Implementasi Support Vector Machine dalam Mendeteksi Kepadatan Jalan

Andre Yosua[#1], Ken Ratri Retno Wardani, S.Kom, M.T.[*2]

[#]*Departemen Informatika, Institut Teknologi Harapan Bangsa*
Jl. Dipati Ukur No.80, Bandung, Indonesia

[1]andreyosua46@gmail.com

[3]ken_ratri@ithb.ac.id

Abstract*— Traffic lights have an important role in the flow of traffic. Most of the traffic lights in Indonesia are controlled automatically with a set time, but this method is not effective. This can be avoided if the time that determines how long the traffic lights are on depends on the density of the road. With the development of technology, a computer can detect an object in an image or video. The field in computer technology that can process images or video is Computer Vision. Computer Vision can give the ability to a computer like the human eye. By dividing the frame video into grid which contains 11(horizontal) x 10(vertical) cell and each cell has a size of 44x44 pixels. Each cell will be calculated using Histogram of Oriented Gradients(HOG) and Local Binary Pattern(LBP) to find its feature and after that it will be classified by Support Vector Machine(SVM) into a positive class (Traffic) and negative class (Not-Traffic). With the dataset used there are several cells that are in the wrong class so that the highest accuracy is 83%.

Keywords*— Traffic density, Image Processing, Computer Vision, Histogram of Oriented Gradient, Local Binary Pattern, Support Vector Machine.

Abstrak*— Lampu lalu lintas memiliki peranan yang penting dalam kelancaran lalu lintas. Sebagian besar lampu lalu lintas di Indonesia di kendalikan secara otomatis dengan waktu yang ditentukan, namun cara ini tidak efektif. Hal ini dapat dihindari jika waktu yang menentukan lama lampu lalu lintas menyala tergantung dari kepadatan jalan tersebut. Dengan berkembangnya teknologi, sebuah komputer dapat mendeteksi sebuah objek dalam suatu gambar atau video. Bidang dalam teknologi komputer yang dapat memproses gambar atau video adalah *Computer Vision*. *Computer Vision* dapat memberi kemampuan kepada sebuah komputer layaknya mata manusia. Dengan membagi *frame* video menjadi *grid* yang berisi 11(horizontal) x 10(vertical) cell dan tiap cell memiliki ukuran 44x44 pixel. Setiap cell akan dicari fiturnya dengan menggunakan *Histogram of Oriented Gradients*(HOG) dan *Local Binary Pattern*(LBP) dan setelah itu akan di klasifikasi dengan *Support Vector Machine*(SVM) menjadi kelas positif(kelas *Traffic*) dan kelas negatif(kelas *Not-Traffic*). Dengan dataset yang digunakan ada beberapa *cell* yang salah masuk kelas sehingga mendapat akurasi tertinggi sebesar 83%.

***Kata Kunci*— Kepadatan jalan, *Image Processing, Computer Vision, Histogram of Oriented Gradient, Local Binary Pattern, Support Vector Machine*.**

I. PENDAHULUAN

Lampu lalu lintas memiliki peranan yang penting dalam kelancaran lalu lintas. Sebagian besar lampu lalu lintas di Indonesia di kendalikan secara otomatis dengan waktu yang ditentukan, namun cara ini tidak efektif karena akan terjadi keadaan dimana waktu masih tersisa banyak sedangkan kendaraan yang berada di jalur tersebut hanya sedikit dan juga sebaliknya, waktu sedikit namun kendaraan banyak. Hal ini dapat dihindari jika waktu yang menentukan lama lampu lalu lintas menyala tergantung dari kepadatan jalan tersebut. Dengan berkembangnya teknologi, sebuah komputer dapat mendeteksi sebuah objek dalam suatu gambar atau video. Bidang dalam teknologi komputer yang dapat memproses gambar atau video adalah *Computer Vision*. *Computer Vision* dapat memberi kemampuan kepada sebuah komputer layaknya mata manusia. Dengan menggunakan *Computer Vision*, sebuah komputer dapat mendeteksi objek-objek yang tertangkap sebuah kamera atau CCTV. Hal ini dapat dimanfaatkan untuk membangun sebuah sistem pengaturan lampu lalu lintas otomatis yang dapat mendeteksi kendaraan dan menghitung kepadatan jalan tersebut. Ada beberapa metode yang dapat digunakan untuk mendeteksi kepadatan jalan.

Pada penelitian yang difokuskan , penelitian ini menggunakan metode gabungan dari *Histogram of Oriented Gradients*(HOG), *Local Binary Pattern*(LBP), dan *Support Vector Machine*(SVM)[1]. Penelitian lainnya memakai *Convolutional Neural Network* yang memiliki 6 *layer*; *Convolution Layer, ReLU Layer, Max-Pooling Layer, Fully Connected Layer, Softmax Layer, dan Classification Layer*[2].

Penelitian lain menggunakan metode *Background Substraction* dan *Invariant Charlier moments*. Tahap pendeteksian dimulai dari pemodelan *background* dari semua gambar yang diambil dari video laju jalan. Setelah itu, akan dilakukan *background substraction* dari gambar *input* dan model *background* untuk mendeteksi kendaraan dan akan menghasilkan gambar biner. Lalu gambar biner tersebut akan diubah menjadi gambar *gray scale*. Lalu *region* kendaraan yang sudah terdeteksi akan dimasukkan ke dalam kotak untuk menandai bahwa *region* tersebut adalah kandidat untuk klasifikasi, dan bayangan kendaraan yang berada di *region* tersebut akan dihilangkan[3]. Penelitian lain menggunakan metode *Image Substraction*. Pada studi ini, ada 2 sistem yang digunakan, sistem yang digunakan pada siang dan malam. Pada sistem yang siang, pertama gambar RGB akan diubah ke gambar *gray scale*. Setelah itu *background* dan *foreground*

dari gambar tersebut akan di ekstrak dengan *mask* yang memiliki koordinat spesifik untuk jalan yang sudah ditentukan. Kemudian, *foreground* akan di *substract* dengan *background* dan juga dilakukan sebaliknya. Ini dilakukan karena terkadang jika melakukan *substraction* dari *pixel* yang lebih gelap dengan *pixel* yang lebih terang akan menghasilkan nilai negatif. Kemudian kedua gambar tersebut akan diubah ke gambar biner. Setelah keduanya diubah menjadi gambar biner, kedua gambar tersebut akan digabungkan dan membuang pixel yang bernilai negatif. Pada sistem untuk malam hari, hanya *foreground* dari gambar yang akan di ekstrak. Setelah itu, gambar akan diubah menjadi gambar biner dengan menggunakan nilai treshold yang tinggi supaya hanya lampu bagian depan kendaraan yang akan muncul[4].

Tujuan dari penelitian ini untuk menganalisis kepadatan di jalan secara real-time dan otomatis dengan menggunakan *Computer Vision*. Metode yang akan digunakan dalam penelitian ini adalah gabungan 3 metode, yaitu; *Histogram of Oriented Gradients* (HOG), *Local Binary Pattern* (LBP), dan *Support Vector Machine* (SVM). HOG dan LBP digunakan untuk mengambil fitur objek, sedangkan SVM digunakan untuk klasifikasi.

II. METODOLOGI

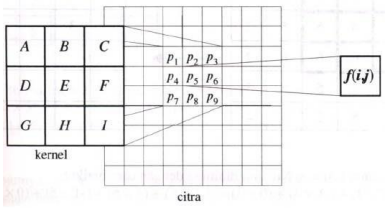
A. *Pengolahan Citra*

Pengolahan citra merupakan bidang dalam Ilmu Komputer yang menggunakan gambar sebagai input lalu gambar tersebut akan diolah dan akan menghasilkan gambar yang baru[5].

Pengolahan gambar untuk *computer vision* dapat dibagi menjadi proses tingkat rendah, menengah dan tinggi. Proses tingkat rendah meliputi operasi pengolahan gambar yang sederhana seperti penghilangan *noise*, mempertajam gambar. Proses tingkat menengah biasanya mencangkup segmentasi, pengenalan objek. Proses tingkat tinggi membuat komputer mampu untuk memahami dan menganalisa gambar[5].

B. *Konvolusi*

Konvolusi gambar teknik untuk menghaluskan suatu gambar atau memperjelas gambar dengan menggantikan nilai pixel dengan sejumlah nilai *pixel* yang sesuai atau berdekatan dengan *pixel* aslinya. Dalam konvolusi, biasanya akan digunakan sebuah *operator* yang disebut *filter,mask,* atau *kernel*[12].



Gambar 2.1 Ilustrasi Konvolusi

$$f(i,j) = Ap_1 + Bp_2 + Cp_3 + Dp_4 + Ep_5 + Fp_6 + Gp_7 + Hp_8 + Ip_9 \quad (2.1)$$

dimana:

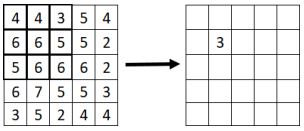
F(i,j) = posisi *pixel* yang akan dikonvolusi

A,B,C,..,H,I = nilai *kernel*

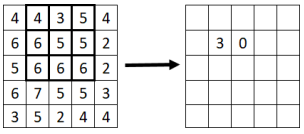
p1,p2,p3,...,p8,p9 = nilai *pixel* gambar

Operasi konvolusi dapat diilustrasikan sebagai berikut :

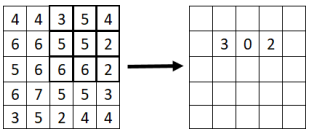
1. Terapkan *kernel* pada gambar pada posisi (0,0)



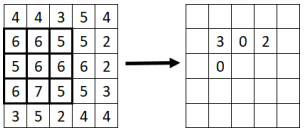
2. Geser satu *pixel* ke kanan dan terapkan *kernel*



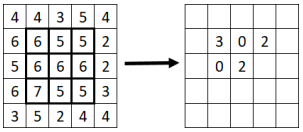
3. Geser satu *pixel* ke kanan dan terapkan *kernel*



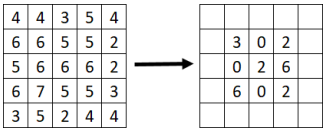
4. Geser satu *pixel* ke bawah dan terapkan *kernel* dari sisi kiri gambar



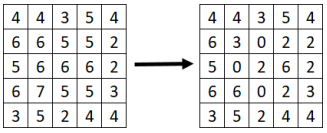
5. Geser satu *pixel* ke kanan dan terapkan *kernel*



6. Lakukan hal yang sama dengan *pixel* selanjutnya sehingga menghasilkan seperti gambar dibawah



7. Untuk nilai pinggir, ada beberapa cara yang dapat dilakukan. Salah satunya yaitu tidak dikonvolusi sehingga nilai pinggir *pixel* sama dengan nilai dari gambar asli



C. *Region of Interest(ROI)*

Region of Interest(ROI) adalah sebuah teknik dalam pengolahan citra dimana pengguna dapat mengolah data yang hanya mengandung informasi/gambar yang diinginkan. ROI bekerja dengan cara dengan mengambil bagian tertentu dalam

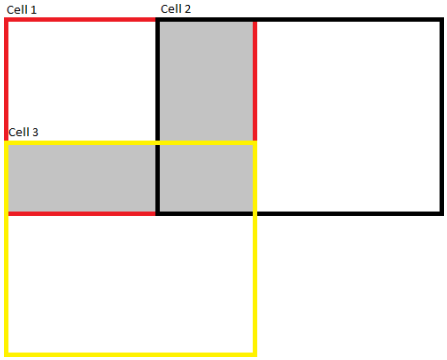
gambar sehingga bagian gambar yang tidak penting dalam penelitian akan dibuang.

Gambar 2.2 Contoh ROI

Gambar 2.2 menunjukkan contoh gambar asli dan bagian dalam kotak putih menunjukkan ROI yang diambil untuk dipakai dalam penelitian.

D. Intersection Over Union(IOU)

Intersecion over Union (IOU) adalah sebuah istilah yang menggambarkan daerah *overlap* 2 kotak. Dalam suatu *grid* yang berisi *cell*, jika ada satu *cell* yang *overlap* dengan *cell* yang lain, maka daerah *overlap* antara 2 kotak tersebut disebut *Intersection over Union* (IOU). *Intersection over Union* biasanya digunakan untuk mengukur akurasi hasil prediksi dalam mendeteksi sebuah objek berdasarkan daerah *overlap* 2 kotak.



Gambar 2.3 *Overlapped Cell*

Dalam Gambar 2.3, ada 3 cell yang saling *overlap*. Daerah yang berwarna abu menunjukkan IOU antara 3 *cell*.

E. Histogram of Oriented Gradient(HOG)

Histogram merupakan sebuah diagram yang menunjukkan frekuensi munculnya suatu nilai. *Histogram* gambar adalah histogram yang menunjukkan distribusi nilai intensitas *pixel* dari suatu gambar. Dari sebuah histogram dapat diketahui banyak hal tentang gambar tersebut, misalnya kecerahan atau kontras gambar tersebut[5]. *Histogram Of Oriented Gradients* (HOG) adalah *feature descriptor* yang digunakan untuk deteksi objek. HOG digunakan dalam *Computer Vision* dan *Image Processing* untuk mendeteksi objek. Teknik menghitung kemunculan orientasi gradien di bagian gambar yang dilokalkan. Deskriptor HOG berfokus pada struktur atau bentuk suatu objek. Ini lebih baik daripada deskriptor tepi mana pun karena menggunakan besaran serta sudut gradien untuk menghitung fitur. Untuk wilayah gambar itu menghasilkan histogram menggunakan besaran dan orientasi gradien[7]. Dalam gambar 3x3, tiap pixel akan dicari nilai gradiennya (Gx, Gy). Nilai Gx dan Gy dihitung dengan menggunakan sebuah *kernel* dan dihitung dengan menggunakan konvolusi. *Kernel* yang digunakan untuk menghitung nilai gradien dalam penelitian ini adalah *kernel Sobel*.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \qquad (2.2)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I \qquad (2.3)$$

dimana :

I = nilai *pixel* sebuah gambar

Jika Gx dan Gy sudah didapat, maka *magnitude* dan sudut(*angle*) tiap pixel dihitung dengan persamaan berikut :

$$Magnitude(\mu) = \sqrt{G_x^2 + G_y^2} \qquad (2.4)$$

atau bisa juga :

$$Magnitude(\mu) = |G_x| + |G_y| \qquad (2.5)$$

Dan persamaan untuk mencari *angle* :

$$Angle(\theta) = |\tan^{-1}((G_y/G_x))| \qquad (2.6)$$

dimana:

G_x = nilai gradien *pixel* gambar ke arah x

G_y = nilai gradien *pixel* gambar ke arah y

Setelah mendapatkan gradien setiap *pixel*, matriks gradien (matriks *magnitude* dan sudut) dibagi menjadi cell 8x8 untuk membentuk *block*. Untuk setiap *block*, akan dihitung 9-titik *histogram*. 9-titik *histogram* mengembangkan *histogram* dengan 9 *bin* dan setiap bin memiliki rentang sudut 20 derajat. Masing-masing 9-titik histogram ini dapat diplot sebagai histogram dengan bin yang menghasilkan intensitas gradien di *bin* tersebut. Karena sebuah *block* berisi 64 nilai yang berbeda, untuk semua 64 nilai *magnitude* dan gradien, perhitungan berikut dilakukan.

$$Jumlah\ bin = 9(mulai\ dari\ 0^\circ\ sampai\ 180^\circ) \qquad (2.7)$$

$$Jumlah\ step(\Delta\theta) = 180^\circ / jumlah\ bin = 20^\circ \qquad (2.8)$$

Setiap *bin* ke-j akan memiliki batasan dari :

$$[\Delta\theta \cdot j, \Delta\theta \cdot (j+1)] \qquad (2.9)$$

Dimana :

j=indeks *bin*

Δθ =Jumlah Step

Nilai tengah dari setiap *bin* akan menjadi :

$$C_j = \Delta\theta(j+0.5) \quad (2.10)$$

Dimana :

j=indeks *bin*

Δθ=Jumlah step

C_j=Nilai pusat *bin*

Untuk setiap *cell* dalam sebuah *block*, akan dihitung *bin* ke-j dan kemudian nilai tersebut akan diberikan masing-masing ke *bin* ke-j dan (j+1). Nilai diberikan oleh persamaan berikut:

$$j = \lfloor(\frac{\theta}{\Delta\theta} - \frac{1}{2})\rfloor \quad (2.11)$$

$$V_j = \mu \cdot [\frac{\theta}{\Delta\theta} - \frac{1}{2}] \quad (2.12)$$

$$V_{j+1} = \mu \cdot [\frac{\theta - C_j}{\Delta\theta}] \quad (2.13)$$

Dimana :

j=indeks *bin*

Δθ=Jumlah step

μ=Magnitude

C_j=Nilai pusat *bin*

Array diambil sebagai *bin* untuk *block* dan nilai Vj dan Vj+1 ditambahkan dalam *array* pada indeks bin ke-j dan (j+1) yang dihitung untuk setiap *pixel*. Matriks yang dihasilkan setelah perhitungan di atas akan berbentuk 16x8x9. Setelah komputasi *histogram* selesai untuk semua *block*, 4 *block* dari matriks *histogram* 9 titik disatukan untuk membentuk *block* baru (2x2). *Clubbing* ini dilakukan secara *overlapping* dengan langkah 8 *pixel*. Untuk semua 4 sel dalam satu *block*, semua *histogram* 9 titik kan digabungkan untuk setiap sel penyusun untuk membentuk 36 vektor fitur.

$$f_{bi} = [b_1, b_2, b_3, ..., b_{36}] \quad (2.14)$$

Dimana :

$f_(bi)$=vektor fitur

Nilai fb untuk setiap *block* dinormalisasi dengan L2 *norm* :

$$f_{bi} \leftarrow \frac{f_{bi}}{\sqrt{\|f_{bi}\|^2 + \varepsilon}} \quad (2.15)$$

Dimana :

$f_(bi)$=vektor fitur

ε = nilai kecil yang ditambahkan ke kuadrat fb untuk menghindari pembagian nol.

Untuk menormalisasikan, nilai k terlebih dahulu dihitung dengan persamaan berikut :

$$k = \sqrt{b_1^2 + b_2^2 + b_3^2 + ... + b_{36}^2} \quad (2.16)$$

$$fbi = \left[\left(\frac{b1}{k}\right), \left(\frac{b2}{k}\right), \left(\frac{b3}{k}\right), ..., \left(\frac{b36}{k}\right)\right] \quad (2.17)$$

Dimana :

$f_(bi)$=vektor fitur

k = nilai yang didapat dari Persamaan 2.16

Normalisasi ini dilakukan untuk mengurangi efek perubahan kontras antara gambar objek yang sama. Dari setiap *block*. Sebuah vektor fitur 36 titik dikumpulkan. Pada arah mendatar terdapat 7 *block* dan pada arah vertikal terdapat 15 *block*. Jadi panjang total fitur HOG adalah : 7 x 15 x 36 = 3780.

F. *Local Binary Pattern(LBP)*

Local Binary Patterns (LBP) adalah operator sederhana namun sangat efisien yang melabeli *pixel* dari suatu gambar dengan mempertimbangkan properti dari *pixel* tetangganya. Ide dasar di balik LBP adalah bahwa gambar terdiri dari beberapa pola mikro. LBP merupakan turunan dari *first-order circular* dari pola yang dihasilkan dengan menggabungkan arah gradien biner. *Histogram* dari pola mikro ini berisi informasi tentang distribusi tepi dan fitur lokal lainnya dalam sebuah gambar. Properti lain yang sama pentingnya adalah kesederhanaan komputasinya, yang memungkinkan untuk menganalisis gambar dalam pengaturan waktu nyata[6].

Dalam domain 3 × 3 operator LBP ambang batas delapan *pixel* periferal dari lingkungan pada nilai *pixel* pusat, sehingga mendefinisikan satu set 28 pola biner yang mungkin. Berikut adalah persamaan LBP untuk mencari nilai s(x) pada setiap pixel tetangga :

$$LBP(N,R) = \sum_{n=0}^{N-1} s(I_n - I_c)2^n \quad (2.18)$$

$$s(x) = 1, jikax \geq 0$$

$$s(x) = 0, jikax < 0$$

dimana :

c = *pixel* tengah dalam matriks 3x3

n = *pixel* tetangga dari *pixel* c

Untuk mendapat nilai LBP pada *pixel* tengah, harus dicari dengan persamaan berikut :

$$LBP = (2^0) * s(x_0) + (2^1) * s(x_1) + ... + (2^7) * s(x_7) \quad (2.19)$$

G. *Machine Learning*

Machine Learning (ML) yaitu teknik untuk melakukan inferensi terhadap data dengan pendekatan matematis. Inti *machine learning* adalah untuk membuat model (matematis) yang merefleksikan pola-pola data. Tujuan *machine learning* adalah untuk memprediksi masa depan dan/atau memperoleh ilmu pengetahuan. Dari sisi metode pembelajaran, algoritme *machine learning* dapat dikategorikan sebagai : *supervised learning*, *semi-supervised learning*, *unsupervised learning*, dan *reinforcement learning*[9].

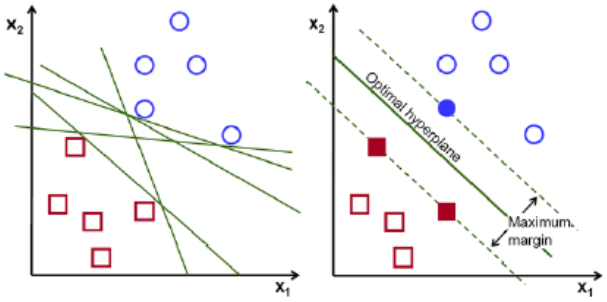
Supervised Learning, jika diterjemahkan adalah pembelajaran terarah/terawasi. Dalam *supervised learning*, mesin akan diawasi/diarahkan supaya dapat mengeluarkan *output* yang diinginkan. Salah satu contoh algoritme *supervised learning* adalah *Naive Bayes*, *Iterative Dichotomiser* (ID3), *Support Vector Machine* (SVM).

Dalam *semi-supervised learning*, mesin tidak akan diberi *output* yang diinginkan. Namun, mesin akan memperoleh *output* yang diinginkan secara otomatis (misal dengan *clustering*).

Dalam *unsupervised learning*, mesin tidak diarah/diawasi. *Unsupervised learning* bekerja dengan mesin mencari dan mengelompokkan sifat/karakter data yang mirip. Salah satu bentuk *unsupervised learning* adalah *Clustering*. Contoh algoritme *unsupervised learning* adalah *K-Means*.

H. Support Vector Machine

Tujuan dari *Support Vector Machine* adalah untuk menemukan *hyperplane* dalam ruang N-dimensi (N adalah jumlah fitur) yang secara jelas mengklasifikasikan titik data.



Gambar 2.4 Kemungkinan *Hyperplane*[8]

Untuk memisahkan kedua kelas titik data tersebut, terdapat banyak kemungkinan *hyperplane* yang dapat dipilih, salah satunya dengan menemukan bidang yang memiliki margin maksimum. Memaksimalkan jarak *margin* akan memberikan beberapa penguatan sehingga titik data lain dapat diklasifikasikan dengan lebih pasti. *Hyperplanes* adalah batas keputusan yang membantu mengklasifikasikan titik data. Titik data yang jatuh di kedua sisi *hyperplane* dapat dikaitkan dengan kelas yang berbeda. Dimensi *hyperplane* tergantung pada jumlah fitur. Jika jumlah fitur input adalah 2, maka *hyperplane* hanya akan membentuk sebuah garis. Jika jumlah fitur *input* adalah 3, maka *hyperplane* menjadi bidang dua dimensi. *Support vector* adalah titik data yang lebih dekat dengan *hyperplane* dan mempengaruhi posisi dan orientasi *hyperplane*. Dengan menggunakan vektor pendukung ini, kami memaksimalkan margin pengklasifikasi. Menghapus *support vector* akan mengubah posisi *hyperplane*[8]. SVM menggunakan satu set fungsi matematika yang didefinisikan sebagai *kernel*. *Fungsi kernel* adalah untuk mengambil data sebagai *input* dan mengubahnya menjadi bentuk yang diperlukan. Algoritme SVM yang berbeda menggunakan berbagai jenis fungsi kernel. Fungsi-fungsi ini dapat menjadi jenis yang berbeda. Misalnya *linear*, *nonlinear*, *polynomial*, *radial basis function*(RBF), dan *sigmoid*. Jenis fungsi *kernel* yang paling banyak digunakan adalah RBF. Karena memiliki respons terlokalisasi dan terbatas di sepanjang sumbu x.Fungsi *kernel* mengembalikan produk dalam antara dua titik dalam ruang fitur yang sesuai. Jadi dengan mendefinisikan gagasan kesamaan, dengan sedikit biaya komputasi bahkan dalam ruang dimensi yang sangat tinggi[10].

I. Kernel yang Digunakan

Kernel atau *kernel function* berguna untuk mentransformasi data menjadi lebih tinggi / menambah fitur data[9]. Dari beberapa jenis *kernel*, yang akan digunakan dalam penelitian ini adalah *linear kernel*. *Linear Kernel* adalah jenis *kernel* yang paling dasar, biasanya satu dimensi. Ini terbukti menjadi fungsi terbaik ketika ada banyak fitur. *Linear kernel* dipilih menjadi *kernel* yang akan digunakan dalam penelitian ini karena *kernel linear* merupakan *kernel* yang paling sederhana dan *kernel* ini bekerja dengan baik jika data yang digunakan hanya memiliki 2 kelas (kelas positif dan kelas negatif). Berikut adalah persamaannya :

$$F(x,x_j) = sum(x \cdot x_j) \qquad (2.20)$$

dimana x dan xj merupakan data yang akan diklasifikasi.

J. Confusion Matrix

Confusion Matrix digunakan untuk mendeskripsikan performa klasifikasi sebuah model. Ada 4 variabel dalam *confusion matrix* yang digunakan untuk menentukan performa sebuah model. 4 variabel tersebut adalah :

1. *True Positive* (TP) adalah bagian dari kelas positif(+) dan hasil prediksi adalah kelas positif(+)
2. *True Negative* (TN) adalah bagian dari kelas negatif(-) dan hasil prediksi adalah kelas negatif(-)
3. *False Positive* (FP) adalah bagian dari kelas positif(+) dan hasil prediksi adalah kelas negatif(-)
4. *False Negative* (FN) adalah bagian dari kelas negatif(-) dan hasil prediksi adalah kelas positif(+)

Untuk menghitung performa dengan *confusion matrix*, akan digunakan sebuah *table*:

		Actual Values: Positive (1)	Actual Values: Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Gambar 2.5 *Confusion Matrix table*[11]

Pertama, nilai awal setiap variabel di set menjadi 0. Setiap data yang masuk ke variabel tertentu, nilai variabel tersebut ditambah 1. Setelah setiap data dimasukkan kedalam *confusion matrix table*, maka akan dicari nilai *recall*. Nilai *recall* dapat dijelaskan sebagai, dari semua kelas positif berapa banyak hasil prediksi yang benar. Nilai *recall* dapat dihitung dengan persamaan berikut :

$$Recall = \frac{TP}{TP + FN} \qquad (2.21)$$

dimana:

TP = *True Positive*

FN = *False Negative*

Setelah itu, akan dicari nilai *precision.* Nilai *precision* dapat dijelaskan sebagai, dari semua data yang diprediksi sebagai positif berapa banyak yang benar positif. Nilai *precision* dapat dihitung dengan persamaan berikut :

$$Precision = \frac{TP}{TP+FP} \quad (2.22)$$

dimana:

TP = *True Positive*

FP = *False Positive*

Setelah didapat nilai *recall* dan *precision*, nilai *F-Score* akan dihitung untuk dapat menentukan performa sebuah model. Nilai *F-Score* dapat dihitung dengan persamaan berikut :

$$F-Score = \frac{2*Recall*Precision}{Recall+Precision} \quad (2.23)$$

Untuk mencari akurasi, dapat menggunakan persamaan berikut

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.24)$$

dimana:

TP = *True Positive*

TN = *True Negative*

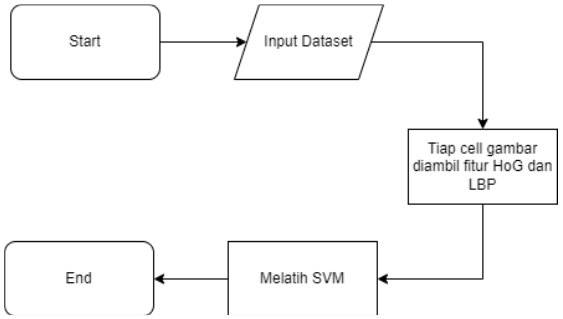
FP = *False Positive*

FN = *False Negative*

K. Perancangan Sistem

1. Urutan Proses Global (*Flowchart*)

Urutan proses global dalam sistem akan dibagi menjadi 2, yaitu untuk *training* dan *testing*. Berikut uraian *flowchart* yang akan digunakan dalam tahap *training* :



Gambar 2.6 *Flowchart Training*

Berikut penjelasan tentang *flowchart training* :

1. *Input dataset*

Untuk *training*, *dataset* yang digunakan merupakan gambar *cell* yang berukuran 44x44 *pixel*. *Dataset* berjumlah 20915 gambar dan dimasukkan ke dalam 2 kelas; *Traffic* dan *Not-Traffic*.

2. Mengekstrak fitur HoG dari tiap gambar

Pada setiap gambar, akan diambil fitur HOG. Setiap gambar akan dibagi menjadi *sub cell* yang berukuran 3x3 dan dihitung dengan 8 orientasi. Fitur HOG yang akan didapat merupakan fitur *vector* sebanyak 5408.

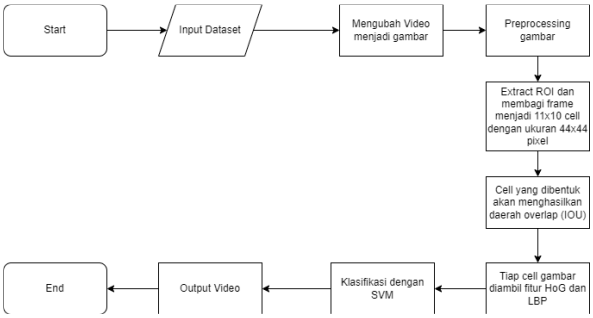
3. Mengekstrak fitur LBP dari tiap gambar

Fitur LBP diambil dari tiap gambar. LBP hanya bisa menerima gambar *grayscale* sebagai *input*. LBP memerlukan 2 parameter yang harus disediakan, parameter untuk radius dan parameter untuk jumlah titik di sekitar radius luar yang nilainya bergantung pada berbagai faktor seperti ukuran sel. Fitur LBP akhir yang didapat sebanyak 26.

4. Melatih SVM

Fitur HOG dan LBP yang telah didapat akan digunakan untuk klasifikasi. Klasifikasi dilakukan menggunakan SVM dengan kernel linear.

Berikut uraian *flowchart* yang digunakan dalam tahap *testing* :



Gambar 2.7 *Flowchart Testing*

Berikut penjelasan tentang *flowchart testing* :

1. *Input dataset*

Untuk testing, *dataset* yang digunakan merupakan 7 video yang masing memiliki 25 fps dan memiliki durasi 5 menit

2. Menentukan Region of Interest (RoI)

Setiap frame ke-8 dari video, akan diambil framenya. Frame tersebut akan dibagi menjadi grid dengan ukuran 10x11 cell, dan tiap cellnya berukuran 44x44 pixel.

3. *Cell Overlap*

Cell yang dibentuk akan *overlap* dengan *cell* lainnya dan akan menghasilkan daerah *overlap*(IOU) yang berukuran 10x44 *pixel* untuk *cell* yang *overlap* ke samping dan 44x10 *pixel* untuk *cell* yang *overlap* ke bawah.

4. Mengekstrak fitur HoG dari tiap cell

Pada setiap *cell*, akan diambil fitur HOG. Setiap *cell* akan dibagi menjadi *sub cell* yang berukuran 3x3 dan dihitung dengan 8 orientasi. Fitur HOG yang akan didapat merupakan fitur *vector* dengan 5408.

5. Mengekstrak fitur LBP dari tiap *cell*

Fitur LBP diambil dari tiap *cell*. LBP hanya bisa menerima gambar *grayscale* sebagai *input*. LBP memerlukan 2 parameter yang harus disediakan, parameter untuk radius dan parameter untuk jumlah titik di sekitar radius luar yang nilainya bergantung pada berbagai faktor seperti ukuran sel. Fitur LBP akhir yang didapat sebanyak 26.

6. Klasifikasi dengan SVM

Setiap *cell* yang sudah dicari nilai HOG dan LBPnya akan diklasifikasi dengan model yang didapat dari tahap *training*.

7. *Output* Video

Output merupakan video *real-time* dan setiap *frame* ke-8 akan dibentuk *grid* dengan ukuran 10x11 *cell*. Setiap *cell* akan diberi border merah/hijau tergantung dari hasil klasifikasi.

2. Dataset

Dataset diambil dari *Queen Mary University of London*. Ada 2 *dataset* yang digunakan dalam penelitian ini. *Dataset* yang digunakan untuk training merupakan kumpulan gambar yang berjumlah 20915 gambar, masing-masing gambar berukuran 44x44 dan dibagi menjadi 2 kelas; *Traffic* dan *Not-Traffic*. Kelas *Traffic* memiliki 5617 gambar, dan kelas *Not-Traffic* memiliki 15298 gambar. Gambar tersebut diambil dari video yang digunakan dalam *testing*.

Gambar 2.8 *Dataset training*[1]

Gambar 2.8 merupakan contoh gambar *dataset* yang akan digunakan dalam tahap *training*. Gambar sebelah kiri merupakan contoh gambar dalam kelas *Not-Traffic*, dan gambar sebelah kanan merupakan contoh gambar kelas *Traffic*.

Dataset yang digunakan untuk *testing* merupakan video CCTV jalan yang memiliki durasi 52 menit dengan dimensi 360x268 dengan 25 fps dan dibagi menjadi 7 instan yang masing-masing memiliki durasi 5 menit.

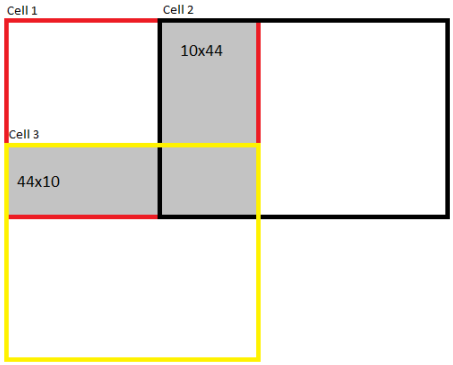
Gambar 2.9 *Dataset testing*[1]

Gambar 2.9 menunjukkan contoh *frame* dari video yang akan digunakan dalam tahap *testing*.

III. HASIL DAN PEMBAHASAN

Pada tahap pelatihan, akan digunakan dataset gambar yang diambil dari video *dataset*. Gambar yang digunakan merupakan RoI dari video *dataset* yang berukuran 44 x 44 *pixel* dan telah di ubah menjadi *grayscale*. Dari setiap gambar tersebut akan dicari nilai fitur LBP dan HOG dan akan disimpan dalam sebuah file *pickle*. Tahap pelatihan akan menggunakan library untuk membentuk sebuah model dari nilai fitur LBP dan HOG yang telah didapat.

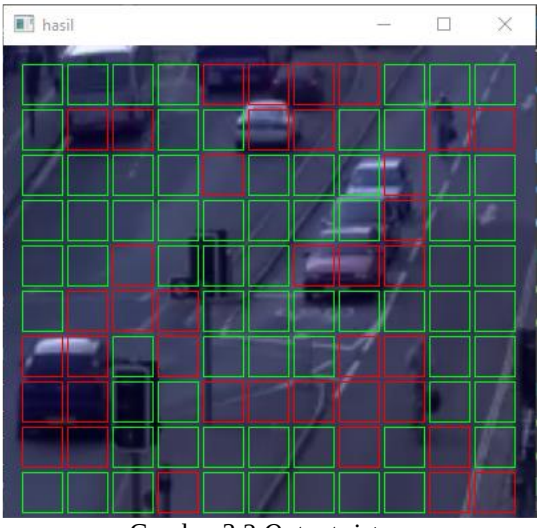
Dalam tahap pengujian, video dataset akan digunakan. Setiap 8 *frame* dari video akan digunakan dalam pengujian. Setiap *cell*(44x44 pixel) dari *frame* yang diambil akan dicari nilai fitur HOG dan LBPnya. Tahap ini akan terus dilakukan sampai didapat 11(*horizontal*)x10(*vertical*) *cell*, dan tiap *cell* akan bergerak ke kanan/bawah setiap 34 *pixel*. Dengan cara ini, *cell* yang ada pasti akan *overlap* dengan *cell* sebelah/bawah, dan ini akan menghasilkan daerah *overlap*(IOU) yang berukuran 10x44 *pixel* untuk *cell* yang *overlap* ke samping dan 44x10 *pixel* untuk *cell* yang *overlap* ke bawah.



Gambar 3.1 *Cell Overlap*

Gambar 4.2 menunjukkan *cell* yang dibentuk dan saling *overlap* dengan *cell* yang lain.

Daerah berwarna abu menunjukkan daerah *overlap*(IOU). Setelah itu, dengan menggunakan *library* akan di prediksi mengunakan model yang didapat dari tahap pelatihan. Jika *frame* tersebut merupakan *traffic*, *frame* tersebut akan diberi border merah, dan jika *frame* tersebut bukan merupakan *traffic*, maka *frame* tersebut akan diberi border hijau.



Gambar 3.2 *Output* sistem

Untuk menghitung akurasi, model SVM yang sama akan di *test* dengan *frame* pertama dari masing-masing video *dataset testing*. Berikut hasil analisa akurasi dengan menggunakan *confusion matrix*.

Tabel 3.1 Hasil *Confusion Matrix* dari 7 instan video

Data	TP	FP	TN	FN	Recall	Precision	F-Score	Akurasi
Video 1	51	5	40	14	91%	78%	84%	83%
Video 2	32	9	55	14	78%	70%	74%	79%
Video 3	39	18	38	15	68%	72%	70%	70%
Video 4	30	18	52	10	63%	75%	68%	75%
Video 5	66	7	15	22	90%	75%	82%	74%
Video 6	39	14	42	15	74%	72%	73%	74%
Video 7	34	11	40	25	76%	58%	65%	67%

Dari hasil pengujian diatas, didapat nilai akurasi tertinggi sebesar 83% dan terendah sebesar 67%.

Dari hasil *output* yang didapat, ada beberapa *cell* yang terdeteksi salah kelas; *cell Not-Traffic* terdeteksi sebagai *cell Traffic*, dan sebaliknya. Beberapa penyebab munculnya salah klasifikasi adalah tidak seimbangnya jumlah data training yang digunakan dan adanya bagian kecil kendaraan yang muncul dalam sebuah *cell*.

IV. SIMPULAN

Dari penelitian ini, dapat disimpulkan bahwa kombinasi dari pengambilan fitur dengan metode HOG dan LBP dan klasifikasi menggunakan SVM dapat digunakan untuk pendeteksian kepadatan di jalan. Dengan menerapkan ROI terhadap *dataset* dan membaginya menjadi sebuah *grid* yang berisi beberapa *cell* dengan ukuran tertentu, maka didapat hasil klasifikasi dengan nilai akurasi tertinggi sebesar 83% dan terendah sebesar 67%.

DAFTAR REFERENSI

[1] D. Prasad, K. Kapadni, A. Gadpal, M. Visave and K. Sultanpure, ”HOG, LBP and SVM based Traffic Density Estimation at Intersection,” 2019 IEEE Pune Section International Conference (PuneCon), 2019, pp. 1-5, doi: 10.1109/PuneCon46936.2019.9105731

[2] I. A. Tarmizi and A. A. Aziz, ”Vehicle Detection Using Convolutional Neural Network for Autonomous Vehicles,” 2018 International Conference on Intelligent and Advanced System (ICIAS), 2018, pp. 1-5, doi: 10.1109/ICIAS.2018.8540563.

[3] S. Aqel, A. Hmimid, M. A. Sabri and A. Aarab, ”Road traffic: Vehicle detection and classification,” 2017 Intelligent Systems and Computer Vision (ISCV), 2017, pp. 1-5, doi: 10.1109/ISACV.2017.8054969.

[4] P. N. Chowdhury, T. Chandra Ray and J. Uddin, ”A Vehicle Detection Technique for Traffic Management using Image Processing,” 2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2), 2018, pp. 1-4, doi: 10.1109/IC4ME2.2018.8465599.

[5] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 2nd ed. Prentice Hall, 1992.

[6] S. Brahnam, L. C. Jain, L. Nanni, A. Lumini, Local Binary Patterns: New Variants and Applications, 2014

[7] M. Tyagi, ”HOG (Histogram of Oriented Gradients): An Overview”, towardsdatascience, 2020. [Online]. Available:https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f/. [Accessed:22-Dec-2021]

[8] R. Gandhi, ”Support Vector Machine — Introduction to Machine Learning Algorithms”, towardsdatascience, 2018. [Online]. Available:https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47/. [Accessed:22-Dec-2021]

[9] J. W. G. Putra, Pengenalan Konsep Pembelajaran Mesin dan Deep Learning, 2020

[10] S. Awasthi, ”SEVEN MOST POPULAR SVM KERNELS”, dataaspirant.[Online].Available:https://dataaspirant.com/svm-kernels/#t- 1608054630727/.[Accessed:29-Dec-2021]

[11] S. Narkhede, ”Understanding Confusion Matrix”, towardsdatascience, 2018.[Online].Available:https://towardsdatascience.com/understanding - confusion-matrix-a9ad42dcfd62.[Accessed:30-Jun-2022]

[12] W. Gazali, H. Soeparno, J. Ohliati, ”Penerapan Metode Konvolusi Dalam Pengolahan Citra Digital”, 2012

[13] A. A. Al-Sobky, R. M. Mouse, ”Traffic density determination and its applications using smartphone”, 2016 Alexandria Engineering Journal 55, 2016, pp. 513-523, doi: 10.1016/j.aej.2015.12.010

[14] Victor L. Knoop, Winnie Daamen, ”Automatic fitting procedure for the fundamental diagram”, 2017 Transpormetrica B: Transport Dynamics, 2016, pp. 129-144, doi : 10.1080/21680566.2016.1256239