

# **Laporan Proyek Machine Learning**

## **Human Detection in Images**

Kelompok 14

C14190165 - Sastra Himawan

C14190166 - Kwan, Davin Kanelson

C14190168 - Michael Wong



Semester Ganjil

Fakultas Teknologi Industri

Universitas Kristen Petra

Surabaya

2021

## I. Anggota Kelompok dan Pembagian Kerja

- Michael Wong : Melakukan eksperimen terhadap proses training data, mengklasifikasi dataset menjadi tahap-tahap, melakukan training terhadap dataset yang telah diklasifikasikan.
- Sastra Himawan : Melakukan eksperimen terhadap proses training data, memisahkan dataset menjadi tahapan yang telah ditentukan, Melakukan training terhadap dataset yang telah ditentukan.
- Kwan, Davin Kanelson : Melakukan eksperimen terhadap proses training data, melakukan pemisahan atau *filter* pada dataset sesuai dengan tahapan yang ditentukan, melakukan training pada dataset sesuai tahapan yang telah ditentukan.

## II. Pembahasan

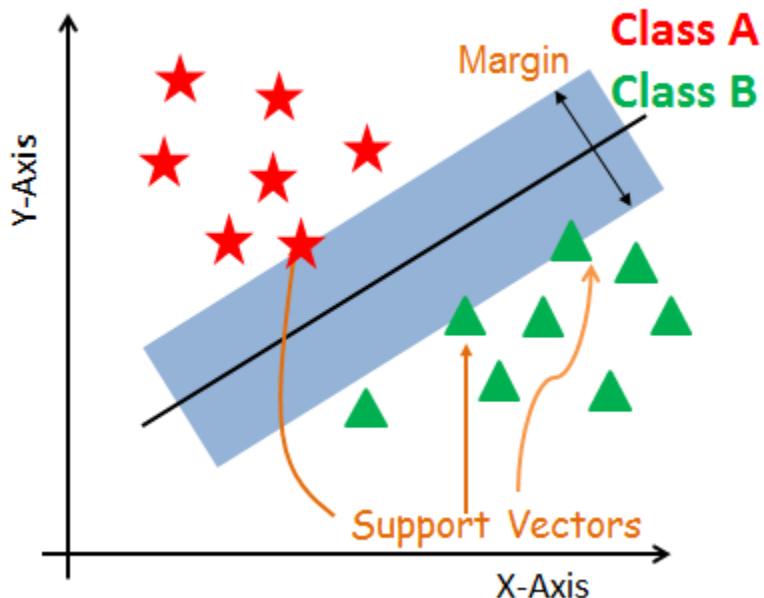
Untuk melakukan pre-processing gambar, kelompok kami menggunakan Histogram of Oriented Gradients (HOG). HOG adalah salah satu *Feature Descriptors* yang menerjemahkan gambar ke dalam bentuk representasi yang sederhana (hanya berisi informasi paling penting tentang gambar). Hal ini menjadi sangat berguna untuk membedakan gambar yang satu dengan yang lainnya. Dengan kata lain, HOG memiliki fungsi untuk mendeskripsikan karakteristik unik dari suatu gambar dengan informasi-informasi tertentu.



Pada penerapan HOG dalam *Machine Learning* kelompok kami, hasil dari gambar yang dideskripsikan oleh HOG berupa *2-dimensional array* yang mewakili row dan height. Row dan height dari array tersebut mewakili setiap cell dari gambar yang telah diproses sehingga menghasilkan *output* sebagai berikut.

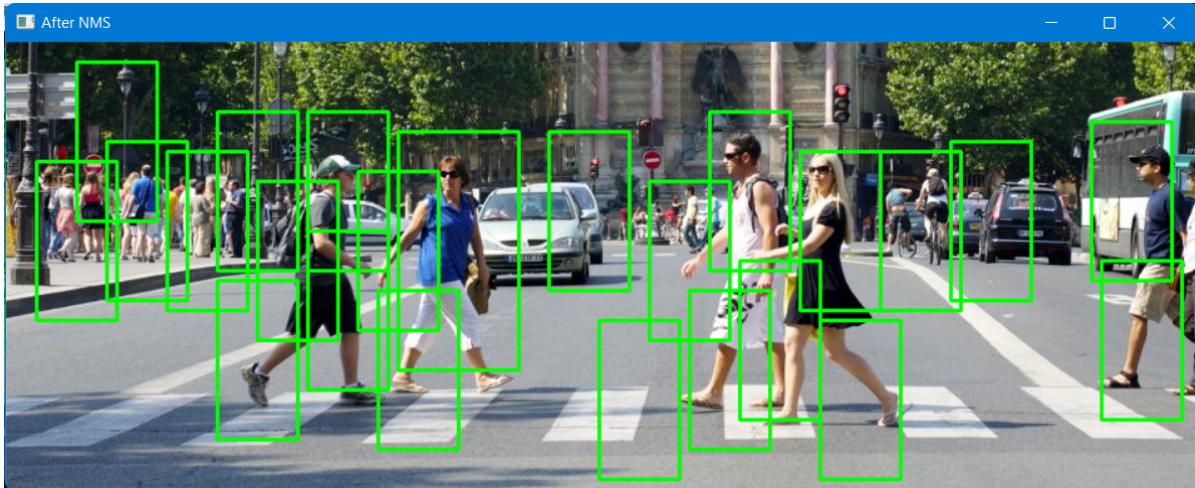
```
[0. 0. 0. ... 0. 0. 0.]
[0.20050187 0.          0.          ... 0.01521921 0.01299267 0.          ]
[0.          0.          0.          ... 0.02464635 0.          0.          ]
[0.          0.          0.          ... 0.01700061 0.00855262 0.          ]
[0.          0.          0.          ... 0.02057383 0.          0.          ]
[0.          0.          0.          ... 0.01243259 0.0582364 0.          ]
[0.16510891 0.          0.          ... 0.05815734 0.01303652 0.          ]
[0.          0.          0.          ... 0.02306412 0.03894239 0.          ]
[0.          0.          0.          ... 0.02379125 0.05276261 0.          ]
[0.          0.          0.          ... 0.27467766 0.02675054 0.          ]
[0.11789658 0.          0.02381871 ... 0.05021217 0.          0.          ]
[0.          0.          0.          ... 0.03218227 0.          0.          ]
[0.          0.          0.          ... 0.01628357 0.          0.          ]
[0.09105167 0.          0.          ... 0.21019272 0.          0.          ]
[0.          0.          0.          ... 0.23561249 0.          0.          ]
[0.          0.          0.          ... 0.08099584 0.03603683 0.          ]
[0.06162584 0.          0.02178803 ... 0.00634349 0.01002994 0.01418447]
[0. 0. 0. ... 0. 0. 0.]
```

Dari *output* yang dideskripsikan oleh HOG, data *array* tersebut ditambahkan kedalam numpy *array* dan diberi label untuk melakukan *training* dengan metode *supervised learning*. Gambar yang terdapat manusia diberi label 1 sedangkan gambar yang tidak terdapat manusia diberi label 0. Untuk metode pengklasifikasian data, kami menggunakan metode Support-vector Machine (SVM) yang merupakan salah satu metode pembelajaran *supervised learning* yang melakukan analisis data untuk klasifikasi dan analisis regresi. Pada proses pengklasifikasian data, SVM mencari *hyperplane* terbaik dengan memaksimalkan jarak antar kelas. *Hyperplane* adalah sebuah fungsi yang dapat digunakan untuk pemisah antar kelas sehingga akan terbentuk 2 kategori atau kelompok jika diberikan suatu dataset.



Untuk implementasinya, digunakan Linear SVC yang termasuk salah satu metode dalam SVM yang bekerja dengan cara menyesuaikan *hyperplane* dengan data yang diberikan agar menghasilkan pemisahan data yang paling tepat dan sesuai. Dari *hyperplane* yang didapatkan

tersebut, kita dapat menggunakan fitur *predict* untuk mengkategorikan data-data baru yang diinputkan oleh pengguna dalam bentuk gambar.



### III. Hasil Pengujian

Untuk menguji tingkat efektivitas dan akurasi dari penerapan metode *machine learning* SVM (Support-machine Vector) Linear SVC (Support Vector Classifier), dilakukan pengujian dengan 4 tahap dataset yang telah di-*filter* pada setiap tahapnya. Tahap pertama akan dilakukan *filter* untuk memisahkan manusia asli dengan objek yang menyerupai manusia (patung, orang-orangan sawah). Pada tahap kedua akan dilakukan *filter* untuk memisahkan kerumunan manusia yang terlalu ramai dengan dataset awal. Tahap ketiga akan dilakukan *filter* untuk memisahkan kerumunan manusia yang tubuhnya tidak terlihat utuh di gambar dengan dataset awal. Serta tahap terakhir akan memisahkan kerumunan manusia dengan dataset awal sehingga yang dianggap manusia hanyalah gambar manusia utuh (dari kepala hingga kaki) yang terlihat pada gambar.

**Tahap 1 (pos=1892, neg=2782)**



**Diberi label negatif**



**Diberi label positif**

**Tahap 2 (pos=1833, neg=2782)**



Diberi label negatif



Diberi label positif

**Tahap 3 (pos=99, neg=2782)**



Diberi label negatif



Diberi label positif

#### Tahap 4 (pos=1128, neg=2782)



**Diberi label negatif**



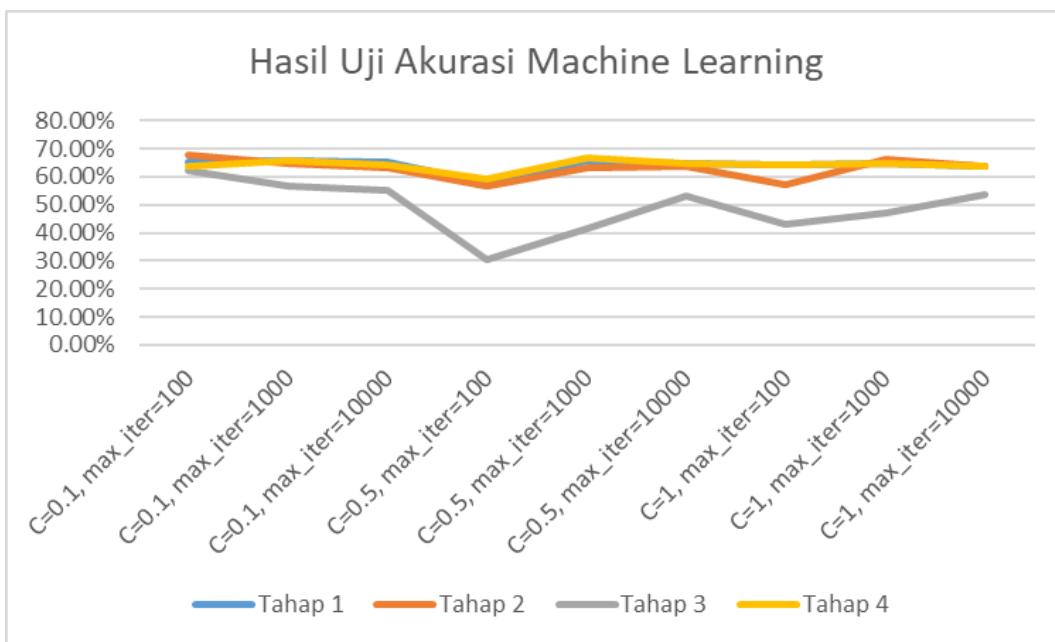
**Diberi label positif**

Selain melakukan manipulasi terhadap dataset yang tersedia, kami juga melakukan manipulasi terhadap 2 parameter dari fungsi `svm.LinearSVC()`. Parameter tersebut adalah variabel *C* dan *max\_iter*. Variabel *C* merupakan parameter untuk menentukan penilaian seberapa ketat kita ingin mengklasifikasikan atau menyesuaikan data atau disebut dengan regularisasi. Sedangkan parameter *max\_iter* mempengaruhi berapa banyak maksimal iterasi yang ingin dilakukan ketika menjalankan proses. Dari kedua parameter tersebut, kami melakukan manipulasi untuk variabel *C* sebanyak 3 kali percobaan dengan nilai 0,1, 0,5, dan 1. Sedangkan untuk parameter *max\_iter* kami melakukan percobaan sebanyak 3 kali dengan nilai 100, 1000, dan 10000 sehingga dari percobaan parameter tersebut, diperlukan 9 kali percobaan dengan kombinasi parameter yang berbeda-beda untuk setiap tahap dataset yang telah di-filter. Dengan kata lain, akan ada 36 percobaan training dataset untuk mengetahui tingkat akurasi dari penerapan metode SVM SVC. Dari hasil pengujian terhadap dataset dan parameter tersebut, didapatkan hasil sebagai berikut.

Percobaan Parameter	Tahap 1	Tahap 2	Tahap 3	Tahap 4
C=0.1, max_iter=100	65,14%	67,79%	62,21%	63,81%
C=0.1, max_iter=1000	65,89%	64,70%	56,86%	65,51%
C=0.1, max_iter=10000	65,46%	63,14%	55,36%	64,23%
C=0.5,	57,76%	56,73%	30,51%	59,25%

max_iter=100				
C=0.5, max_iter=1000	65,54%	63,24%	41,30%	66,74%
C=0.5, max_iter=10000	64,75%	63,91%	53,34%	64,60%
C=1, max_iter=100	64,28%	56,95%	42,86%	64,17%
C=1, max_iter=1000	64,68%	66,17%	47,05%	64,64%
C=1, max_iter=10000	63,76%	63,95%	53,72%	63,83%

Grafik Hasil Uji Akurasi



Hasil uji coba:

1. Tingkat akurasi antara tahapan yang satu dengan yang lain tidak jauh berbeda (tidak terlalu signifikan).
2. Mengubah nilai atau *value* terhadap variabel C dan max\_iter akan mempengaruhi tingkat akurasi terhadap setiap tahapan yang diuji coba.
3. Semakin banyak data yang dilatih, maka akan semakin akurat juga nilai akurasi yang didapatkan.

4. Jika terlalu jauh jumlah data nya yang dilatih (jumlah data dari neg dan pos) maka hal tersebut dapat berpengaruh pada nilai akurasi, misalnya total data pos hanya sekitar 100 data, akan tetapi total data neg sekitar 10.000 data.

5. Ukuran semua gambar diusahakan harus sama besar, misal 128 pixel x 128 pixel karena jika ukurannya tidak sama, maka saat gambar dicrop menjadi satu ukuran yang sama, bisa saja gambar yang terdapat manusia juga ikut dicrop, sehingga gambar manusia menjadi tidak utuh.

#### IV. Kesimpulan

Kesimpulan yang bisa didapatkan mengenai percobaan dari kelompok kami adalah melakukan pemisahan atau mengelompokkan dataset menjadi beberapa tahapan atau ketentuan yang berbeda-beda memiliki dampak atau pengaruh dalam hal waktu untuk melakukan proses *training data* dan seberapa besar akurasi data dengan menggunakan *data test* yang tersedia. Faktor-faktor yang mendukung adanya perbedaan antara setiap *training* dataset telah kita analisis kedalam 4 poin. Poin pertama yang mempengaruhi akurasi adalah parameter dari fungsi LinearSVC() yaitu variable C dan iter dimana parameter ini berkaitan dengan *stopping criteria* dan *penalty*. Poin yang kedua yaitu berpengaruh nya jumlah / banyak data yang dilatih (ditrain) kepada nilai akurasi, maksudnya adalah semakin banyak data yang dilatih, maka akan semakin akurat juga nilai akurasi yang didapatkan. Poin ketiga adalah keseimbangan jumlah data antara gambar yang diberi label positif dan gambar yang diberi label negatif. Hal ini sangat mempengaruhi karena dengan jumlah data yang seimbang proses pelatihan mesin akan lebih akurat. Poin keempat adalah ukuran dari tiap gambar akan mempengaruhi nilai akurasi yang akan didapatkan. Sangat direkomendasikan ukuran dari tiap gambar bernilai sama misalnya 128 pixel x 128 pixel karena bila gambar yang ada memiliki ukuran yang berbeda, pada saat penjalanan fungsi crop\_centre bisa saja gambar yang dicrop akan memiliki hasil pemotongan yang berbeda, misal nya gambar yang memiliki size lebih besar dari 128 x 128 pixel bisa saja pada proses crop\_centre dijalankan maka hasil pemotongan gambar akan tidak sesuai.

#### V. Lampiran

##### Kode untuk melakukan training data (train.py)

```
import cv2
from sklearn import svm
import os
import numpy as np
import joblib
from skimage.feature import hog
from sklearn.utils import shuffle
import sys
import argparse
import random

MAX_HARD_NEGATIVES = 40000
```

```
parser = argparse.ArgumentParser(description='Parse Training Directory')
parser.add_argument('--pos', help='Path to directory containing Positive Images')
parser.add_argument('--neg', help='Path to directory containing Negative images')

args = parser.parse_args()
pos_img_dir = args.pos
neg_img_dir = args.neg

def crop_centre(img):
    h, w, _ = img.shape
    l = (w - 64)/2
    t = (h - 128)/2
    crop = img[int(t):int(t+128), int(l):int(l+64)]
    return crop

def ten_random_windows(img):
    h, w = img.shape
    if h < 128 or w < 64:
        return []
    h = h - 128;
    w = w - 64
    windows = []

    for i in range(0, 10):
        x = random.randint(0, w)
        y = random.randint(0, h)
        windows.append(img[y:y+128, x:x+64])

    return windows

def read_filenames():
    f_pos = []
    f_neg = []
    mypath_pos = pos_img_dir
    for (dirpath, dirnames, filenames) in os.walk(mypath_pos):
        f_pos.extend(filenames)
        break
    mypath_neg = neg_img_dir
    for (dirpath, dirnames, filenames) in os.walk(mypath_neg):
        f_neg.extend(filenames)
        break
```

```

    return f_pos, f_neg

def read_images(pos_files, neg_files):
    X = []
    Y = []
    pos_count = 0
    for img_file in pos_files:
        print(os.path.join(pos_img_dir, img_file))
        img = cv2.imread(os.path.join(pos_img_dir, img_file))
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        windows = ten_random_windows(gray)
        for win in windows:
            features = hog(win, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(
                2, 2), block_norm="L2", transform_sqrt=True,
feature_vector=True)
            pos_count += 1
            X.append(features)
            Y.append(1)

    neg_count = 0
    for img_file in neg_files:
        print(os.path.join(neg_img_dir, img_file))
        img = cv2.imread(os.path.join(neg_img_dir, img_file))
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        windows = ten_random_windows(gray_img)
        for win in windows:
            features = hog(win, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), block_norm="L2", transform_sqrt=True,
feature_vector=True)
            neg_count += 1
            X.append(features)
            Y.append(0)
    return X, Y, pos_count, neg_count

def sliding_window(image, window_size, step_size):
    ...
    This function returns a patch of the input image `image` of size equal
    to `window_size`. The first image returned top-left co-ordinates (0, 0)
    and are increment in both x and y directions by the `step_size`
    supplied.
    So, the input parameters are -
    * `image` - Input Image

```

```

* `window_size` - Size of Sliding Window
* `step_size` - Incremented Size of Window

The function returns a tuple -
(x, y, im_window)
where
* x is the top-left x co-ordinate
* y is the top-left y co-ordinate
* im_window is the sliding window image
...
for y in range(0, image.shape[0]-128, step_size[1]):
    for x in range(0, image.shape[1]-64, step_size[0]):
        yield (x, y, image[y:y + window_size[1], x:x + window_size[0]])

def hard_negative_mine(f_neg, winSize, winStride):
    hard_negatives = []
    hard_negative_labels = []
    count = 0
    num = 0
    for imgfile in f_neg:
        img = cv2.imread(os.path.join(neg_img_dir, imgfile))
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        for (x, y, im_window) in sliding_window(gray, winSize, winStride):
            features = hog(im_window, orientations=9, pixels_per_cell=(8,
8), cells_per_block=(2, 2), block_norm="L2", transform_sqrt=True,
feature_vector=True)
            if (clf1.predict([features]) == 1):
                hard_negatives.append(features)
                hard_negative_labels.append(0)
                count = count + 1
            if (count == MAX_HARD_NEGATIVES):
                return np.array(hard_negatives),
np.array(hard_negative_labels)

        num = num + 1
        sys.stdout.write("\r" + "\tHard Negatives Mined: " + str(count) +
"\tCompleted: " + str(round((count / float(MAX_HARD_NEGATIVES))*100, 4)) +
" %")

        sys.stdout.flush()

    return np.array(hard_negatives), np.array(hard_negative_labels)

```

```

pos_img_files, neg_img_files = read_filenames()

print("Total Positive Images : " + str(len(pos_img_files)))
print("Total Negative Images : " + str(len(neg_img_files)))
print("Reading Images")

X, Y, pos_count, neg_count = read_images(pos_img_files, neg_img_files)

X = np.array(X)
Y = np.array(Y)

X, Y = shuffle(X, Y, random_state=0)

print("Images Read and Shuffled")
print("Positives: " + str(pos_count))
print("Negatives: " + str(neg_count))
print("Training Started")

clf1 = svm.LinearSVC(C=0.5, max_iter=100, class_weight='balanced', verbose
= 1)

clf1.fit(X, Y)
print("Trained")

joblib.dump(clf1, 'person_pre-eliminary.pkl')

print("Hard Negative Mining")

winStride = (8, 8)
winSize = (64, 128)

print ("Maximum Hard Negatives to Mine: " + str(MAX_HARD_NEGATIVES))

hard_negatives, hard_negative_labels = hard_negative_mine(neg_img_files,
winSize, winStride)

sys.stdout.write("\n")

hard_negatives = np.concatenate((hard_negatives, X), axis = 0)
hard_negative_labels = np.concatenate((hard_negative_labels, Y), axis = 0)

hard_negatives, hard_negative_labels = shuffle(hard_negatives,

```

```

hard_negative_labels, random_state=0)

print("Final Samples Dims: " + str(hard_negatives.shape))
print("Retraining the classifier with final data")

clf2 = svm.LinearSVC(C=0.5, max_iter=100, class_weight='balanced', verbose
= 1)

clf2.fit(hard_negatives, hard_negative_labels)

print("Trained and Dumping")

joblib.dump(clf2, 'person_final.pkl')

```

**Kode untuk melakukan evaluasi dari hasil training (test.py)**

```

import cv2
from sklearn import svm
import os
import numpy as np
import joblib
from skimage.feature import hog
import argparse

parser = argparse.ArgumentParser(description='Parse Training Directory')
parser.add_argument('--pos', help='Path to directory containing Positive
Images')
parser.add_argument('--neg', help='Path to directory containing Negative
images')

args = parser.parse_args()

pos_img_dir = args.pos
neg_img_dir = args.neg

clf = joblib.load('person_final.pkl')

total_pos_samples = 0
total_neg_samples = 0

def crop_centre(img):
    h, w, d = img.shape

```

```

l = (w - 64)/2
t = (h - 128)/2
#print (h, w, l, t)
crop = img[int(t):int(t+128), int(l):int(l+64)]
return crop

def read_filenames():
    f_pos = []
    f_neg = []
    for (dirpath, dirnames, filenames) in os.walk(pos_img_dir):
        f_pos.extend(filenames)
        break

    for (dirpath, dirnames, filenames) in os.walk(neg_img_dir):
        f_neg.extend(filenames)
        break

    print("Positive Image Samples: " + str(len(f_pos)))
    print("Negative Image Samples: " + str(len(f_neg)))

    return f_pos, f_neg

def read_images(f_pos, f_neg):
    print ("Reading Images")
    array_pos_features = []
    array_neg_features = []
    global total_pos_samples
    global total_neg_samples
    # print("Positive Samples: " + str(len(f_pos)))
    for imgfile in f_pos:
        img = cv2.imread(os.path.join(pos_img_dir, imgfile))
        cropped = crop_centre(img)
        gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
        features = hog(gray, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), block_norm="L2", feature_vector=True)
        array_pos_features.append(features.tolist())
        total_pos_samples += 1

    # print("Negative Samples: " + str(len(f_neg)))
    for imgfile in f_neg:
        img = cv2.imread(os.path.join(neg_img_dir, imgfile))
        cropped = crop_centre(img)
        gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)

```

```

        features = hog(gray, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), block_norm="L2", feature_vector=True)
        array_neg_features.append(features.tolist())
        total_neg_samples += 1

    return array_pos_features, array_neg_features

pos_img_files, neg_img_files = read_filenames()

pos_features, neg_features = read_images(pos_img_files, neg_img_files)

pos_result = clf.predict(pos_features)
neg_result = clf.predict(neg_features)

# print("Positive Predictions: " + str(pos_result))
true_positives = cv2.countNonZero(pos_result)
false_negatives = len(pos_result) - true_positives

# print("Negative Predictions: " + str(neg_result))
false_positives = cv2.countNonZero(neg_result)
true_negatives = len(neg_result) - false_positives

print("True Positives: " + str(true_positives), "False Positives: " +
str(false_positives))
print("True Negatives: " + str(true_negatives), "False Negatives: " +
str(false_negatives))

precision_pos = float(true_positives) / (true_positives + false_positives)
precision_neg = float(true_negatives) / (true_negatives + false_negatives)
precision = (precision_pos + precision_neg) / 2

print("Precision: " + str(precision))

```

Kode untuk melakukan predict dengan output sebuah gambar (detectmulti.py)

```

import cv2
from sklearn import svm
import os
import numpy as np
import joblib
from skimage.feature import hog
import argparse

```

```
parser = argparse.ArgumentParser(description='Parse Training Directory')
parser.add_argument('--pos', help='Path to directory containing Positive Images')
parser.add_argument('--neg', help='Path to directory containing Negative images')

args = parser.parse_args()

pos_img_dir = args.pos
neg_img_dir = args.neg

clf = joblib.load('person_final.pkl')

total_pos_samples = 0
total_neg_samples = 0

def crop_centre(img):
    h, w, d = img.shape
    l = (w - 64)/2
    t = (h - 128)/2
    #print (h, w, l, t)
    crop = img[int(t):int(t+128), int(l):int(l+64)]
    return crop

def read_filenames():
    f_pos = []
    f_neg = []
    for (dirpath, dirnames, filenames) in os.walk(pos_img_dir):
        f_pos.extend(filenames)
        break

    for (dirpath, dirnames, filenames) in os.walk(neg_img_dir):
        f_neg.extend(filenames)
        break

    print("Positive Image Samples: " + str(len(f_pos)))
    print("Negative Image Samples: " + str(len(f_neg)))

    return f_pos, f_neg

def read_images(f_pos, f_neg):
    print ("Reading Images")
    array_pos_features = []
```

```

array_neg_features = []
global total_pos_samples
global total_neg_samples
# print("Positive Samples: " + str(len(f_pos)))
for imgfile in f_pos:
    img = cv2.imread(os.path.join(pos_img_dir, imgfile))
    cropped = crop_centre(img)
    gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
    features = hog(gray, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), block_norm="L2", feature_vector=True)
    array_pos_features.append(features.tolist())
    total_pos_samples += 1

# print("Negative Samples: " + str(len(f_neg)))
for imgfile in f_neg:
    img = cv2.imread(os.path.join(neg_img_dir, imgfile))
    cropped = crop_centre(img)
    gray = cv2.cvtColor(cropped, cv2.COLOR_BGR2GRAY)
    features = hog(gray, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), block_norm="L2", feature_vector=True)
    array_neg_features.append(features.tolist())
    total_neg_samples += 1

return array_pos_features, array_neg_features

pos_img_files, neg_img_files = read_filenames()

pos_features, neg_features = read_images(pos_img_files, neg_img_files)

pos_result = clf.predict(pos_features)
neg_result = clf.predict(neg_features)

# print("Positive Predictions: " + str(pos_result))
true_positives = cv2.countNonZero(pos_result)
false_negatives = len(pos_result) - true_positives

# print("Negative Predictions: " + str(neg_result))
false_positives = cv2.countNonZero(neg_result)
true_negatives = len(neg_result) - false_positives

print("True Positives: " + str(true_positives), "False Positives: " +
str(false_positives))
print("True Negatives: " + str(true_negatives), "False Negatives: " +

```

```
str(false_negatives))

precision_pos = float(true_positives) / (true_positives + false_positives)
precision_neg = float(true_negatives) / (true_negatives + false_negatives)
precision = (precision_pos + precision_neg) / 2

print("Precision: " + str(precision))
```

**Sumber atau referensi:**

[https://github.com/vinay0410/Pedestrian\\_Detection](https://github.com/vinay0410/Pedestrian_Detection)

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html>

[https://www.reddit.com/r/computervision/comments/2ggc5l/what\\_is\\_hard\\_negative\\_mining\\_and\\_how\\_is\\_it/](https://www.reddit.com/r/computervision/comments/2ggc5l/what_is_hard_negative_mining_and_how_is_it/)