الجلسة السابعة

Texture Analysis

تحليل النسيج

1.1 توليد النسيج Texture Synthesis

المقصود بهذه العملية استخدام نسيج بسيط والانطلاق منه لتوليد نسيج بحجم أكبر وبنفس مواصفات النسيج الأصلي.

1.2 تصنيف الصور بالاعتماد على النسيج Texture-based classification

يتم في هذا النوع من التطبيقات استخلاص السمات النسيجية من الصورة باستخدام إحدى خوارزميات استخلاص سمات النسيج مثل Local Binary Pattern LBP ثم يتم تحويل مصفوفة السمات الناتجة إلى شعاع من القيم باستخدام إحدى طرق تقليل الأبعاد مثل خوارزمية PCA أو من خلال الهيستوغرام. بعدها يتم استخلاص سمات صور التدريب جميعاً وتدريب مصنف Machine Learning عليها ليتعلم ربط هذه السمات النسيجية للصور مع صنفها الصحيح.

1.3 التنفيذ العملي:

نفذ الكود التالي:

```
from skimage import feature
from skimage.io import imread
import numpy as np
import os
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from skimage.transform import resize
import matplotlib.pyplot as plt
```

تضمين مكتبة Skimage بسبب الحاجة لخوارزمية LBP

تضمين مكتبة sklearn بسبب الحاجة لخوارزميات تعلم الآلة اللازمة لعملية التصنيف

RandomForestClassifier مصنف تعلم آلة

Train_test_split لتقسيم مجموعة الصور لتدريب واختبار

من حزمة transform نستخدم تابع resize

```
# Define LBP parameters
radius = 8
n_points = 6* radius
```

تحديد بارامترات خوارزمية LBP وهي نصف قطر دائرة التركيز أو حجم الجوار المراد العمل ضمنه وعدد النقاط المأخوذة من ضمن الدائرة بعين الحسبان

```
# Initialize an empty array for storing LBP histograms and labels
histograms = []
labels = []
# Define the classes
classes = ['Cinder','Cracked', 'Sandy']
# Loop over the training images
for soil_type in classes:
  for i in range(1, 31):
    # Load and resize the image
    image = imread(f'{soil_type}/{i}.jpg', as_gray=True)
    image = resize(image, (200, 200))

    # Compute LBP features
    lbp = feature.local_binary_pattern(image, n_points, radius, method="uniform")

  # Compute histogram of LBP features
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points +1), range=(0, n_points ))
```

تهيئة مصفوفة الهيستوغرام ومصفوفة Labels

تحديد الأصناف الثلاثة (أصناف نسيج التربة المدروسة)

يتم في الحلقة قراءة صور كل مجلد حيث أن كل مجلد يمثل أحد أصناف التربة الثلاث ثم يتم استخدام حجم محدد وهو 200*200 لجميع الصور والسبب أن أبعاد الصور مختلفة وليست مضبوطة
الحلقة الداخلية تتكرر 30 مرة لأن كل مجلد يتضمن 30 صورة

حساب سمات LBP (أو سمات النسيج المحلي) ونمرر لهذا التابع الصورة المراد حساب سماتها النسيجية وعدد النقاط ونصف قطر دائرة التركيز

سنحصل على مصفوفة سمات LBP لذلك لنتمكن من إدخالها لمصنف machine learning يجب تحويلها لشعاع 1D ولذلك يتم حساب هيستوغرام مصفوفة السمات
سنستخدم عدد نقاط هيستوغرام مساوي لعدد نقاط البيانات المدروسة
سنحصل من خوارزمية LBP على 48 قيمة إضافة للصفر
نحصل من خوارزمية LBP على 48 سمة مميزة والصفر يعني غياب السمة

```
[ 0.  2. 49. ... 0. 0. 0.]
[ 0.  0.  1. ... 0. 0. 0.]
[ 0.  0.  0. ... 1. 0. 0.]]
[[12. 11.  9. ... 49. 49.  1.]
[10.  9. 49. ... 49. 49. 49.]
[49. 49.  5. ... 49. 49. 49.]
```

```
# Normalize the histogram
hist = hist.astype("float")
hist /= (hist.sum())
```

بعدها يتم تطبيق Histogram Normalization من خلال تقسيم قيم الهيستوغرام على عددها للحصول على قيم ضمن المجال 0-1

```
# Append the histogram to the list
histograms.append(hist)
```

في نهاية كل تكرار في الحلقة يتم إضافة قيم الهيستوغرام (شعاع) إلى مصفوفة الهيستوغرامات لكل الصور، ثم يتم إضافة رقم الصنف الذي تنتمي له العينة لمصفوفة Labels

```
# Append the label to the list
labels.append(soil_type)
```

نحصل بنهاية الحلقة على عينات التدريب الخاصة بنسيج التربة لكل الأصناف حيث كل عينة تتضمن هيستوغرام الصورة وصنف الصورة (دخل-خرج)

```
# Convert the histograms and labels to NumPy arrays
histograms = np.array (histograms)
labels = np.array(labels)
```

تحويل مصفوفات الهيستوغرام والأصناف إلى مصفوفات numpy.array لنتمكن من إدخالها لخوارزمية تعلم الآلة RF

```
# Split the data into training and testing sets
(trainData, testData, trainLabels, testLabels) = train_test_split(histograms, labels, test_size=0.3, random_state=42)
```

مثل أي عملية تصنيف نقسم البيانات التي حصلنا عليها إلى قسمين تدريب train بنسبة 70% لندرب بها مصنف RF و اختبار test بنسبة 30% لنختبر المصنف بعد التدريب للتحقق من الأداء

```
# create a Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

بناء مصنف RF باستخدام 100 شجرة قرار ويتم تهيئة بارامترات المصنف بشكل عشوائي ولمدة تنفيذ مقدارها 42 مرة سنحصل على نفس التهيئة العشوائية وبالتالي أداء متطابق للمصنف

```
# fit the model (example)
model.fit(trainData, trainLabels)
```

تدريب نصنف RF باستخدام تابع fit حيث نمرر له بيانات التدريب (دخل-خرج) وهي سمات الهيستوغرام مع صنف كل منها لجميع عينات التدريب للأصناف الثلاثة

```
# Predict the labels for the test data
predictions = model.predict(testData)
```

بعد انتهاء عملية التدريب يتم اختبار الموديل المدرب باستخدام تعليمة model.predict حيث نمرر للموديل المدرب سمات عينات الاختبار وهنا نمرر فقط السمات ونحصل من المصنف على predictions تمثل أرقام الأصناف التي تم التعرف عليها

```
# Print the classification report
print(classification_report(testLabels, predictions))
print(confusion_matrix(testLabels, predictions))
```

نطبع نتيجة الاختبار وهي تقرير التصنيف ومصفوفة التشتت confusion matrix والهدف منها معرفة دقة التصنيف النهائية

```
# Load the test image
test_image = imread('test1.jpg', as_gray=True)
test_image_C = imread('test1.jpg', as_gray=False)
```

والآن نختبر المصنف المدرب بسمات LBP النسيجية على صور اختبار تم تحميلها من الانترنت هنا نقرأ صورة الاختبار ثم نستخلص سماتها بنفس طريقة استخلاص سمات صور التدريب ثم نمرر السمات التي نتجت عن الهيستوغرام للموديل المدرب ونحصل منه على prediction لعينة الاختبار ثم نعرض صورة الاختبار ونضع نتيجة الاختبار كعنوان للصورة (صنف الصورة)

```
# Compute LBP features for the test image
test_lbp = feature.local_binary_pattern(test_image, n_points, radius, method="uniform")

# Compute histogram of LBP features for the test image
(test_hist, _) = np.histogram(test_lbp.ravel(), bins=np.arange(0, n_points + 1), range=(0, n_points ))
# Normalize the histogram
test_hist = test_hist.astype("float")
test_hist /= (test_hist.sum() + 1e-7)
# Predict the class of the test image
test_prediction = model.predict(test_hist.reshape(1, -1))
# Display the test image and the prediction
plt.imshow(test_image_C, cmap='gray')
plt.title(f'Predicted class: {test_prediction[0]}')
plt.axis('off')
plt.show()
```

والآن نقرأ نتائج عملية التدريب والاختبار لموديل RF

```
              precision    recall  f1-score   support

      Cinder       1.00      0.89      0.94         9
     Cracked       0.89      0.89      0.89         9
       Sandy       0.90      1.00      0.95         9

    accuracy                           0.93        27
   macro avg       0.93      0.93      0.93        27
weighted avg       0.93      0.93      0.93        27

[[8 1 0]
 [0 8 1]
 [0 0 9]]
```

نلاحظ أننا توصلنا لدقة 93% في تمييز عينات الاختبار للأصناف الثلاثة مع ملاحظة أنَّ الصنف الأسوأ كان Cracked.

تحسب قيم Precision بمسح عمودي لمصفوفة confusion matrix في حين تحسب قيم recall بمسح أفقي لها.

مثلاً أول صنف Cinder يمتلك قيمة (100%) 1=8/8 = precision و (89%) 0.89=8/9 = recall.

تكمن المشكلة في أن هناك عينة تنتمي للصنف الأول لكن تم تصنيفها ضمن الصنف الثاني (تم رفض عينة بشكل خاطئ).

وهناك عينة من الصنف الثاني لكن تم تصنيفها ضمن الصنف الثالث. (تم رفض عينة بشكل خاطئ).

الصنف الثالث تم تصنيف عيناته بشكل صحيح.

وهذا ما جعل قيمة recall للصنفين الأول والثاني 89% وللصنف الثالث 100% (تقل recall بسبب الرفض الخاطئ FN).

بالنسبة لمفهوم precision بالعكس فالصنف الأول لم يقبل أي عينة خارجة عنه لذلك قيمة precision لديه 100%، في حين تم قبول عينة تنتمي للصنف الأول ضمن الصنف الثاني وعينة تنتمي للصنف الثاني ضمن الثالث ما جعل precisionلهما تقل وتصبح 89% و90% (تقل precision بسبب القبول الخاطئ FP).

والآن نختبر الموديل الذي قمنا بتدريبه على عينات اختبار من الأصناف الثلاثة لنتحقق من إمكانية تمييزه للأصناف الثلاثة اعتماداً على تحليل النسيج الخاص بها:

Predicted class: Cinder

Predicted class: Cracked

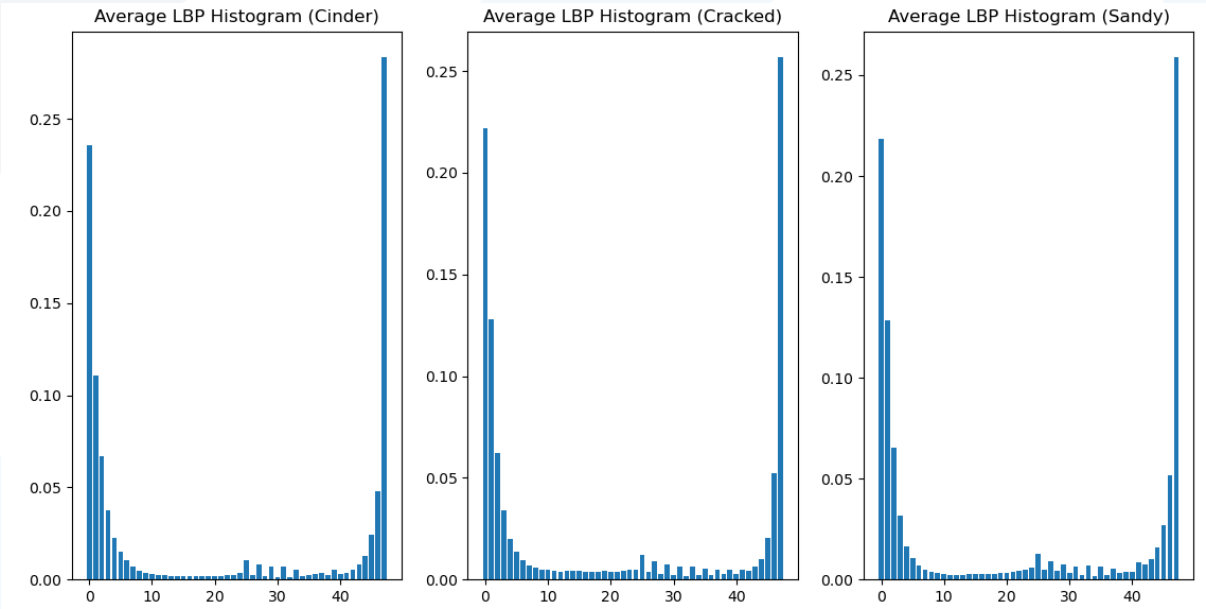
في نهاية الكود قمنا بعرض الهيستوغرام الوسطي لجميع عينات مجموعة الصور لكل صنف من الأصناف الثلاثة لنرى إمكانية التمييز بين سمات الأصناف الثلاثة:

1.4 كود إضافي (غير مطلوب اطلاع فقط):

لرسم سمات LBP بعد توليد هيستوغرامها لمقارنة السمات من الأصناف الثلاثة.

```
#### let's show the LBP features

# Assuming histograms and labels are numpy arrays

histograms = np.array(histograms)

labels = np.array(labels)


# Get unique classes

classes = np.unique(labels)


# Plot histogram for each class
```

```
plt.figure(figsize=(10, 5))
for i, soil_type in enumerate(classes):
    # Get histograms for this class
    class_histograms = histograms[labels == soil_type]

    # Average histograms (you could also choose to plot them individually)
    avg_hist = np.mean(class_histograms,axis=0)

    # Plot
    plt.subplot(1, len(classes), i+1)
    plt.bar(np.arange(len(avg_hist)), avg_hist)
    plt.title(f'Average LBP Histogram ({soil_type})')

plt.show()
```



صحيح أن الهيستوغرام الوسطي يتشابه في بعض المناطق إلا أنه مختلف في مناطق أخرى وهو ما يعطي إمكانية التمييز ويساعد خوارزمية تعلم الآلة Machine Learning Algorithm على تصنيف العينات للصنف الصحيح.