EE-655 HW-1

Naman Mohan Singh

Introduction

- Dataset: MNIST (handwritten digits in JPG format).
- Task: Apply HOG feature extraction and classify images using ML algorithms.
- Objective: To achieve the best accuracy by experimenting with HOG and ML parameters.

Dataset Preparation

- Used google colab to download the dataset and resized the images to 28x28 grayscale images.

HOG Feature Extraction

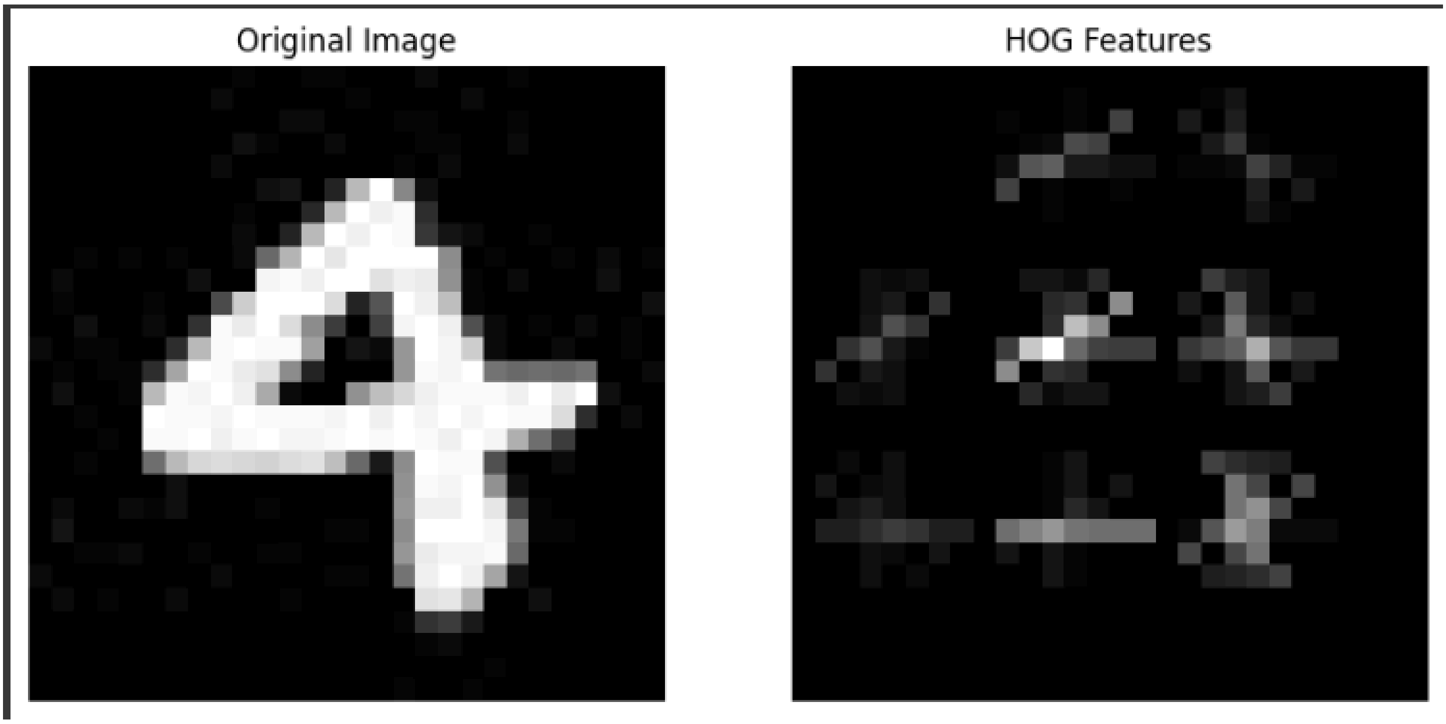
- Tried to extract hog features from the image :

1. By Defining a HOG function from scratch (with a little help from internet)

```
import os
import cv2
import numpy as np

# Computing gradients
def compute_gradients(image):
    # Sobel kernels for gradients
    gx_kernel = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
    gy_kernel = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

    # Convolving the image with kernels
    gx = cv2.filter2D(image, -1, gx_kernel)
    gy = cv2.filter2D(image, -1, gy_kernel)

    # Computing magnitude and angle
    magnitude = np.sqrt(gx**2 + gy**2)
    angle = np.arctan2(gy, gx) * (180 / np.pi)  # Convert to degrees
    angle[angle < 0] += 180  # Make angles positive
    return magnitude, angle
```

```python
def extract_hog_features_manual(image, cell_size=8, block_size=3, bins=15):
    # Step 1: Computing  gradients
    magnitude, angle = compute_gradients(image)

    # Step 2: Dividing the image into cells
    h, w = image.shape
    cell_rows, cell_cols = h // cell_size, w // cell_size
    histograms = np.zeros((cell_rows, cell_cols, bins))

    for i in range(cell_rows):
        for j in range(cell_cols):
            # Extracting a cell
            cell_magnitude = magnitude[i * cell_size:(i + 1) * cell_size, j * cell_size:(j + 1) * cell_size]
            cell_angle = angle[i * cell_size:(i + 1) * cell_size, j * cell_size:(j + 1) * cell_size]

            # Creating histogram
            hist = np.zeros(bins)
            bin_width = 180 // bins

            for row in range(cell_size):
                for col in range(cell_size):
                    mag = cell_magnitude[row, col]
                    ang = cell_angle[row, col]

                    # Determine the bin
                    bin_idx = int(ang // bin_width) % bins
                    hist[bin_idx] += mag

            histograms[i, j] = hist


    block_stride = cell_size
    blocks = []
    for i in range(cell_rows - block_size + 1):
        for j in range(cell_cols - block_size + 1):
            block = histograms[i:i + block_size, j:j + block_size].ravel()
            block /= np.sqrt(np.sum(block ** 2) + 1e-6)
            blocks.append(block)

    return np.hstack(blocks)
```

HOG features Visualized



2 . Used Sk.Image Library also To Improve Accuracy

```
def extract_hog_features(image):
    features, _ = hog(image, orientations=9, pixels_per_cell=(8, 8),
                      cells_per_block=(3, 3), visualize=True)
    return features
```

Experiments with HOG Parameters

- The initial parameters used (orientations=9, pixels_per_cell=(8,8), cells_per_block=(2,2)).

1. **Orientations**: Increased/decreased the number of bins for gradient directions (e.g., from 9 to 12).
2. **Pixels per Cell**: Tried smaller/larger cell sizes (e.g., (4,4) vs. (8,8)).
3. **Cells per Block**: Adjusted block size (e.g., (3,3) vs. (2,2)).

ML Classifiers Used :

SUPPORT VECTOR MACHINES

RANDOMFOREST CLASSIFIER

KNeighbours CLASSIFIER

RESULTS

The First Approach yielded a maximum accuracy of ***80.80% by using svm classifier***

```
[12]
    clf = RandomForestClassifier(n_estimators=100, random_state=42)
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 78.15%
```

```
[13] from sklearn.neighbors import KNeighborsClassifier
     knn = KNeighborsClassifier()
     knn.fit(X_train, y_train)
     y_pred = knn.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 76.54%
```

```
[14] from sklearn import svm
     model = svm.SVC(kernel='linear')
     model.fit(X_train, y_train)
     y_pred = model.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 80.80%
```

To improve the accuracy I used SkImage Library's Pre-Defined HOG function which yielded a maximum accuracy of ***95.85% on Kneighbours classifier***

```
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 95.39%
```

```
from sklearn import svm
model = svm.SVC(kernel='linear')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 95.59%
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 95.85%
```

These accuracies were achieved by tweaking the parameters of the HOG feature extractor!

HOG Parameters	Test Accuracy
Orientations = 12, Cell = 2x2	95.85%
Orientations = 9, Cell = 4x4	94.7%
Orientations = 9, Cell = 8x8	95.2%

To optimize HOG feature extraction, I experimented with the orientations, pixels_per_cell, and cells_per_block parameters. Increasing the number of orientations from 9 to 12 improved accuracy by capturing more detailed gradient directions. Reducing the cell size from 8x8 to 4x4 allowed finer spatial resolution but increased computation time. The best combination was orientations=12, pixels_per_cell=(8,8), and cells_per_block=(2,2), which yielded the highest test accuracy of 95.85%."

SUMMARY

The Best Classifier came out to be the SVMs

The Overall Best Test Accuracy was 95.85%

Link To Colab File : -

EE-655-/EE655_ASSN1.ipynb at main · naman065/EE-655-

NAMAN MOHAN SINGH

230678