Ciuculan Stefan
333AA

Raport de Proiect

Clasificare Imagini Utilizând Metoda SVM cu Extracție de Caracteristici HOG

Descrierea Proiectului

Proiectul constă în dezvoltarea unui sistem de clasificare a imaginilor în categoriile "urban" și "rural" folosind o metodă bazată pe Support Vector Machines (SVM) și extracția de caracteristici HOG (Histogram of Oriented Gradients). SVM este o tehnică de învățare supervizată potrivită pentru clasificarea de date, în timp ce HOG este o tehnică eficientă de extragere a caracteristicilor din imagini.

Obținere și Organizare Set de Date

Setul de date a fost obținut prin colectarea imaginilor "**urban**" și "**rural**" din surse diverse. Imaginile de antrenament, în număr de 100 (50 + 50), au fost organizate într-un director "**train**" și separate pe 2 subdirectoare "**urban**" și "**rural**". Ulterior, am separat setul de antrenament în **training set (80%)** și **validation set (20%).** Datele de test, în număr de 20 (11 rurale + 9 urbane), au fost puse într-un director numit "**test_accuracy**" și separate la fel ca în directorul de antrenament. Am avut nevoie de această a doua separare pentru ca, la evaluarea rezultatelor, să putem fi conștienți de veridicitatea lor și de acuratețea algoritmului.

Biblioteci Python Utilizate

Proiectul a fost implementat folosind următoarele biblioteci Python:

- OpenCV pentru manipularea imaginilor
- NumPy pentru manipularea datelor
- scikit-learn pentru implementarea SVM și funcțiile de evaluare
- Matplotlib pentru afișarea imaginilor și rezultatelor
- Seaborn pentru afișarea matricelor de confuzie sub formă de **heatmap**

- Pandas pentru crearea DataFrame-urilor matricelor de confuzie

```
import os
import cv2
import numpy as np
from sklearn import svm
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from skimage.feature import hog
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

Această secțiune importă modulele necesare pentru manipularea imaginilor, implementarea SVM, prelucrarea datelor și extragerea caracteristicilor HOG (Histogram of Oriented Gradients).

```
def load_images_from_folder(folder, target_shape=None):
    images = []
    labels = []
    for subfolder in os.listdir(folder):
        subfolder_path = os.path.join(folder, subfolder)
        if os.path.isdir(subfolder_path):
            for filename in os.listdir(subfolder_path):
                if filename.endswith(('.jpeg', '.jpg')):
                    img = cv2.imread(os.path.join(subfolder_path, filename))
                    if img is not None:
                        if target_shape is not None:
                            img = cv2.resize(img, target_shape)
                        images.append(img)
                        labels.append(subfolder)
                    else:
                        print(f"Warning: Unable to load {filename}")
    return images, labels
```

Funcția **load_images_from_folder** încarcă imaginile dintr-un director specific și extrage etichetele (labels) asociate. Se redimensionează imaginile la dimensiunea specificată dacă este specificată.

```python
def extract_hog_features(images):
    features = []
    for img in images:
        img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        features.append(hog(img_gray, block_norm='L2-Hys'))
    return np.array(features)
```

Funcția **extract_hog_features** primește o listă de imagini și returnează caracteristicile HOG (Histogram of Oriented Gradients) ale acestora. Se convertește fiecare imagine la tonuri de gri înainte de extragerea caracteristicilor HOG.

```python
data_folder = './train'

images, labels = load_images_from_folder(data_folder, target_shape=(200, 200))
```

Se încarcă imaginile de antrenare din directorul specificat (**./train**) și se apelează funcția **load_images_from_folder** pentru a obține imagini și etichetele corespunzătoare.

```python
X_train_features = extract_hog_features(images)
```

Se extrag caracteristicile HOG pentru setul de antrenare folosind funcția **extract_hog_features**.

```python
X_train, X_val, y_train, y_val = train_test_split(images, labels, test_size=0.2, random_state=42)
```

Se împart datele în două subseturi un set de antrenare, folosit pentru instruirea modelului, și un set de validare, care va fi folosit pentru a evalua performanța modelului.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_features)
```

Se scalează datele pentru a le aduce la aceeași scară folosind **StandardScaler** din **sklearn**.

```
clf = svm.SVC(kernel='linear', C=1, random_state=22)
clf.fit(X_train_scaled, y_train)
```

Se antrenează un clasificator SVM (Support Vector Machine) cu un kernel liniar, folosind datele de antrenare scalate.

```
X_val_features = extract_hog_features(X_val)
X_val_scaled = scaler.transform(X_val_features)
```

Se pregătește setul de validare pentru a fi utilizat în procesul de evaluare a modelului.

```
y_val_pred = clf.predict(X_val_scaled)
accuracy_val = accuracy_score(y_val, y_val_pred)
```

Se obține o măsură a calității modelului pe datele de validare, indicând cât de bine generalizează modelul la date noi și necunoscute.

```
print("\nValidation Labels:")
print(y_val)
print("Predicted Labels:")
print(y_val_pred)

print(f"Accuracy on Validation Images: {accuracy_val * 100:.2f}%")
```

Se afișează etichetele imaginilor din setul de validare, urmate de etichetele prezise, ca mai apoi să putem observa relevanța lor în calculul scorului de acuratețe a algoritmului.

```
conf_matrix_val = confusion_matrix(y_val, y_val_pred)
df_conf_matrix_val = pd.DataFrame(conf_matrix_val, index=np.unique(labels), columns=np.unique(labels))

plt.figure(figsize=(10, 7))
sns.heatmap(df_conf_matrix_val, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title("Confusion Matrix - Validation Set")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Se creează și se afișează matricea de confuzie pentru imaginile din setul de validare.

```
test_folder = './test_accuracy'
test_images, test_labels = load_images_from_folder(test_folder, target_shape=(200, 200))
```

Se încarcă imaginile de testare din directorul specificat (**./test_accuracy**) și se apelează funcția **load_images_from_folder** pentru a obține imagini și etichetele corespunzătoare.

```
X_test_images_features = extract_hog_features(test_images)
X_test_images_scaled = scaler.transform(X_test_images_features)
```

Se extrag caracteristicile HOG pentru imaginile de testare și se scalează, utilizând același scalator utilizat anterior pe setul de antrenare.

```
y_test_images_pred = clf.predict(X_test_images_scaled)
```

Se prezic etichetele pentru imaginile de testare folosind clasificatorul SVM antrenat.

```
conf_matrix_test = confusion_matrix(test_labels, y_test_images_pred)
df_conf_matrix_test = pd.DataFrame(conf_matrix_test, index=np.unique(labels), columns=np.unique(labels))

plt.figure(figsize=(10, 7))
sns.heatmap(df_conf_matrix_test, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title("Confusion Matrix - Test Set")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Se realizează și se afișează într-o nouă figura matricea de confuzie pentru setul de test, ca mai devreme, in cazul setului de validare.

```
print("Test Labels:", test_labels)
print("Predicted Labels:", y_test_images_pred)

accuracy_test_images = accuracy_score(test_labels, y_test_images_pred)
print(f"Accuracy on Test Images: {accuracy_test_images * 100:.2f}%")
```

Etichetele imaginilor de test, cât și etichetele prezise sunt afișate, urmate de acuratețea modelului antrenat.

```
for i, (test_image, predicted_label) in enumerate(zip(test_images, y_test_images_pred)):

    plt.figure()
    plt.imshow(cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB))

    plt.title(f"Test Image {i+1}-=={predicted_label.upper()}==")

    plt.axis('off')
    plt.show()
```

Se afișează imaginile de testare, împreună cu etichetele prezise, pentru a evalua vizual performanța modelului.

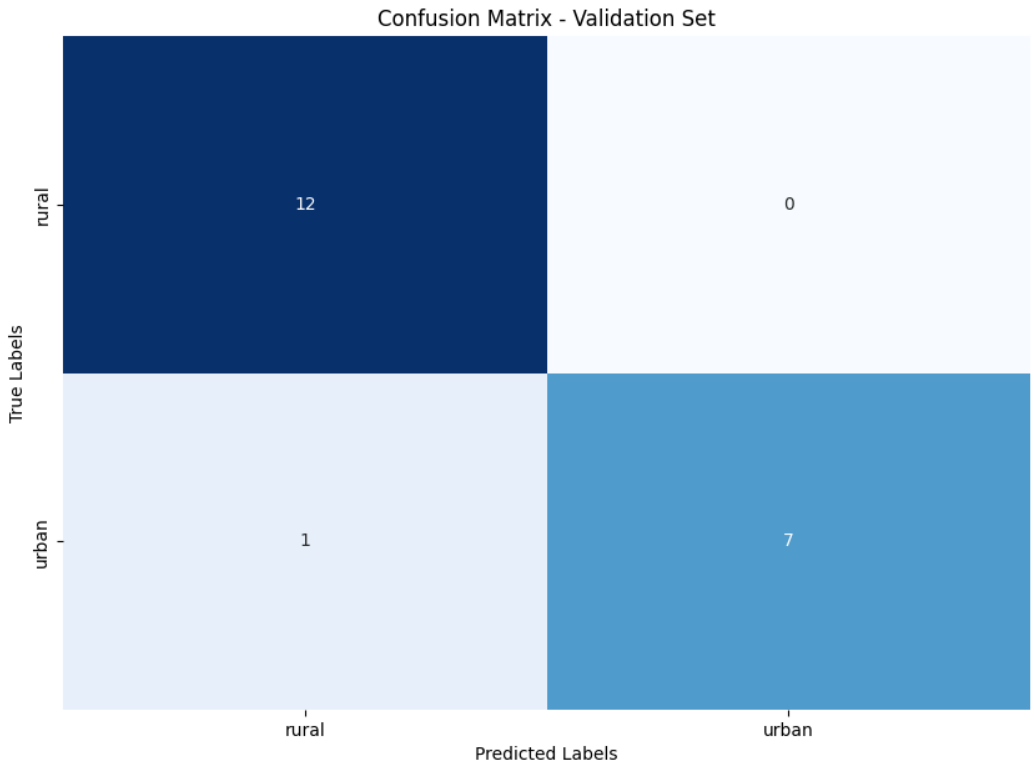
Ciuculan Stefan
333AA

Rezultate Obținute

În primă instanță, s-au afișat etichetele imaginilor din setul de validare, alături de cele prezise și de scorul de acuratețe calculat, de 95%.

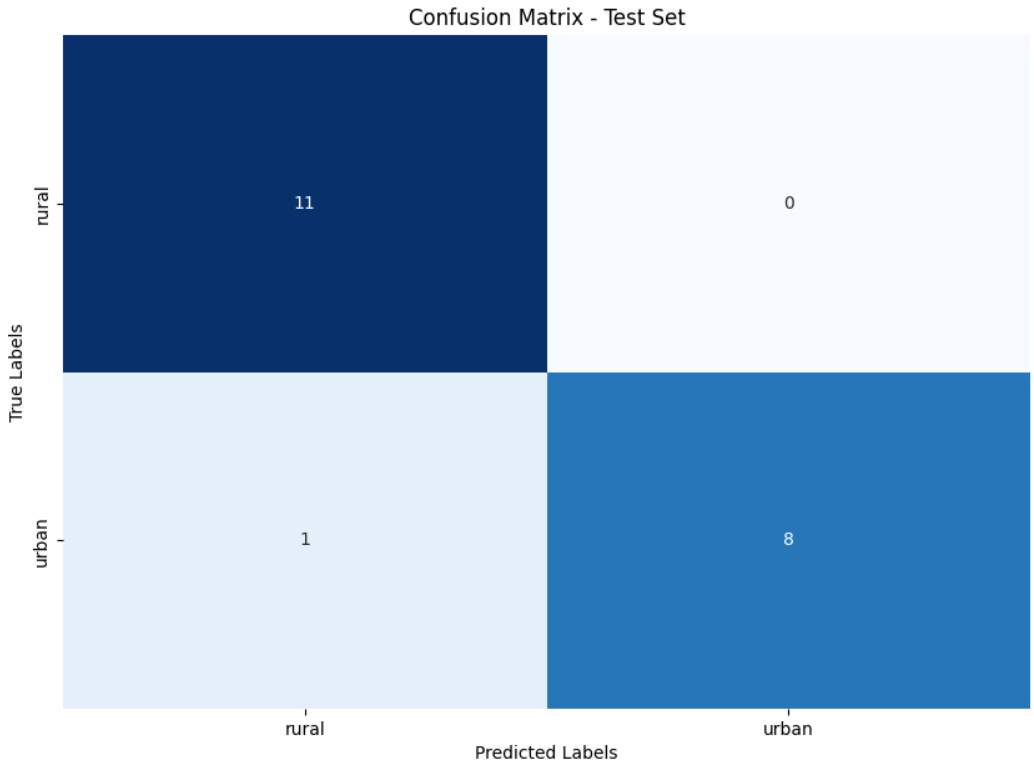
```
Validation Labels:
['urban', 'urban', 'urban', 'rural', 'rural', 'rural', 'rural', 'urban', 'rural', 'rural', 'rural', 'rural', 'urban', 'rural', 'urban', 'rural', 'urban', 'urb
an', 'rural', 'rural']
Predicted Labels:
['urban' 'urban' 'urban' 'rural' 'rural' 'rural' 'rural' 'urban' 'rural'
 'rural' 'rural' 'rural' 'rural' 'rural' 'urban' 'rural' 'urban' 'urban'
 'rural' 'rural']
Accuracy on Validation Images: 95.00%
```

Totodată, apare și matricea de confuzie din acest caz.



Ciuculan Stefan
333AA

Ulterior, apare și matricea de confuzie pentru imaginile de test.



Apoi, am afișat etichetele de test reale, precum și pe cele prezise. Algoritmul antrenat cu setul de date a obținut o acuratețe de 95% pe setul de testare. Mai jos, în dreptul fiecărei imagini de test se observă eticheta prezisă, alături de cea reală.

```
Test Labels: ['rural', 'rural', 'rural', 'rural', 'rural', 'rural', 'rural', 'rural', 'rural', 'rural', 'rural', 'urban', 'urban', 'urban', 'urban', 'urban', 'urban', 'urban', 'urban
', 'urban']
Predicted Labels: ['rural' 'rural' 'rural' 'rural' 'rural' 'rural' 'rural' 'rural' 'rural'
 'rural' 'rural' 'urban' 'urban' 'urban' 'urban' 'rural' 'urban' 'urban'
 'urban' 'urban']
Accuracy on Test Images: 95.00%
Test Image 1: Predicted - rural, Actual - rural
Test Image 2: Predicted - rural, Actual - rural
Test Image 3: Predicted - rural, Actual - rural
Test Image 4: Predicted - rural, Actual - rural
Test Image 5: Predicted - rural, Actual - rural
Test Image 6: Predicted - rural, Actual - rural
Test Image 7: Predicted - rural, Actual - rural
Test Image 8: Predicted - rural, Actual - rural
Test Image 9: Predicted - rural, Actual - rural
Test Image 10: Predicted - rural, Actual - rural
Test Image 11: Predicted - rural, Actual - rural
Test Image 12: Predicted - urban, Actual - urban
Test Image 13: Predicted - urban, Actual - urban
Test Image 14: Predicted - urban, Actual - urban
Test Image 16: Predicted - rural, Actual - urban
Test Image 17: Predicted - urban, Actual - urban
Test Image 18: Predicted - urban, Actual - urban
Test Image 19: Predicted - urban, Actual - urban
Test Image 20: Predicted - urban, Actual - urban
```

Ciuculan Stefan
333AA

Concluzii și observații:

Proiectul a demonstrat că metoda SVM, împreună cu caracteristicile HOG, poate fi utilizată cu succes pentru clasificarea imaginilor în scenarii de tip "urban" sau "rural". Eficiența modelului poate fi îmbunătățită prin ajustarea parametrilor și prin extinderea setului de date cu mai multe exemple.

De altfel, se observă scorul de acuratețe identic pe setul de validare cu setul de test. În plus, deducem erori strict în cazul imaginilor cu peisaje urbane, chiar pe setul de test singura imagine prezisă greșit fiind aceasta: