Code Description

Contents

Compare.py

Import Libraries

```
import numpy as np
```

وارد نمودن کتابخانه نامپی: NumPy یک کتابخانه برای زبان برنامه نویسی پایتون (Python) است. با استفاده از این کتابخانه امکان استفاده از آرایه‌ها و ماتریس‌های بزرگ چند بعدی فراهم می‌شود. هم‌چنین می‌توان از تابع‌های ریاضیاتی سطح بالا بر روی این آرایه‌ها استفاده کرد.

```
import matplotlib.pyplot as plt
```

وارد نمودن کتابخانه مت پلات: مت پلات لیب (Matplotlib) یک کتابخانه پایتون برای بصری کردن داده‌ها است. ما می‌توانیم با استفاده از آن برای نشان دادن بینش حاصل از تجزیه و تحلیل داده‌ها، انواع نمودارهای علمی و آماری یک‌بعدی مانند: نمودار خطی، دوبعدی مانند: میله‌ای، پراکندگی، هیستوگرام و سه‌بعدی مانند: کانتور پلات را ایجاد کنیم.

```
from scipy import ndimage as ndi
```

وارد نمودن کتابخانه سای پای – تصویر چند بعدی: پکیج SciPy یک پکیج علمی و اوپن سورس مبتنی بر زبان پایتون است و برای انجام محاسبات علمی و مهندسی مورد استفاده قرار می‌گیرد. کتابخانه‌ی SciPy بر مبنای کتابخانه‌ی NumPy است و امکان کار با آرایه‌های n بُعدی را فراهم می‌کند. این کتابخانه برای کار با آرایه‌های Numpy ایجاد شده است و بسیاری از عملیات محاسباتی و بهینه‌سازی را به طور کارا ممکن می‌کند. تشخیص لبه‌ی تصاویر تکنیکی در پردازش تصویر است که برای یافتن مرز و محدوده‌ی اشیا (Object)موجود در تصویر به کار می‌آید. این کار از طریق شناسایی ناپیوستگی‌ها در مقدار روشنایی پیکسل صورت می‌گیرد.

```
from skimage.util import random_noise
from skimage import feature
from skimage.feature import peak_local_max
from skimage import data, img_as_float
from skimage.feature import canny
from skimage import data
from skimage import filters
```

Scikit-image یک کتابخانه متن باز پردازش تصویر برای پایتون است که بر مبنای کتابخانه محاسبات علمی NumPy و SciPy توسعه یافته است. الگوریتم‌های متنوعی از مباحث قسمت بندی تصاویر، تبدیلات هندسی، کار در فضای رنگ، آنالیز تصاویر، فیلترینگ، ریخت شناسی (Morphology)، تشخیص ویژگی‌ها و ...همراه این کتابخانه در دسترس می‌باشد.

برای نصب کتابخانه پردازش تصویر Scikit-image کافی است کد زیر را استفاده کنیم:

```
pip install scikit-image
```

```
import cv2
```

برای اینکه از پکیج opencv استفاده شود باید cv2. نوشته شود و بعد آن توابعی را که از آن کتابخانه احتیاج داریم را بنویسیم .

```
from PIL import Image
```

کتابخانه PIL که به کتابخانه تصویر پایتون معروف است، مخفف عبارت Python Imaging Library می باشد. کتابخانه PIL یکی از کتابخانه‌های پردازش تصویر با پایتون محسوب می‌شود. این کتابخانه، پشتیبانی از عملیات مرتبط با پردازش تصویر نظیر باز کردن، دستکاری و ذخیره‌سازی تصاویر در فرمت‌های مختلف را به زبان پایتون اضافه می‌کند. برای استفاده از این کتابخانه ، ابتدا باید آن را نصب کنید و برای این کار کافی است که کد زیر ار در cmd سیستم ویندوز خود وارد کنید:

```
pip install pil
```

LCP – Local Context Perception

Parallel Dilation

skimage.morphology.**dilation(***image*, *selem=None*, *out=None*, *shift_x=False*, *shift_y =False***)**

اتساع مورفولوژیکی مقیاس خاکستری تصویر را برمیگرداند. اتساع مورفولوژیکی پیکسل را در (i، j) به حداکثر نسبت به تمام پیکسل های محله با مرکز (i، j) تنظیم می کند. در حقیقت این گشاد شدن ، مناطق روشن را بزرگتر و مناطق تاریک را کوچک می کند.

```
image = morphology.dilation(image)
```

Selemمحله به صورت آرایه 2 بعدی 1 و 0 بیان می شود. در صورت عدم وجود ، از عنصر ساختاری متقاطع استفاده کنید (اتصال = 1).

Ndarray Out ، اختیاری

آرایه ای برای ذخیره نتیجه مورفولوژی. در صورت عدم پذیرش ، آرایه جدیدی تخصیص داده می شود.

shift_x، shift_ybool ، اختیاری

تغییر عنصر ساختار در مورد نقطه مرکزی. این فقط بر روی عناصر ساختاری غیر عادی تأثیر می گذارد (به عنوان مثال نامطلوب با اضلاع حتی شماره دار).

Image Segmentation

تقسیم بندی تصویر وظیفه برچسب گذاری پیکسل بخشهای مورد مطالعه در یک تصویر است.

Edge-based segmentation

اولین ایده استفاده از کنتراست محلی است ، یعنی استفاده از شیب ها و نه مقادیر خاکستری.

از آنجا که زمینه بسیار صاف است ، تقریباً تمام لبه ها در مرز تصویر ها یا داخل تصویر ها یافت می شوند.

```
image_max = ndi.maximum_filter(image, size=1, mode='constant')
image_lcp_net = feature.canny(image_max)
```

Local Context Embedding

تعبیه ترجمه ای از بردار با ابعاد بالا به فضای کم بعد است. در حالت ایده آل ، یک تعبیه با قرار دادن ورودی های مشابه از نظر معنایی نزدیک به هم در فضای تعبیه ، برخی از معانی اضافی معنای ورودی را می گیرد.

```
#Make Embeddings Boundaries for Segmentation Result For LCP-Net #
result_image_lcp = color.label2rgb(image_lcp_net, image)
```

U-Net

```
edges_U_Net = feature.canny(gray)
```

Display Plot

```
# display results
fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(8, 4))
```

نمایش یک پلات یک ردیفه با چهار ستون و بزرگی فیگ سایز

تصویر نرمال خاکستری و عنوان

```
ax[0].imshow(gray, cmap='gray')
ax[0].set_title('test image', fontsize=15)
```

تصویر با فیلترینگ.

```
ax[1].imshow(edges_U_Net, cmap='gray')
ax[1].set_title('truth', fontsize=15)
```

تصویر به روش یو نت

```
ax[2].imshow(edges_U_Net, cmap='gray')
ax[2].set_title(r'U-Net', fontsize=15)
```

تصویر به روش ال سی پی نت

```
ax[3].imshow(image_lcp_net, cmap='gray')
```

```
ax[3].set_title(r'LCP-Net', fontsize=15)


for a in ax:
    a.axis('off')
fig.tight_layout()
plt.show()
```

Contour.py

Contouring Boundaries

```
result_image_lcp = color.label2rgb(image_lcp_net, image)
result_image_unet = color.label2rgb(edges_U_Net, image)
```

روشی برای کانتور در نقاط همانند بر اساس رنگ بندی قرمز، سبز، آبی.

مدل کانتور روشی است که متناسب خطوط شکاف باز یا بسته در خطوط یا لبه های تصویراست. این کار با به حداقل رساندن انرژی که بخشی از آن توسط تصویر تعریف شده است ، انجام می پذیرد. در مثالها از مدل کانتور rgb(1) برای تقسیم تصویر از بقیه تصویر با قرار دادن یک منحنی بسته رنگی به لبه های بخشها و (2) برای یافتن تاریک ترین منحنی بین دو قسمت ثابت استفاده شده است.

Display results

نمایش یک پلات یک ردیفه با چهار ستون و بزرگی فیگ سایز

تصویر نرمال خاکستری و عنوان

```
fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(8, 4))
ax[0].imshow(gray, cmap='gray')
ax[0].set_title('test image', fontsize=15)
```

تصویر با فیلترینگ.

```
ax[1].imshow(edges_U_Net, cmap='gray')
ax[1].set_title('truth', fontsize=15)
```

تصویر به روش یو نت

```
ax[2].imshow(result_image_unet, cmap='gray')
ax[2].set_title(r'U-Net', fontsize=15)
```

تصویر به روش ال سی پی نت

```
ax[3].imshow(result_image_lcp, cmap='gray')

ax[3].set_title(r'LCP-Net', fontsize=15)

for a in ax:

    a.axis('off')

fig.tight_layout()

plt.show()
```

Prediction.py

```
img_path = r"C:\Users\Mahsa\Desktop\Tasks\8-#P3110-CT-
Scan_Segmentation\Code\MyCode\Dataset\train" # Enter Directory of all
images
```

وارد نمودن مسیر تصاویر دیتاست DRIVE که در فولدر train است.

```
folder = img_path

images = [os.path.join(root, filename(

          for root, dirs, files in os.walk(folder)

          for filename in files

          if filename.lower().endswith('.jpg')[
```

ذخیره تمام تصاویر موجود در آرایهimages

وارد نمودن مسیر تصاویر دیتاست DRIVE که در فولدر test است.

```
img_path_test = r"C:\Users\Mahsa\Desktop\Tasks\8-#P3110-CT-
Scan_Segmentation\Code\MyCode\Dataset\test" # Enter Directory of all
images


folder_test = img_path_test

images_test = [os.path.join(root, filename(

          for root, dirs, files in os.walk(folder_test)
```

```
          for filename in files
          if filename.lower().endswith('.jpg')]
```

ذخیره تمام تصاویر موجود در آرایهٔimages_test

```
count=0
train_labels=[]
train_features_LCP_Net=[]
train_features_U_Net=[]
train_features_HDC_Net=[]
train_features_R2U_Net=[]
train_features_GC_Net=[]
train_features_DU_Net=[]
```

تعریف ماتریکس برای نگهداری ویژگیهای تصاویر دیتاست.

```
for image in images:
    img = Image.open(image)
    gray = img.convert('L')   # 'L' stands for 'luminosity'
    gray = np.asarray(gray)
    image=gray
    ##################Dilation - maximum_filter#######################
    # image_max is the dilation of im with a 20*20 structuring element
    # It is used within peak_local_max function
    image_max = ndi.maximum_filter(image, size=1, mode='constant')
    # Compute the Canny filter for two values of sigma
    image_lcp_net = feature.canny(image_max)
    # Make Embeddings Boundaries for Segmentation Result For LCP-Net
    prediction_image_lcp = color.label2rgb(image_lcp_net, image)
    train_features_LCP_Net.append(prediction_image_lcp)
```

```
    # Compute the Canny filter for two values of sigma
    edges_U_Net = feature.canny(image)
    # Make Boundaries for Segmentation Result For U-Net
    prediction_image_unet = color.label2rgb(edges_U_Net, image)
    train_features_U_Net.append(prediction_image_unet)
```

آموزش تصاویر و خواندن ویژگیهای تصاویر در حلقه با استفاده از روشهای مطرح شده ال سی پی نت.

```
distance_pred = []
distance_image = []

for image_test in images_test:
    img = Image.open(image_test)
    gray = img.convert('L')   # 'L' stands for 'luminosity'
    gray = np.asarray(gray)
    image1=gray
    for train_feature in train_features_LCP_Net:
        image2 = train_feature
        image2 = image2[:, :, 0]
        n_m= compare_images(image1, image2)
        distance_pred.append(n_m)
        distance_image.append(image2)
```

مقایسه با تصاویر تست ست با استفاده از فانکشنهای ذیل:

مقایسه تصاویر با استفاده از محاسبه فاصله منهتن دو تصویر.

```
Def compare_images(img1, img2):
    # normalize to compensate for exposure difference, this may be
unnecessary
```

```
    # consider disabling it
    img1 = normalize(img1)
    img2 = normalize(img2)
    diff = img1 - img2  # elementwise for scipy arrays
    m_norm = np.sum(abs(diff))  # Manhattan norm
    return m_norm
```

مبدل نمودن تصویر به خاکستری

```
def to_grayscale(arr):
    "If arr is a color image (3D array), convert it to grayscale (2D
array)."
    if len(arr.shape) == 3:
        return average(arr, -1)  # average over the last axis (color
channels)
    else:
        return arr
```

نرمالیزه کردن

```
def normalize(arr):
    rng = arr.max()-arr.min()
    amin = arr.min()
    return (arr-amin)*255/rng
```

انتخاب نزدیک ترین تصویر به تصویر اصلی با مینیمایز کردن ماتریکس تصاویر.

```
min_hist=min(distance_pred)
min_hist_index=distance_pred.index(min(distance_pred))
# print("{:.2f}".format(round(min_hist, 2)))
# print(min_hist)
image_pred=distance_image[min_hist_index]
```

نمایش تصویر اصلی و تصویر بدست آمده.

```
img_test = Image.open(r'C:\Users\Mahsa\Desktop\Tasks\8-#P3110-CT-
Scan_Segmentation\Code\MyCode\Dataset\test\image3.jpg')
```

```
# display prediction results
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(4, 2))
ax[0].imshow(img_test, cmap='gray')
ax[0].set_title('test image', fontsize=15)
ax[1].imshow(image_pred, cmap='gray')
ax[1].set_title('predicted image', fontsize=15)
for a in ax:
    a.axis('off')
fig.tight_layout()
plt.show()
```

Metrics Parameter

Cross Entropy

عدم نظم یا پیش بینی کاهش تدریجی به بی نظمی.

Cross Entropy Loss:

$$L(\Theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i)$$

```
from sklearn.metrics import log_loss
cross_entropy_lcp=log_loss(img_test, image_pred)
img_test = img_test[:, :, 0]
cross_entropy_lcp=np.multiply(img_test ,np.log(image_pred))
print(cross_entropy_lcp)
```

DDCLoss – Distance Deviation Loss Function

```
###############Distance Deviation#############
from sklearn.metrics import hamming_loss
# DDCLoss_lcp=hamming_loss(img_test, image_pred)
DDCLoss_lcp=np.std(cross_entropy_lcp)
print(DDCLoss_lcp)
```

```
from sklearn.metrics import explained_variance_score

deviation=explained_variance_score(img_test, image_pred)

print(deviation)
```

$$DDCLoss_{p_{h,w}} = -\sum_{k=0}^{N} (2d_k)^{\beta}(1-c_k)^{\alpha}\log(1-d_k)$$

$$DDCLoss = \frac{\sum_{h=0}^{H-1}\sum_{w=0}^{W-1} DDCLoss_{p_{h,w}}}{H \cdot W}$$

یک نسخه بهبود یافته از آنتروپی، بر اساس اطمینان از نتایج پیش بینی و فاصله انحراف است وبه عنوان یکی از توابع هزینه کارآمد برای بهینه سازی مدل پیشنهاد می کنیم. عملکرد این تابع با در نظر گرفتن اطمینان از پیکسل های پیش بینی شده و فاصله انحراف از برچسب به عنوان عوامل تنظیم کننده ، وزن پیکسل های نمونه را به صورت کارآمدی تنظیم می کند ، که این عمل، مدل را قادر می سازد تمرکز بیشتری در بهینه سازی پیکسل های نمونه داشته باشد که دارای برچسب نامناسب یا نامشخص در حین آموزش هستند.

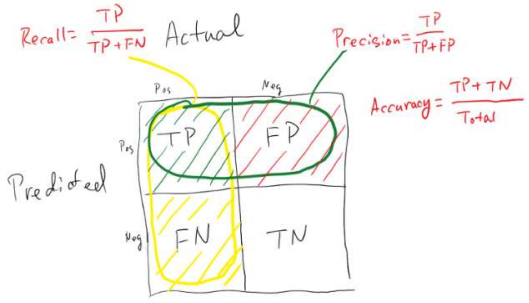
Focal Loss (FL) تابع هزینه

```
###############Focal Loss###############

from sklearn.metrics import precision_recall_fscore_support

focal_Loss=precision_recall_fscore_support(img_test, image_pred,
average='macro')

print(focal_Loss)
```

Accuracy

دقت Accuracy

به طور کلی، دقت به این معناست که مدل تا چه اندازه خروجی را درست پیش‌بینی می‌کند. با نگاه کردن به دقت ، بلافاصله می‌توان دریافت که آیا مدل درست آموزش دیده است یا خیر و کارآیی آن به طور کلی چگونه است. اما این معیار اطلاعات جزئی در مورد کارآیی مدل ارائه نمی‌دهد.



Accuracy = (TP+TN) / (TP+FN+FP+TN)

```
############Accuracy#########################3
from sklearn.metrics import accuracy_score
accuracy_score_lcp=accuracy_score(img_test, image_pred)
print(accuracy_score_lcp)
```

AUC

```
############AUC#############################
from sklearn.metrics import roc_auc_score
roc_auc_score=roc_auc_score(img_test, image_pred)
```

MIoU

Mean Intersection over Union = mIou

میانگین نقاط مشترک در سطوح متقاطع

$$mIoU = \frac{1}{N}\sum_{i=1}^{N}\frac{p_{ii}}{\sum_{j=1}^{N}p_{ij} + \sum_{j=1}^{K}p_{ji} - p_{ii}}$$

```
######### MIoU ##########
from sklearn.metrics import jaccard_similarity_score
jac = jaccard_similarity_score(img_test, image_pred, Normalize =
True/False)
```

Precision

وقتی که مدل نتیجه را مثبت (positive) پیش‌بینی می‌کند، این نتیجه تا چه اندازه درست است؟ زمانی که ارزش false positivesبالا باشد، معیار صحت، معیار مناسبی خواهد بود. فرض کنید، مدلی برای تشخیص سرطان داشته باشیم و این مدل Precision پایینی داشته باشد. نتیجه این امر این است که این مدل، بیماری بسیاری از افراد را به اشتباه سرطان تشخیص می‌دهد. نتیجه این امر استرس زیاد، آزمایش‌های فراوان و هزینه‌های گزافی را برای بیمار به دنبال خواهد داشت.

Precision =TP / (TP+FP)

حساسیت Sensitivity

در واقع، «حساسیت» معیاری است که مشخص می‌کند دسته‌بند، به چه اندازه در تشخیص تمام افراد مبتلا به بیماری موفق بوده‌است. همانگونه که از رابطه فوق مشخص است، تعداد افراد سالمی که توسط دسته‌بند به اشتباه به عنوان فرد بیمار تشخیص داده شده‌اند، هیچ تاثیری در محاسبه این پارامتر ندارد و در واقع زمانی که پژوهشگر از این پارامتر به عنوان پارامتر ارزیابی برای دسته‌بند خود استفاده می‌کند، هدفش دستیابی به نهایت دقت در تشخیص نمونه‌های کلاس مثبت است.

Specificity

در نقطه مقابل این پارامتر، ممکن است در مواقعی دقت تشخیص کلاس منفی حائز اهمیت باشد. از متداول‌ترین پارامترها که معمولا در کنار حساسیت بررسی می‌شود، پارامتر خاصیت(Specificity) ، است که به آن «نرخ پاسخ‌های منفی درست (True Negative Rate) «نیز می‌گویند. خاصیت به معنی نسبتی از موارد منفی است که آزمایش آن‌ها را به درستی به عنوان نمونه منفی تشخیص داده است. این پارامتر به صورت زیر محاسبه می‌شود .زمانی که ارزش false negatives بالا باشد، معیارRecall ، معیار مناسبی خواهد بود. فرض کنیم مدلی برای تشخیص بیماری کشنده ابولا داشته باشیم. اگر این مدل Recall پایینی داشته باشد چه اتفاق خواهد افتاد؟ این مدل افراد زیادی که آلوده به این بیماری کشنده هستند را سالم در نظر می‌گیرد و این فاجعه است. نسبت مقداری موارد صحیح طبقه‌بندی شده توسط الگوریتم از یک کلاس به تعداد موارد حاضر در کلاس مذکور که به‌صورت زیر محاسبه می‌شود:

$$\text{Recall} = \text{Sensitivity} = (\text{TPR}) = \text{TP} / (\text{TP}+\text{FN})$$

$$\text{Specificity (TNR)} = \text{TN} / (\text{TN}+\text{FP})$$

```
######### Sensitivity #############

# Note that in binary classification, recall of the positive class is also
known as “sensitivity”

# precision    recall  f1-score

from sklearn.metrics import classification_report

classification_report(img_test, image_pred)
```

Save Result on csv file

```
with open('LCP_Net_Cross_Entropy.csv', 'w', encoding='UTF8') as f1:

    csvcreator_x = csv.writer(f1)

    csvcreator_x.writerow(distance_pred)
```