BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Bidang penelitian deteksi dan pengenalan rambu lalu lintas secara garis besar terbagi atas 2 bagian. Bagian pertama yaitu tahap pendeteksian keberadaan rambu lalu lintas dijalan. Tujuannya yaitu untuk memisahkan antara rambu lalu lintas dan latar belakang yang begitu kompleks. Bagian kedua yaitu tahap pengenalan. Rambu lalu lintas yang terdeteksi kemudian dikenali dan diklasifikasikan sesuai dengan yang disimpan dalam image database (Azad et al., 2014; Houben et al., 2013; Mogelmose et al., 2012).

Banyak penelitian yang dilakukan dalam pendeteksian rambu lalu lintas, menerapkan metode deteksi warna dan deteksi bentuk. Wu et al., (2007) dalam penelitiannya menggunakan metode yang mengambil informasi warna dari citra rambu lalu lintas. Pertama kali citra dikonversi kedalam citra HSV untuk mendeteksi keberadaan rambu lalu lintas. Selanjutnya diterapkan metode *morphology* untuk menghasilkan citra lalu lintas yang lebih jelas. Setelah itu citra diekstrak menggunakan sistem *geometric property.* Sebanyak 196 citra yang berhasil di kenali dengan benar dari pengujian dengan menggunakan sebanyak 200 citra dengan ukuran pixel sebesar 640 x 480 (Wu et al., 2007).

Pankaj & Patil (2013) mengusulkan sistem pengenalan dan pendeteksian rambu lalu-lintas berdasarkan segmentasi warna citra dan dikombinasikan dengan analisa bentuk (Pankaj & Patil, 2013). Begitu juga dengan Danti & Kulkarni

(2013) menerapkan segmentasi warna dan pemodelan bentuk dengan *TPS (Thin Spline Transformation)* yang digunakan untuk mencari nilai terdekat ketetanggan pada pendeteksian dan klasifikasi (Danti & Kulkarni, 2013).

Fleyeh, (2006) dalam penelitiannya mengembangkan beberapa teknik pengolahan citra untuk mendeteksi keberadaan rambu lalu lintas pada beberapa kondisi pencahayaan yang bervariasi. Bagian pixel citra dengan tingkat *saturation* dan *value* tertentu yang tergolong sebagai bagian akromatik (tidak bewarna) dan diluar batas tertentu akan dihilangkan untuk mengurangi jumlah *noise*. Citra kemudian dibagi menjadi beberapa bagian sub citra dengan ukuran 16 x16, kemudian dilihat jumlah pixel yang sesuai dengan ketentuan. Jika sub citra mempunyai jumlah pixel merah kurang dari batas yang ditentukan, maka sub citra akan dieliminasi. Hal ini juga dilakukan untuk mengurangi bagian citra yang dianggap sebagai *noise.* Setelah itu citra diproses dengan metode *region growing,* dengan melihat tingkat kemiripan antar tetangga pixel. Percobaan dilakukan dalam berbagai kondisi cuaca yang berbeda seperti; cerah, mendung, berkabut, dan bersalju, dengan tingkat keberhasilan sampai dengan 95% dari total kasus (Fleyeh, 2006).

Ruta et al., (2008) Pada penelitiannya menggunakan pendekatan komprehensif dalam pengenalan rambu lalu lintas dari input video. Untuk memastikan pendeteksian yang stabil terutama pada latar belakang yang rumit, diterapkan pemodelan relevansi pixel, dimana dilakukan pengukuran setiap pixel pada setiap bagian dari permukaan rambu. Pada proses pengklasifikasiannya, Ruta et al., (2008) menerapkan metode *template matching* sedangkan Lorasakul

dan Suthakom, (2007) menggunakan teknik jaringan saraf tiruan untuk mengenali ciri rambu lalu lintas. Citra terlebih dahulu di proses dengan beberapa teknik pengolahan citra seperti *threshold, gaussian filter, canny edge, contour* dan *fit elipse*. Kemudian jaringan saraf tiruan digunakan untuk mengenali rambu lalu lintas tersebut (Ruta et al., 2008; Lorasakul & Suthakom, 2007). Begitu juga dengan Al-Azawi, (2012) dalam penelitiannya membahas mengenai pengenalan citra rambu-rambu lalu lintas menggunakan jaringan saraf tiruan. Jaringan saraf tiruan yang digunakan adalah *backpropagation* yang sudah dimodifikasi (Al-Azawi, 2012).

John, (2004) mengembangkan algoritma untuk pengenalan rambu lalu lintas pada citra digital dengan program An.Si yang disusun dari kata angnorisi simation yang artinya pengenalan tanda. Metode yang dipakai dalam mendeteksi tanda rambu lalu lintas berdasarkan pada karakter geometri gambar dan informasi warna. Pada tahap deteksi, pertama kali citra diubah kedalam citra biner. Kemudian dengan beberapa proses segmentasi lainnya seperti proses *thinning, edge detection* citra diproses untuk mendapatkan daerah citra rambu berada. Pada tahap pengenalan, citra akan dibandingkan dengan *template* yang spesifik. Citra yang akan dibandingkan harus memiliki koordinat yang sama. Rambu lalu lintas yang berbentuk segitiga dan kotak akan diproses dengan teknik *affline transformation.* Rambu lalu lintas yang berbentuk lingkaran, diproses dengan teknik *similiarity transformation*. Hasil yang didapat pada proses deteksi cukup efektif namun masih memiliki kelemahan jika warna citra rambu dengan warna

pada background hampir serupa. Bahasa pemrograman yang digunakan adalah IDL (*interactive development language)* versi 5.4 (John, 2004).

Dean & Jabir, (2013) menggunakan ruang warna YcbCr pada proses segmentasi warna dan teknik template matcing pada proses pencocokan bentuk-bentuk seperti segitiga, lingkaran, dan lainnya dengan objek hasil segmentasi warna. Setelah proses deteksi selesai, proses pengenalan dilakukan menggunakan jaringan saraf tiruan untuk mengenali rambu-rambu lalu lintas tersebut. Proses deteksi mencapai rata-rata 100% untuk klasifikasi warna dan 94% untuk klasifikasi bentuk. Proses pengenalan mencapai rata-rata sekitar 90% dan dapat meningkat jika menaikan jumlah data pelatihan (Dean & Jabir, 2013).

2.2 Landasan Teori

2.2.1 Deteksi dan pengenalan rambu lalu lintas.

Rambu lalu lintas menaruh peranan yang tinggi dalam hal kecelakaan lalu lintas. Deteksi dan pengenalan rambu lalu lintas dapat membantu pengendara dalam memberikan informasi yang bernilai mengenai keamanan dan navigasi, jika dilakukan dengan akurat dan cepat oleh sistem sehingga berdampak pada efisiensi navigasi (Tekadpande & Giri,2012; Bhattacharcya & Giri, 2013).

Choudari et al., (2013) mengemukakan bahwa pendeteksian rambu lalu lintas oleh sistem merupakan sebuah hal penting dalam berkendaraan secara otomatis (Choudari et al., 2013). Sistem pengenalan rambu lalu lintas merupakan bagian dari sistem pembantu pengendara yang secara otomatis memberikan peringatan dan informasi kepada pengendara tentang rambu-rambu lalu lintas

(Dean & Jabir,2013). Pengenalan rambu-rambu lalu lintas adalah sebuah hal yang sulit jika fokusnya pada pendeteksian gambar dari lingkungan yang tidak bersahabat. Latar belakang yang kompleks, faktor cuaca, bayangan dan masalah pencahayaan membuat sulit dalam mendeteksi dan mengenali rambu lalu lintas dijalan (Shelke & Karde, 2012).

Gambar 2.1 merupakan contoh rambu pembatas kecepatan dalam berbagai kondisi pencahayaan.

Gambar 2.1 Citra rambu pembatas kecepatan.

Rambu pembatas kecepatan yang diambil adalah rambu pembatas kecepatan yang ada di Indonesia.

2.2.2 Pengolahan Citra

Sebuah citra dapat di representasikan sebagai sebuah 2-Dimensi fungsi f(x,y) dimana x dan y adalah ruang koordinat, dan luas f pada setiap pasangan dari (x,y) disebut intensitas atau level keabuan dari citra yang dimaksud. Pada saat (x,y) dan luas dari f terbatas, berlainan dalam jumlah banyak, dapat disebut citra sebagai citra digital. Bidang pengolahan citra digital berhubungan dengan pemrosesan citra digital menggunakan komputer (Gonzales & C, 2002).

Pengenalan pola citra merupakan salah satu bidang penelitian yang paling banyak disukai. Banyak aplikasi yang digunakan dalam mendeteksi citra salah satunya pengenalan rambu-rambu lalu lintas. Pengolahan citra digital merupakan penggunaan algoritma komputer dalam pengolahan citra pada citra digital (Al-Azawi, 2012). Operasi pengolahan citra secara garis besar dapat dibagi menjadi 3 kategori utama : kompresi citra, perbaikan dan peningkatan kualitas citra, segmentasi citra. (Nitasha et al., 2012)

1. Kompresi citra : berarti pengurangan jumlah data yang dibutuhkan untuk menghasilkan kembali sebuah citra.
2. Perbaikan dan peningkatan kualitas citra: Ketika sebuah citra dikonversi dari suatu bentuk ke bentuk yang lain, penurunan kualitas terjadi padi hasil keluaran. Peningkatan kualitas dapat dilakukan dengan aplikasi teknik perbaikan dan/atau peningkatan. Peningkatan kualitas citra meningkatkan kejernihan citra dari penglihatan manusia, penghilangan noise dan kekaburan, peningkatan *contrast, revealing details*, adalah contoh dari peningkatan kualitas citra.

3. Segmentasi Cita : proses pembagian sebuah citra ke dalam unsur-unsur dan objek-objek tertentu.

Segmentasi citra adalah sebuah cara untuk mengekstrak dan merepresentasikan informasi dari sebuah gambar ke grup pixel yang mempunyai kesamaan,membagi sebuah citra ke beberapa bagian yang bernilai atau kumpulan pixel-pixel yang berhubungan dengan sebuah aplikasi tertentu. Pixel akan dikelompokan sesuai dengan rata-rata perubahan dari intensitas setiap bagian pixel atau rata-rata perubahan kedalaman pixel sesuain dengan pixel dalam suatu permukaan seperti, bidang datar, silinder, bulatan, dll (Pandey et al., 2012; Islam & Ahmed, 2013; Ramadevi et al., 2010). Interpretasi dari isi citra merupakan salah satu tujuan dalam *computer vision* secara khusus pengolahancitra (Singh & Datar, 2013).

2.2.3 Deteksi warna

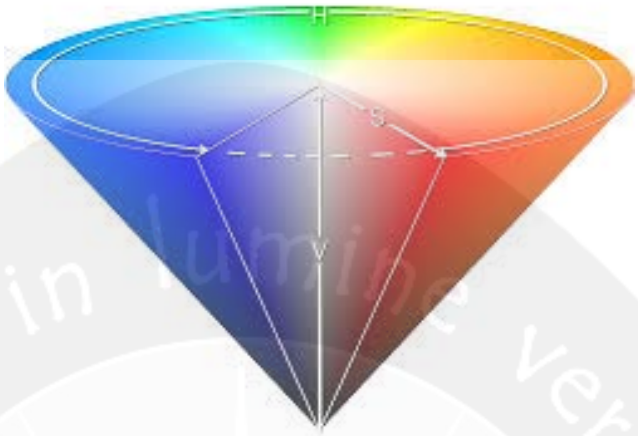
Warna sangat membantu dalam mengidentifikasi objek pada beberapa penelitian belakangan ini. Deteksi warna membantu untuk menyederhanakan masalah monokrom dengan mempertajam kontrast. Proses klasifikasi warna melibatkan ekstraksi informasi yang berguna dari ciri permukaan objek dan pencarian kecocokan yang paling baik dari sekumpulan gambar atau kelas model untuk mengimplementasikan proses pengenalan (Arivazhagan et al., 2010).

Banyak penelitian menggunakan metode deteksi warna berdasarkan pada ruang warna yang berbeda-beda seperti RGB, YcbCr, HSV (Akarlar & Yagimli, 2014; Dean & Jabir, May 2013; Wu et al., 2007). Ruang warna RGB terdiri dari 3

nilai warna yaitu *red*, *green*, dan *blue*. Sulit untuk melakukan penentuan warna pada citra RGB, sehingga banyak penelitian yang tidak menggunakan ruang warna RGB untuk pendeteksian warna. Gambar 2.2 merupakan gambaran ruang warna RGB.

Gambar 2.2 Ruang warna RGB

Banyak penelitian lebih memilih melakukan pendeteksian pada ruang warna HSV *(Hue Saturation Value)* karena dinggap lebih dapat merepresentasikan warna sama seperti konsep mata manusia dalam melihat sebuah warna (Surinwarangkoon et al., 2013; Wu et al., 2007). Gambar 2.3 merupakan gambaran ruang warna HSV.



Gambar 2.3 Ruang Warna HSV

Transformasi citra dari ruang warna RGB keruang warna HSV dapat dilakukan dengan persamaan berikut :

$$H = \begin{cases} H', if\ B \leq G \\ 360 - H', \end{cases} , \tag{2.1}$$

$$S = \frac{\mathrm{Max(R,G,B)} - \mathrm{Min(R,G,B)}}{\mathrm{Max(R,G,B)}},$$

$$V = \frac{\mathrm{Max(R,G,B)}}{255},$$

$$H' = \cos^{-1}\left\{\frac{0{,}5[(\mathrm{R}-\mathrm{G})+(\mathrm{R}-\mathrm{B})]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

Beberapa penelitian lain yang memanfaatkan informasi warna untuk pendeteksian objek dapat dilihat pada tabel 2.1 :

Tabel 2.1 Penelitian Deteksi Warna

No.	Penulis	Judul	Pembahasan
1.	(Akarlar & Yagimli, 2014)	Target Recognition With Color Components And Sobel Operator	Dalam penelitiannya memamparkan mengenai pengenalan target objek dengan membandingkan citra yang dihasilkan melalui kamera dan citra yang ada didatabase menggunakan operator sobel. Citra yang sama ditentukan presentasi kemiripan, kemudian komponen warnanya dibandingkan.
2.	(Choudari et al., 2013)	Detection Of Road Sign Using Space Variant Sensor.	Dalam penelitiannya menggunakan ruang warna HSI untuk melakukan proses segmentasi. Citra RGB dikonversi menjadi citra HSI.
3	(Kaur, 2012)	Detection of Moving Objects in Colour Based.	Dalam penelitiannya menggunakan algoritma untuk mendeteksi objek bergerak pada bidang bewarna. Metode ini

			tergantung pada intensitas cahaya dan kecerahan warna dan berlaku hanya ketika objek dan latar belakang mempunyai warna yang berbeda.
4.	(Arivazhagan et al., 2010)	Fruit Recognition using Color and Texture Features	Dalam penelitiannya menggunakan ruang warna HSV untuk mengenali jenis buah.
5	(Nirapure & Reddy, 2013)	Fast Retrieval of Images Using Filtered HSV Color Level Detection	Dalam penelitiannya memanfaatkan ruang warna HSV untuk pencarian citra pada *database*.

2.2.4 Deteksi Tepi

Deteksi tepi merupakan salah satu metode yang paling banyak digunakan dalam aplikasi *computer vision* (Almadhoun, 2013). Deteksi tepi adalah salah satu dari sekian banyak teknik segmentasi citra. Deteksi tepi sangat berguna dalam banyak hal. Tepi menciri-khaskan batasan objek dan untuk itu sangat berguna pada proses segmentasi, identifikasi dan pengenalan objek dalam bagian-bagian. Keluaran dari deteksi tepi merupakan tepi gambar yang mana nilai dari setiap pixel menggambarkan seberapa kuat hubungan pixel dalam gambar asli dengan keperluan pixel tepi. Banyak metode deteksi tepi yang telah

dikembangkan seperti *sobel, prewit* dan *robert* (Nitasha et al., 2012). Gambar 2.4, 3.5, dan 2.6 merupakan contoh operator *sobel, prewit* dan *robert.*

+1	+2	+1
0	0	0
-1	-2	-1

-1	0	+1
-2	0	+2
-1	0	+1

Gambar 2.4 Operator Sobel

+1	+1	+1
0	0	0
-1	-1	-1

-1	0	+1
-1	0	+1
-1	0	+1

Gambar 2.5 Operator Prewit

1	0
0	-1

0	1
-1	0

Gambar 2.6 Operator Robert

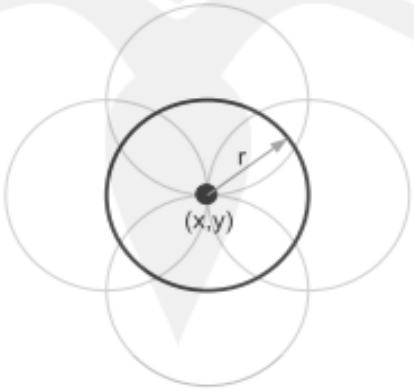
Deteksi tepi pada citra adalah sebuah hal yang penting dan mendasar dalam pengolahan citra dan komputer vision. Deteksi tepi merupakan salah satu operasi yang paling sering digunakan pada pengolahan citra dan terutama pengenalan pola. Alasannya karena tepi-tepi merupakan bagian luar yang sangat penting dari sebuah objek. Sebuah tepi adalah batasan antara sebuah objek dan latar belakang dan mengindikasikan batas antara satu objek dan objek lain. Deteksi tepi pada citra keabuan merupakan bidang yang sering dipakai dalam penelitian sedangkan deteksi tepi pada citra bewarna kurang citra begitu mendapat perhatian. Perbedaan mendasar dari citra bewarna dan citra keabuan yaitu pada

citra bewarna sebuah warna (terdiri dari 3 komponen) ditandakan sebagai sebuah vektor, sedangkan sebuah skala citra keabuaan ditandakan sebagai sebuah pixel (Kamboj et al., 2012).

2.2.5 Circle Hough Transformation

Hough transform merupakan sebuah teknik ekstraksi fitur yang digunakan dalam pengolahan citra. Transformasi klasik mengidentifikasi gari-garis dalam sebuah citra, kemudian akhirnya digunakan untuk identifikasi posisi pada bentuk-bentuk yang berubah-ubah. Ekstraksi fitur dari citra digital, sangat berguna untuk mencari garis yang lurus, lingkaran, dan elips (Barrile et al., 2012). Hough transform berdasarkan pada point-point fitur yang diekstrak dari citra asli, dan biasanya tepi digunakan sebagai *point-point* fiturnya. Berbagai macam metode deteksi tepi digunakan untuk aplikasi-aplikasi yang berbeda (Jain & Jain, 2012).

Metode-metode yang terkenal untuk mendeteksi bentuk lingkaran ataupun bentuk elips adalah CHT (*Circle Hough Transform)* dan EHT (*Elliptical Hough Transform)* (Smerka & Duleba, 2008). Gambar 2.7 merupakan metode CHT.



Gambar 2.7 Circle Hough Transform.

Hough transfrom dan beberapa versi modifikasi telah dikenali sebagai teknik yang handal untuk deteksi kurva. *Circle hough transform* digunakan untuk mendeteksi lingkaran berdasarkan pada beberapa dasar *transformasi hough* untuk segmentasi garis.

Sebuah pola lingkaran berdasarkan pada persamaan :

$$(x-a)^2 + (y-b)^2 = r^2. \quad (2.2)$$

Dimana *a* dan *b* adalah koordinat pusat dalam arah *x* dan *y* sesuai dengan *r* yang adalah radius dari lingkaran.

$$x = a + r\cos\theta. \quad (2.3)$$

$$y = b + r\sin\theta \quad (2.4)$$

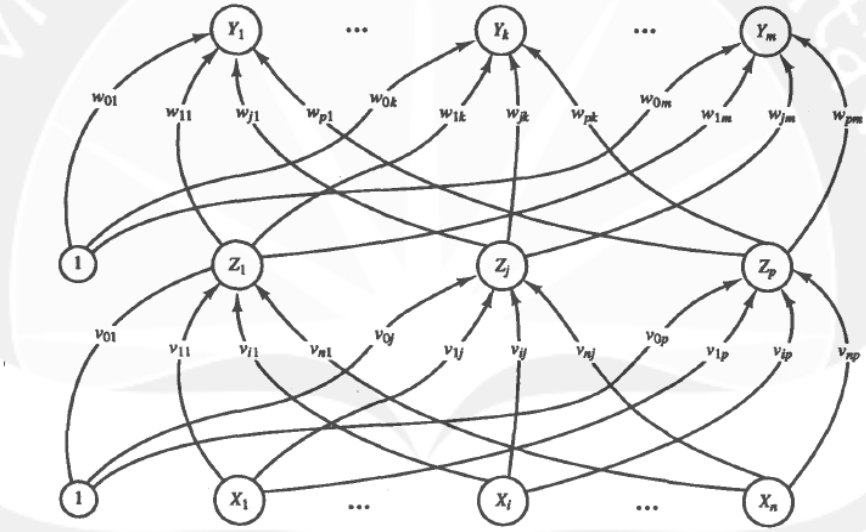
Sebuah lingkaran didefinisikan oleh 3 parameter : koordinat pusat (a,b) dan radius/jari-jari (r), ruang hough adalah ruang 3 dimensi, dengan Z-axis sebagai lingkaran (Jain & Jain, 2012).

2.2.6 Jaringan Saraf Tiruan Backpropagation

Jaringan saraf tiruan sering digunakan dalam aplikasi pada bidang *computer vision* dikarenakan kemudahannya dalam beradaptasi dan proses pembelajaran. Input jaringan saraf tiruan merupakan vektor nilai yang diambil dari citra. Vektor ini yang kemudian digunakan untuk mencocokkan sebuah gambar dengan vektor yang lain yang tersimpan dalam sebuah databse (Al-Azawi, 2012).

Backpropagation merupakan jaringan saraf tiruan yang terdiri dari beberapa *layer*. Gambar 2.8 merupakan gambaran proses dari jaringan saraf

tiruan backpropagation dimana x merupakan input, z *hidden layer* dan y sebagai *output*. Terdapat juga yang namanya bias yang merupakan suatu nilai konstant yaitu 1 dan bobot dari masing-masing neuron termasuk bobot bias yang di inisialisasi dengan v dan w. Tahap pembelajaran dari JST *backpropagation* terdiri dari 3 langkah yaitu; tahap perhitungan maju dari *input*, tahap perhitungan balik (*backpropagation*) dari kumpulan *error* yang dihasilkan, dan tahap pembaharuan nilai bobot (Fausett, 1993).



Gambar 2.8 Arsitektur Jaringan Saraf Tiruan (Fausett, 1993)

Setelah proses pelatihan, aplikasi jaringan hanya terdiri dari fase komputasi *feedforward*. Walaupun proses pelatihan tergolong lambat, sebuah jaringan yang dilatih dapat menghasilkan keluaran secara secpat. Banyak variasi *backpropagation* telah dikembangkan untuk meningkatkan kecepatan dari proses pelatihan (Fausett, 1993). Pada dasarnya algoritma standard *backpropagtion* adalah sebagai berikut :

Algoritma standard *backpropagation*:

Langkah ke-1

Inisialisasi Bobot.

(set secara random nilai bobot yang kecil range [0,1]).

Selama kondisi stop belum terpenuhi, lakukan langkah ke-2 sampai dengan langkah ke-9.

***Langkah ke-* 2**.

Untuk setiap pelatihan, lakukan langkah 3-8.

Feedforward:

***Langkah ke-3*.**

Setiap input unit *(Xi, i* = 1, ... , *n)* menerima input signal *Xi* and mengirimkan signal ini kesemua unit ke-*layer* diatasnya (*the hidden units*).

Langkah ke- 4

Setiap hidden unit *(Zj,,j* = 1, ... ,*p)* memproes nilai bobotnya,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} (x_i v_{ij}) \quad (2.5)$$

Menggunakan fungsi aktifasi untuk menghasilkan output signalnya,

$$Zj = f(z\_in_j) \quad (2.6)$$

Dan mengirim signal ini kesemua *neuron* di-*layer* atasnya.

Langkah ke-5

Setiap output unit *(Yk , k* = 1, ... , m) memproses nilai bobotnya,

$$y\_in_k = w_{0k} + \sum_{j=1}^{n} (z_j w_{jk}) \quad (2.7)$$

Menggunakan fungsi aktifitas untuk mengasillnya output signalnya

$$yk = F(z\_in_j) \quad (2.8)$$

Backpropagation error:

Langkah ke-6.

Setap output unit *(Yk , k* = 1, ... ,*m)* menerima pola target sesuai dengan masukan pola pembelajaran, kemudian menghitung errornya

$$\delta k = (tk - yk) f'(y_{in_k}) \quad (2.9)$$

Hitung selisih bobotnya (digunakan untuk memperbaharui *Wjk*),

$$\Delta w_{jk} = a8kzj \quad (2.10)$$

Hitung koreksi bias (digunakan untuk memperbaharui *WOk* kemudian),

$$\Delta Wk = a8k \quad (2.11)$$

Dan mengirim *8k* ke semua unit dibawahnya.

Langkah ke-7

Setiap hidden unit *(Zjo* j = 1, ... ,*p)* memproses selisih input delta inputs (dari unit di-*layer* diatasnya),

$$\delta\_in_j = \sum_{k=1}^{m} (\delta_k w_{jk}) \quad (2.12)$$

Menggunakan fungsi aktifasi untuk menghitungkan nilai errornya

$$\delta j = \delta\_in_j f'(y_{in_j}) \quad (2.13)$$

Hitung selisih bobotnya (digunakan untuk memperbaharui *vij*)

$$\Delta v_{ij} = \text{a}\delta \quad _j x_i \quad (2.14)$$

Hitung selisih bobotnya (digunakan untuk memperbaharui *VOj*),

$$\Delta v_{0j} = \text{a}\delta \quad _j \quad (2.15)$$

Memperbaharui bobot dan bias

Langkah ke-8

Setiap output unit *(Yk, k* = I, , *m)* memperbaharui bias dan bobot (j = 0, , *p):*

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \quad (2.16)$$

Setiap hidden unit *(Zj,* j = 1, *,p)*memperbaharui bias dan bobot (i = 0, , *n):*

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} \quad (2.17)$$

Langkah ke-9 Test kondisi stop.

2.2.7 OpenCV

Dikutip dari situs resminya, OpenCV *(Open Source Computer Vision Library)* adalah sebuah *library* perangkat lunak *open source computer vision* dan *machine learning.* OpenCV dirilis dibawah lisensi BSD oleh karena itu disediakan gratis baik untuk penggunaan akademis dan komersial. OpenCV merupakan *library* yang sering digunakan untuk keperluan akademis baik tesis maupun disertasi.

OpenCV memiliki antarmuka C++, C, Python dan Java dan dapat digunakan pada operating system Windows, Linux, Mac OS, IOS dan Android. OpenCV dirancang untuk hal efisiensi komputasi dan berfokus pada aplikasi *real-time*. OpenCV memiliki lebih dari 47.000 pengguna yang tergabung dalam komunitas dan perkiraan jumlah download melebihi 7.000.000.

OpenCV dibangun untuk menyediakan infrastruktur umum untuk aplikasi *computer vision* dan untuk mempercepat penggunaan *machine perception* dalam produk-produk komersial. Didalam OpenCV terdapat lebih dari 2.500 algoritma yang mencakup sekumpulan algoritma *computer vision* dan *machine learning* baik klasik maupun lanjutan. Algoritma-algoritma itu dapat digunakan untuk mendeteksi dan mengenali wajah, identifikasi objek, ekstrasi objek model 3D, dan lain sebagainya (opencv.org).

2.2.8 IOS

IOS merupakan platform yang memiliki berjuta-juta aplikasi *mobile* dan terus bertambah dari hari kehari. IOS SDK dan Xcode IDE mempermudah para pengembang untuk membuat aplikasi *mobile*. IOS adalah sistem operasi *mobile* tercanggih didunia dilihat berdasarkan pada apa yang dapat dilakukan pengguna dengan sebuah perangkat mobile. Berasal dari teknologi inti OSX, pengalaman luar biasa dari pengguna IOS telah memperkecil pengambilan keuntungan dari iphone, ipad, dan ipod. Teknologi yang digunakan dipisahkan antara iOS dan OS X termasuk kernel OS X, BSD *sockets* untuk jaringan dan kompiler Objective-C dan C/C++ untuk kinerja yang asli.

IOS dirancang untuk mengutamakan privasi pengguna. Hal ini ditunjukan dengan keseriusan tim dalam menangani masalah keamanan seperti serangan *malware* dan *virus* yang telah berkembang pada sistem operasi perangkat *mobile*. IOS juga menjangkau segala kalangan pengguna bahkan menyertakan berbagai macam fitur yang dapat membantu pengguna dengan keterbatasan fisik tertentu (developer.apple.com).