("files/score.csv")
```