

**LAPORAN TUGAS VISI KOMPUTER**  
**EKSTRAKSI FITUR MEGGUNAKAN HISTOGRAM OF ORIENTED**  
**GRADIENTS DAN KLASIFIKASI K-NEAREST NEIGHBORS UNTUK**  
**DATASET RAMBU LALU LINTAS**



**Disusun Oleh:**

Rizky Akhmad Fahreza

24060121130081

**Dosen Pengampu:**

Dr. Aris Sugiharto, S.Si., M.Kom.

NIP. 197108111997021004

**DEPARTEMEN INFORMATIKA**  
**FAKULTAS SAINS DAN MATEMATIKA**  
**UNIVERSITAS DIPONEGORO**  
**SEMARANG**

**2024**

## I. Latar Belakang

Teknologi pengenalan citra merupakan salah satu aspek penting dalam machine learning dan banyak digunakan dalam berbagai aplikasi. Salah satu tugas mendasar dalam pengolahan citra adalah klasifikasi gambar, yang melibatkan pengenalan pola visual dalam gambar untuk membedakan antara kategori yang berbeda. Pada tugas ini, dataset citra berisi dua kelas: gambar dengan objek tertentu (positif) dan gambar tanpa objek tersebut (negatif).

Ekstraksi fitur Histogram of Oriented Gradients (HOG) membantu dalam menangkap informasi tepi dan struktur dari gambar, yang kemudian digunakan oleh algoritma K-Nearest Neighbors (KNN) untuk melakukan klasifikasi. Latihan ini berfokus pada penerapan metode klasifikasi citra menggunakan teknik tersebut.

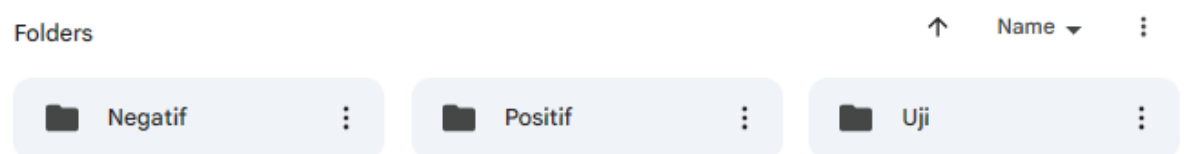
## II. Tujuan

1. Melatih kemampuan mahasiswa dalam menerapkan ekstraksi fitur HOG untuk menangkap pola visual dalam gambar.
2. Menggunakan algoritma K-Nearest Neighbors (KNN) untuk melakukan klasifikasi citra berdasarkan fitur yang diekstraksi.
3. Memberikan pemahaman dasar tentang proses klasifikasi citra dalam machine learning dengan dataset yang terdiri dari dua kelas (positif dan negatif).

## III. Metode Penelitian

### A. Dataset

Dataset yang digunakan pada penugasan kali ini adalah citra rambu-rambu lalu lintas.



Seperti yang dapat dilihat pada gambar di atas, dataset terbagi menjadi data train yang terdiri dari negatif (bukan rambu lalu lintas) dan positif (merupakan rambu lalu lintas). Selain itu juga terdapat data uji yang akan digunakan untuk menguji model yang dibuat sehingga keakuratan model tersebut dapat diketahui.

## B. Data Pre-Processing

Sebelum fitur-fitur pada data citra diekstraksi menggunakan HOG, data-data citra tersebut perlu melalui beberapa tahapan seperti mengubah tiap data yang tadinya memiliki dimensi RGB(Red, Green, Blue) menjadi BW(Black & White) dan memberikan label berupa 0 (negatif) dan 1 (positif).

Fungsi untuk memuat gambar dari dataset:

```
def load_images_from_folder(folder, label):
    images = []
    labels = []
    for filename in os.listdir(folder):
        img = imread(os.path.join(folder, filename),
as_gray=True)
        if img is not None:
            images.append(img)
            labels.append(label)
    return images, labels
```

Implementasi:

```
# Data train
folder_train_positif = os.path.join(dataset_path, 'Positif')
folder_train_negatif = os.path.join(dataset_path, 'Negatif')

# Data test
folder_test_positif = os.path.join(dataset_path, 'Uji/Data
Uji/positif')
folder_test_negatif = os.path.join(dataset_path, 'Uji/Data
Uji/negatif')
```

```
# Data train
positif_train_images, positif_train_labels =
load_images_from_folder(folder_train_positif, 1)
negatif_train_images, negatif_train_labels =
load_images_from_folder(folder_train_negatif, 0)

train_images = positif_train_images + negatif_train_images
train_labels = positif_train_labels + negatif_train_labels

# Data uji
```

```

positif_test_images, positif_test_labels =
load_images_from_folder(folder_test_positif, 1)
negatif_test_images, negatif_test_labels =
load_images_from_folder(folder_test_negatif, 0)

test_images = positif_test_images + negatif_test_images
test_labels = positif_test_labels + negatif_test_labels

```

### C. Ekstraksi Fitur

Ekstraksi fitur menggunakan HOG untuk setiap data latih dan data uji. Tahapan ekstraksi diawali dengan mengubah ukuran citra menjadi 128x128 untuk memberikan sebuah standar ukuran dikarenakan ukuran dari sumber dataset tidaklah sama. Lalu dilakukan ekstraksi fitur menggunakan hog dengan parameter:

- 9 orientation bins
- Sel berukuran 8x8 piksel
- Ukuran blok 2x2 sel
- Normalisasi blok menggunakan L2-Hys

Sehingga total fitur HOG yang dihasilkan dapat dihitung sebagai berikut:

$$\text{cells in one direction} = \frac{\text{image size}}{\text{cell size}} = \frac{128}{8} = 16$$

$$\text{blocks in one direction} = \text{cell in one direction} - 1 = 15$$

$$\begin{aligned} \text{total block} &= \text{blocks in one direction} * \text{blocks in one direction} \\ &= 15 * 15 = 225 \end{aligned}$$

$$\begin{aligned} \text{total features per block} &= \text{cells per block} * \text{orientation} \\ &= 2 * 2 * 9 = 36 \end{aligned}$$

$$\begin{aligned} \text{total HOG features} &= \text{total blocks} * \text{total features per block} \\ &= 225 * 36 = 8100 \end{aligned}$$

Fungsi ekstraksi fitur menggunakan HOG:

```

def extract_hog_features(images, target_size=(128, 128)):
    hog_features = []
    for image in images:
        # Resize gambar
        resized_image = resize(image, target_size)

```

```

        # Ekstraksi fitur HOG dari gambar yang sudah
        di-resize
        features = hog(resized_image, orientations=9,
pixels_per_cell=(8, 8),
                                cells_per_block=(2, 2),
block_norm='L2-Hys')

        # Menambahkan vektor fitur HOG ke dalam list
        hog_features.append(features)

    return hog_features

```

Implementasi:

```

# Features data train
train_features = extract_hog_features(train_images)

# Features data test
test_features = extract_hog_features(test_images)

```

#### D. Klasifikasi

Setelah data fitur hasil ekstraksi menggunakan HOG didapatkan, dilakukan klasifikasi menggunakan algoritma KNN. Algoritma KNN sendiri memiliki nilai akurasi yang fluktuatif tergantung pada nilai k yang digunakan saat pelatihan data. Oleh karena itu, pada penugasan ini penulis menggunakan K-Fold Cross Validation untuk mendapatkan nilai k dengan akurasi tertinggi. Nilai k yang didapatkan dari proses Cross Validation kemudian digunakan ke model utama untuk pelatihan data-data fitur hasil ekstraksi.

Algoritma K-Fold Cross Validation dengan 10 fold:

```

# Rentang nilai N yang ingin diuji
k_values = range(1, 21)

# Menyimpan akurasi untuk setiap nilai K
accuracies = []

```

```

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    # Menggunakan cross-validation untuk mendapatkan akurasi
    scores = cross_val_score(knn, train_features,
train_labels, cv=10) # cv=10 untuk 10-fold
    accuracies.append(scores.mean()) # Rata-rata akurasi

# Menemukan nilai N terbaik
best_k = k_values[np.argmax(accuracies)]
best_accuracy = max(accuracies)

```

Model KNN dan pelatihan dengan k yang memiliki akurasi terbaik:

```

knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(train_features, train_labels)

```

## E. Prediksi dan Confusion Matrix

Setelah melakukan pelatihan model, model diuji keakuratannya dengan melakukan prediksi terhadap tiap data uji yang kemudian kumpulan hasil prediksi tadi akan ditampilkan dalam bentuk Confusion Matrix agar bisa dilakukan perhitungan untuk precision, recall, dan f1-score-nya.

Prediksi model:

```

# Memprediksi setiap gambar dalam data uji menggunakan loop
predictions = []

for features in test_features:
    # Prediksi label untuk fitur saat ini
    prediction = knn.predict([features]) # Input harus
berbentuk array 2D
    predictions.append(prediction[0]) # Menambahkan prediksi
ke dalam daftar

# Menghitung akurasi dari prediksi
accuracy_uji = accuracy_score(test_labels, predictions)
print(f'Akurasi pada data uji: {accuracy_uji * 100:.2f}%')

```

Confusion Matrix:



```
cm = confusion_matrix(test_labels, predictions)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues) # Menggunakan colormap biru
                                untuk visualisasi
plt.title('Confusion Matrix')
plt.show()
```

## IV. Hasil dan Pembahasan

### A. Hasil

#### 1. Data Preprocessing

Data positif	Data negatif
<p>Contoh Gambar Positif</p> 	<p>Contoh Gambar negatif</p> 

Berikut adalah data hasil preprocessing sebelum dilakukan ekstraksi menggunakan HOG.

#### 2. Data hasil HOG

- Data latih:

	0	1	2	3	4	5	6	7	8	9 ...	8090	8091	8092	8093	8094	8095	8096	8097	8098	8099
count	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000
mean	0.183108	0.083654	0.076326	0.083313	0.187110	0.093820	0.078775	0.077509	0.110988	0.223385	0.146607	0.186619	0.080475	0.072567	0.086722	0.185201	0.089406	0.075941	0.082497	0.126456
std	0.120757	0.091765	0.088029	0.092132	0.123134	0.100578	0.090140	0.088958	0.110072	0.122597	0.124597	0.115913	0.086875	0.084848	0.095588	0.121126	0.095710	0.086360	0.087426	0.115762
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.075819	0.011633	0.006920	0.011208	0.081704	0.013626	0.012318	0.010856	0.021251	0.119760	0.037841	0.082956	0.011176	0.007235	0.011514	0.080902	0.012731	0.011764	0.010405	0.026321
50%	0.172888	0.049606	0.044683	0.049065	0.180298	0.068905	0.041972	0.042756	0.071091	0.244397	0.116671	0.188445	0.050248	0.042341	0.049357	0.177923	0.061193	0.040727	0.054088	0.090539
75%	0.273192	0.126571	0.115597	0.123601	0.278298	0.139421	0.110444	0.112367	0.170415	0.302974	0.253131	0.276282	0.124829	0.100048	0.131028	0.278217	0.133172	0.112530	0.126361	0.210872
max	0.589396	0.421651	0.417828	0.421361	0.551747	0.487920	0.536361	0.412036	0.490225	0.623483	0.551602	0.500164	0.361874	0.417955	0.462583	0.611374	0.502818	0.379616	0.417653	0.458569

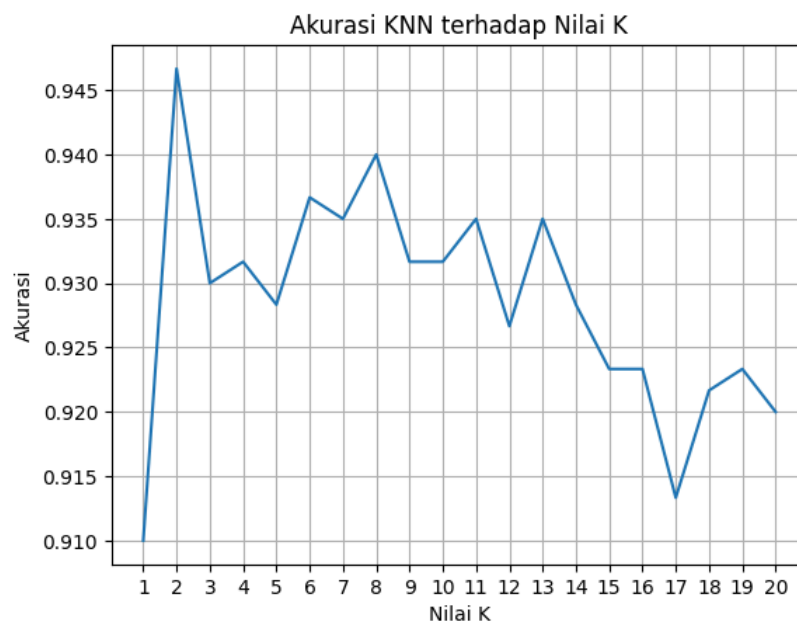
- Data uji:

	0	1	2	3	4	5	6	7	8	9	...	8090	8091	8092	8093	8094	8095	8096	8097	8098	8099
count	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	...	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000
mean	0.206757	0.079528	0.095163	0.068089	0.157892	0.055384	0.061494	0.081200	0.135468	0.237521	...	0.172621	0.202438	0.079175	0.068028	0.092979	0.162944	0.095797	0.090410	0.107697	0.165780
std	0.123997	0.087541	0.069766	0.067544	0.125424	0.057715	0.065446	0.082099	0.127312	0.123843	...	0.140725	0.127299	0.085517	0.075044	0.103669	0.113842	0.105831	0.099537	0.118065	0.143232
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.002460	0.009958	0.000000	0.000000	0.000000	0.007238	0.000000	0.000000	0.000000	0.000000
25%	0.116323	0.016587	0.007404	0.013060	0.041861	0.015432	0.004860	0.012841	0.034675	0.139357	...	0.044673	0.087426	0.018184	0.012695	0.019922	0.056974	0.008254	0.015370	0.020556	0.055198
50%	0.227414	0.059031	0.037827	0.048808	0.149508	0.034539	0.047102	0.044587	0.075188	0.261896	...	0.159142	0.219448	0.051317	0.041551	0.045262	0.152889	0.047082	0.053497	0.047482	0.112495
75%	0.302894	0.095223	0.117071	0.101601	0.249062	0.073142	0.095197	0.152236	0.235891	0.310331	...	0.268793	0.265352	0.087538	0.097801	0.154683	0.272610	0.159110	0.122845	0.185397	0.272610
max	0.430993	0.318726	0.265655	0.261665	0.468103	0.234305	0.298943	0.290982	0.430993	0.543660	...	0.490059	0.460881	0.269285	0.269285	0.308474	0.420997	0.342439	0.342439	0.356598	0.507053

Dapat dilihat bahwa fitur yang didapatkan dari ekstraksi menggunakan HOG berjumlah 8100 kolom, sesuai dengan perhitungan awal.

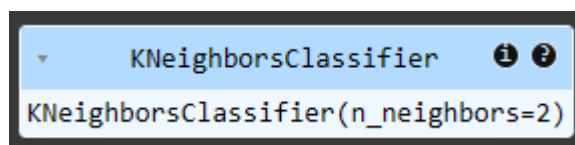
### 3. Klasifikasi KNN dengan K terbaik

- Hasil K-Fold Cross Validation:



Dari grafik yang ditampilkan, dapat dilihat bahwa akurasi tertinggi didapatkan dengan menggunakan nilai  $K = 2$

- Pelatihan model KNN dengan  $K = 2$



### 4. Prediksi

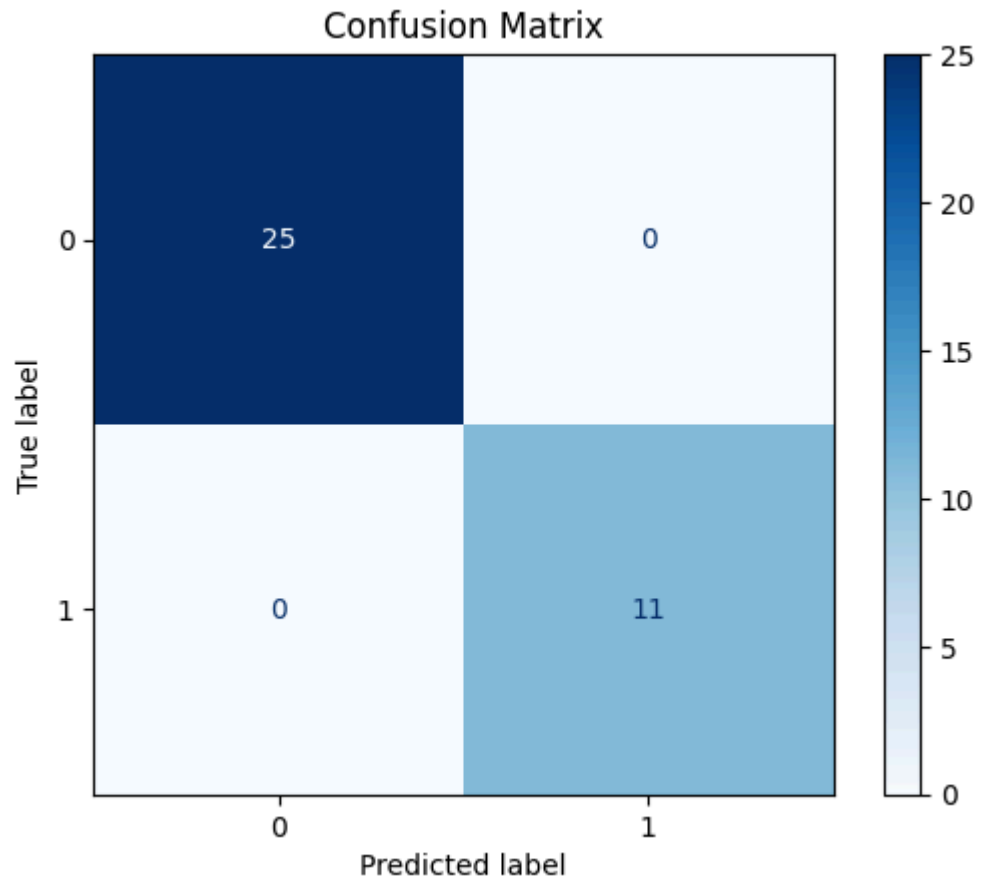
**Akurasi pada data uji: 100.00%**

Akurasi yang didapatkan dari prediksi sebesar 100%, yang berarti model berhasil memprediksikan semua data uji dengan benar sesuai labelnya.

### 5. Evaluasi model menggunakan Confusion Matrix

- Confusion Matrix:





Dari confusion matrix tersebut terlihat bahwa model selalu memprediksi dengan tepat untuk setiap data uji.

- Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25
1	1.00	1.00	1.00	11
accuracy			1.00	36
macro avg	1.00	1.00	1.00	36
weighted avg	1.00	1.00	1.00	36

Dari confusion matrix, dapat diketahui untuk precision, recall, dan f1-score untuk model klasifikasi KNN menggunakan  $K = 2$ .

## V. Kesimpulan

Dari berbagai rangkaian proses yang dikerjakan dalam penugasan ini, penulis menyimpulkan bahwa klasifikasi dataset citra rambu-rambu lalu lintas dapat dilakukan menggunakan algoritma K-Nearest Neighbors (KNN) dengan memanfaatkan Histogram of Oriented Gradients (HOG) untuk mengekstraksi fitur

dari setiap data citra. Dalam penelitian ini, penulis menggunakan algoritma K-Fold Cross Validation untuk menentukan nilai K yang menghasilkan akurasi tertinggi. Ditemukan bahwa nilai K dengan akurasi tertinggi adalah 2, sehingga pelatihan model dilakukan dengan K terbaik tersebut. Model yang telah dilatih memperoleh akurasi 100%, yang divalidasi menggunakan confusion matrix, di mana nilai False Negative dan False Positive masing-masing adalah 0. Dengan demikian, model ini memiliki precision, recall, dan F1-score sebesar 1, yang menunjukkan bahwa model ini berhasil melakukan prediksi dengan tepat untuk setiap data uji.