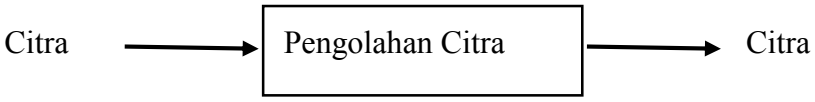
BAB 2
LANDASAN TEORI

2.1. Citra Digital

Secara harfiah, citra (*image*) adalah gambar pada bidang dwimatra (dua dimensi) [3]. Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek kemudian objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya. Sehingga bayangan objek yang disebut citra tersebut terekam.

2.2. Pengolahan Citra

Pengolahan citra (*image processing*) adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik [3]. Pengolahan citra ini sangat diperlukan karena walaupun citra sangat kaya dengan informasi, namun seringkali citra mengalami penurunan mutu (*degradasi*), misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur (*blurring*), dan sebagainya. Tentu saja citra semacam ini menjadi lebih sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang.

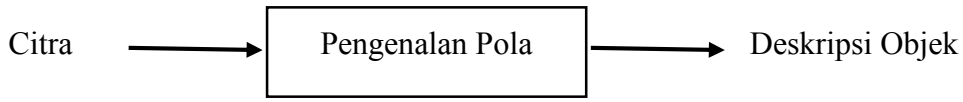


Gambar 2.1 Pengolahan citra

2.3. Pengenalan Pola

Pengenalan pola merupakan kegiatan mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer) [3]. Tujuan pengelompokan adalah untuk mengenali suatu objek di dalam citra.

Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek di sekitarnya sehingga mampu membedakan suatu objek dengan objek lainnya. Kemampuan sistem visual manusia inilah yang dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut, dan memberikan keluaran berupa deskripsi objek di dalam citra.



Gambar 2.2 Pengolahan citra

2.4. Computer Vision

Computer vision merupakan proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akuisisi citra, pengolahan citra, klasifikasi, pengenalan (*recognition*), dan membuat keputusan [3]. Pada hakikatnya, *computer vision* mencoba meniru cara kerja sistem visual manusia (*human vision*). *Human vision* sesungguhnya sangat kompleks. Manusia melihat objek dengan indera penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk pengambilan keputusan (misalnya menghindar kalau melihat mobil melaju di depan).

Pengolahan citra merupakan proses awal (*preprocessing*) pada computer vision, sedangkan pengenalan pola merupakan proses untuk menginterpretasi citra. Teknik-teknik di dalam pengenalan pola memainkan peranan penting dalam *computer vision* untuk mengenali objek.

2.5. Model Warna Pada Citra

Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek [3]. Setiap warna mempunyai panjang gelombang (l) yang berbeda. Warna merah mempunyai

panjang gelombang paling tinggi, sedangkan warna ungu (violet) mempunyai panjang gelombang paling rendah.

Warna-warna yang diterima oleh mata (sistem visual manusia) merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. Penelitian memperlihatkan bahwa kombinasi warna yang memberikan rentang warna yang paling lebar adalah *red* (R), *green* (G), dan *blue* (B).

Dalam pembentukan citra digital model warna yang umum digunakan antara lain sebagai berikut :

2.5.1. Citra RGB

Citra RGB, yang biasa disebut juga citra "true color", disimpan dalam citra berukuran $(m \times n) \times 3$ yang mendefinisikan warna merah (*red*), hijau (*green*), dan warna biru (*blue*) untuk setiap pikselnya. Warna pada tiap piksel ditentukan berdasarkan kombinasi dari warna *red*, *green* dan *blue* (RGB). RGB merupakan citra 24 bit dengan komponen merah, hijau, biru yang masing-masing umumnya bernilai 8 bit sehingga intensitas kecerahan warna sampai 256 level dan kombinasi warnanya kurang lebih sekitar 16 juta warna sehingga disebut "true color".

2.5.2. Citra Keabuan

Citra dengan derajat keabuan berbeda dengan citra RGB, citra ini didefinisikan oleh satu nilai derajat warna. Umumnya bernilai 8 bit sehingga intensitas kecerahan warna sampai 256 level dan kombinasi warnanya 256 varian. Tingkat kecerahan paling rendah yaitu 0 untuk warna hitam dan putih bernilai 255.

Untuk mengkonfersikan citra yang memiliki warna RGB ke derajat keabuan bisa menggunakan :

$$Gray = \frac{R+G+B}{3} \qquad (2.1)$$

atau

$$Gray = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad (2.2)$$

2.5.3. Citra Biner

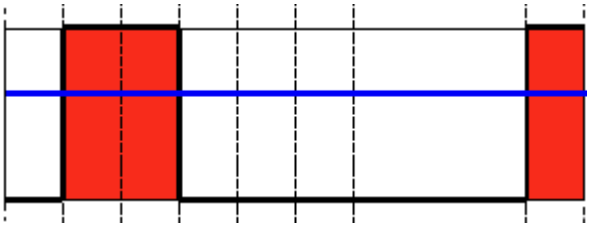
Citra biner merupakan citra yang didefinisikan oleh dua level warna yakni hitam dan putih atau 0 (*false*) dan 1 (*true*). Untuk mengkonfersi citra yang memiliki warna keabuan ke warna biner bisa menggunakan proses *binary thresholding*. Secara sederhana, *binary thresholding* digunakan untuk menghilangkan informasi citra yang memiliki derajat keabuan dibawah *threshold* yang ditentukan. *Binary thresholding* dibagi menjadi dua tipe, yaitu :

a. Threshold Binary

Operasi ini dapat diekspesikan sebagai berikut :

$$dst(x,y) = \begin{cases} 1, & src(x,y) > T \\ 0, & src(x,y) \leq T \end{cases} \qquad (2.3)$$

Sehingga jika nilai pixel $src(x,y)$ lebih besar dari nilai threshold, pixel baru diinisialisasi dengan nilai 1. Sedangkan untuk nilai lainnya diinisialisasi dengan nilai 0.



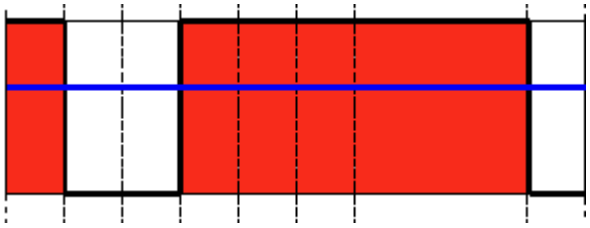
Gambar 2.3 Threshold binary

b. Threshold Binary, Inverted

Operasi ini dapat diekspesikan sebagai berikut :

$$dst(x,y) = \begin{cases} 0, & src(x,y) > T \\ 1, & src(x,y) \leq T \end{cases} \qquad (2.4)$$

Sehingga jika nilai pixel $src(x,y)$ lebih besar dari nilai threshold, pixel baru diinisialisasi dengan nilai 0. Sedangkan untuk nilai lainnya diinisialisasi dengan nilai 1.



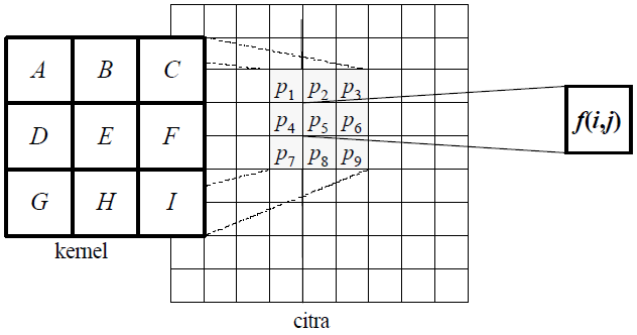
Gambar 2.4 Threshold binary inverted

2.6. Neighborhood Filtering (Convolution)

Convolution merupakan proses dimana citra dimanipulasi dengan menggunakan *eksternal mask* atau *kernel* untuk menghasilkan citra yang baru berdasarkan *neighborhood operator* atau *local operator*. *Convolution* digunakan untuk *local tone adjustment, bluring,* mempertajam detail, menonjolkan tepi atau menghilangkan noise [4]. Dalam citra yang memiliki fungsi dua dimensi, operasi *convolution* didefinisikan sebagai berikut :

$$f(i,j) = \sum_{k,l} p(i+k, j+l) \cdot h(k,l) \qquad (2.5)$$

Fungsi penapis h disebut juga *convolution filter, convolution mask, convolution kernel.* Ilustrasi *convolution* ditunjukan pada Gambar 2.5.



Gambar 2.5 Ilustrasi convolution

Sehingga *convolution* untuk f(i,j) dapat didefinisikan sebagai berikut :

$$\mathrm{f}(i,j) = A \cdot p_1 + B \cdot p_2 + C \cdot p_3 + D \cdot p_4 + E \cdot p_5 + F \cdot p_6 + G \cdot p_7 + H \cdot p_8 + I \cdot p_9$$

Operasi *convolution* dilakukan dengan menggeser *kernel convolution pixel* per *pixel*. Hasil *convolution* disimpan di dalam matriks yang baru.

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

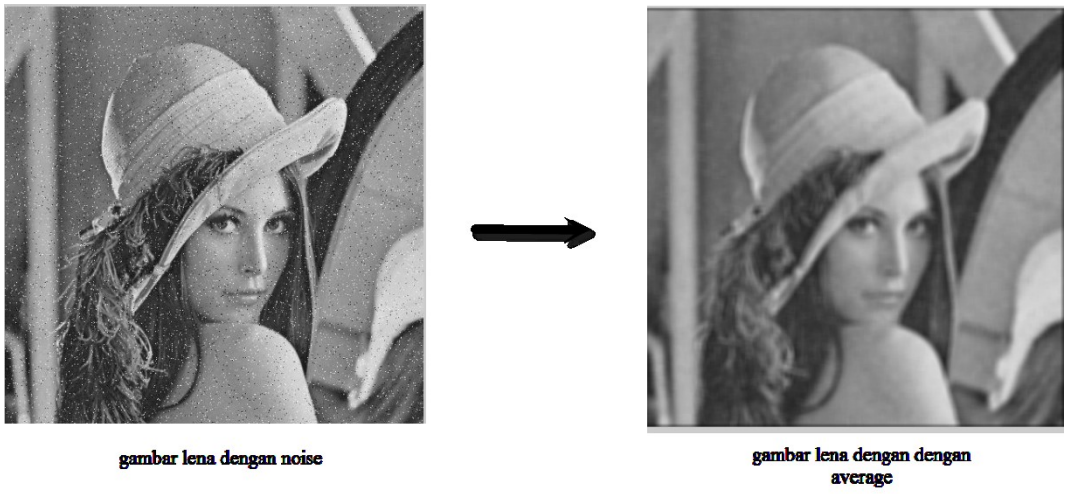
Gambar 2.6 Ilustrasi nilai hasil operasi convolusi

2.7. Penghalusan Citra (*Smoothing*)

Penghalusan citra (*Smoothing*) merupakan prosses untuk meminimalisir *noise* atau mengaburkan citra. Dalam penghalusan citra operasi sederhana yang dapat digunakan adalah *average* atau rata-rata. Citra yang akan diperhalus di-*convolution* dengan kernel h (m x n). Dimana kernel h dapat didefinisikan, sebagai berikut :

$$h = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \cdot \frac{1}{nm}$$

Hasil dari penghalusan dengan rata-rata ini dapat di lihat pada Gambar 2.7.

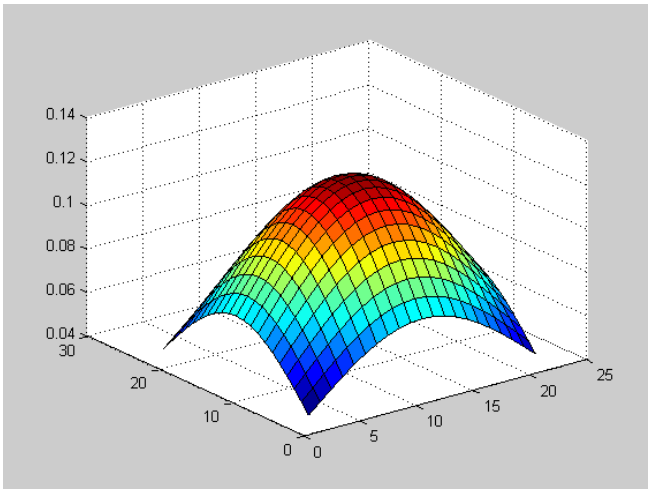


Gambar 2.7 Smoothing lena dengan average

Kernel lain yang dapat digunakan adalah *kernel gaussian*. *Kernel gaussian* dinilai baik dalam penghalusan citra karena mementingkan *pixel* ketentanggan terdekat. Fungsi *gaussian* dapat didefinisikan sebagai berikut :

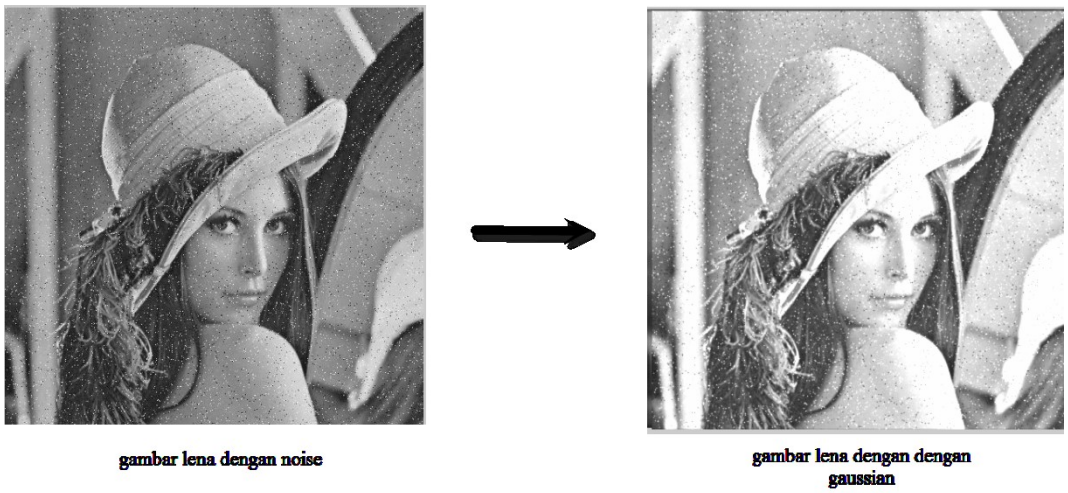
$$g(x,y) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (2.6)$$

Dimana $\frac{1}{\sqrt{2\pi}\sigma}$ merupakan koefisien normalisasi pada fungsi *gaussian*. Gambar 2.8 Menunjukan grafik dari fungsi *gaussian*.



Gambar 2.8 Grafik kernel gaussian

Hasil penghalusan dengan kernel gaussian ini dapat dilihat pada Gambar 2.9.



Gambar 2.9 Smoothing lena dengan kernel gaussian

2.8. Deteksi Tepi (*Canny*)

Canny merupakan algoritma pendeteksi tepi yang banyak digunakan dan dianggap mempunyai kemampuan deteksi tepi yang baik karena bisa mencari dan menandai semua tepi selain itu mementingkan tepi yang dianggap nyata dan tepi yang terdeteksi hanya satu poin sehingga tepi lebih terlihat bersih.

Tahapan dari deteksi tepi canny adalah sebagai berikut :

a. Image Smoothing

Pada tahap pertama citra diterapkan *smoothing* dengan kernel Gaussian.

$$S = I * g(x,y) \qquad (2.7)$$

Dimana :

$$g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

b. Hitung turunan dari citra terfilter

Pada tahap ini matrix citra dihitung turunannya menggunakan *Gaussian* turunan pertama.

$$\nabla S = \nabla(g * I) = (\nabla g) * I$$

$$\nabla g = \begin{bmatrix} \frac{\partial}{\partial x} g \\ \frac{\partial}{\partial y} g \end{bmatrix}$$

$$\frac{\partial}{\partial x} g = \frac{\partial}{\partial x} e^{-\frac{x^2+y^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial y} g = \frac{\partial}{\partial y} e^{-\frac{x^2+y^2}{2\sigma^2}} = -\frac{y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Sehingga :

$$\nabla S = \begin{bmatrix} g'_x \\ g'_y \end{bmatrix} * I = \begin{bmatrix} g'_x * I \\ g'_y * I \end{bmatrix} \qquad (2.8)$$

c. Hitung *gradient magnitude* dan orientasi *gradient*

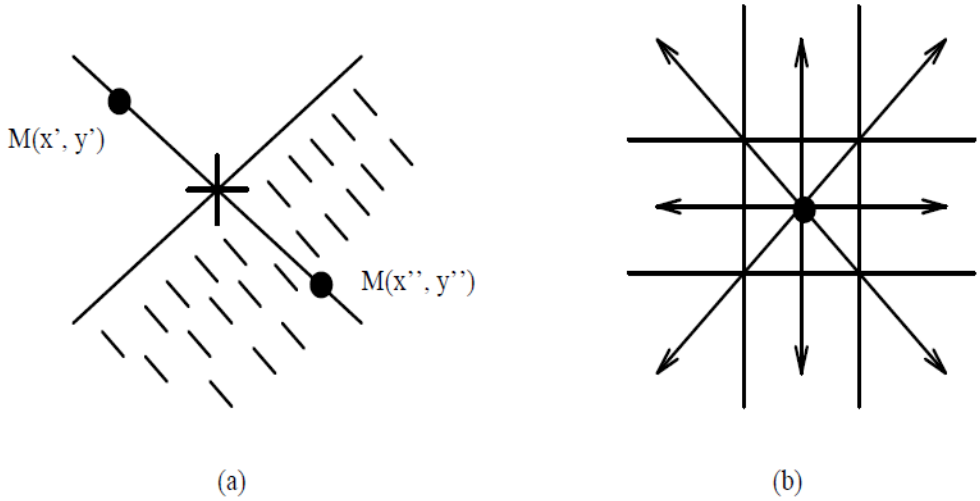
Setelah S_x dan S_y didapatkan, tahapan selanjutnya adalah mencari magnitude dan direction atau orientasi. Dengan cara :

$$magnitude = \sqrt{{S_x}^2 + {S_y}^2} \qquad (2.9)$$

$$direction = \theta = tan^{-1}\frac{S_x}{S_y} \qquad (2.10)$$

d. Terapkan *non-maximum suppression*

Non-maximal supression bertujuan membuang potensi gradien di suatu piksel dari kandidat tepi jika piksel tersebut bukan merupakan maksimal lokal pada arah tepi di posisi piksel tersebut (di sinilah arah gradien diperlukan).



Gambar 2.10 (a) Non-maximum suppression (b) Kemungkinan direction tepi

$$M(x,y) = \begin{cases} \Delta S(x,y), & \Delta S(x,y) > \Delta S(x',y') \\ & dan\ S(x,y) > \Delta S(x'',y'') \\ 0, & lainnya \end{cases} \qquad (2.11)$$

e. Terapkan *hysteresis threshold*

Hysteresis thresholding adalah klasifikasi dengan dua buah nilai *High-Threshold* dan *Low-Threshold*. Suatu piksel disahkan sebagai piksel tepi jika nilainya lebih besar atau sama dengan *High-Threshold* atau jika piksel tersebut

memiliki intensitas kekuatan tepi yang lebih besar dari *Low-Threshold* dan terhubung dengan piksel yang nilainya lebih besar dari *High-Threshold*.

2.9. Corner Detection (*Harris*)

Harris Corner Detector diajukan oleh C. Harris dan M. Stephen pada Tahun 1988. *Harris* didasarkan pada *moment* matriks kedua yang digunakan untuk ekstraksi ciri. Dalam deteksi sudut menggunakan Harris, *konvolution* dengan fungsi *Gaussian* merupakan langkah paling penting karena akan menentukan keseluruhan proses dan hasil akhir.

Tahapan dari deteksi corner harris adalah sebagai berikut :

1. Hitung turunan citra pada arah x dan y

   Pada tahap ini matrix citra dihitung turunannya menggunakan Gaussian turunan pertama.

   $$Ix = g'x * I \, , Iy = g'y * I \qquad (2.12)$$

2. Hitung kembali varian dari turunan citra di setiap *pixel*

   $$I_x{}^2 = I_x \cdot I_x \, , \; I_y{}^2 = I_y \cdot I_y \, , \; I_{xy} = I_x \cdot I_y \qquad (2.13)$$

3. Hitung convolusi dengan gaussian dan varian turunan citra di setiap *pixel*

   $$S_x{}^2 = g' * I_x{}^2 \, , \quad S_y{}^2 = g' * I_y{}^2 \, , \; S_{xy} = g' * I_{xy} \qquad (2.14)$$

4. Definisiakan setiap *pixel (x,y)* kedalam matrix :

   $$H(x,y) = \begin{bmatrix} S_x{}^2(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_y{}^2(x,y) \end{bmatrix} \qquad (2.15)$$

5. Cari respon dari setiap *pixel*

   $$R = det(H) - k(trace(H))^2 \qquad (2.16)$$

6. *Threshold* nilai R dan terapkan *nonmax suppression.*

2.10. *Hu Invarian Moments*

Hu Invarian Moment pertama kali dipublikasikan oleh Hu pada tahun 1961 dan *Hu Invariant Moments* di reformula oleh Li [5]. Nilai-nilai yang dihasilkan dari Hu Invariant Moments ini terdiri dari tujuh nilai yang menidentifikasikan ciri dari sebuah objek citra digital [6]. Nilai-nilai tersebut bersifat independen terhadap translasi, rotasi dan penskalaan [6].

Momen yang mentransformasikan fungsi citra f(i,j) pada sistem diskrit dinyatakan dengan persamaan :

$$m_{pq} = \sum_{x=0}^{H-1} \sum_{y=0}^{W-1} x^p y^q f(x,y) \quad (2.17)$$

Dimana H dan W masing-masing merupakan tinggi dan lebar citra dan p = 0,1,2,… dan q = 0,1,2,… adalah integer. Selanjutnya, momen pusat untuk suatu citra dinyatakan dengan :

$$\mu_{pq} = \sum_{x=0}^{H-1} \sum_{y=0}^{W-1} (x - \bar{x})^p (y - \bar{y})^q f(x,y) \quad (2.18)$$

Dimana :

$$\bar{x} = \frac{m_{10}}{m_{00}} \; dan \; \bar{y} = \frac{m_{01}}{m_{00}} \quad (2.19)$$

Kemudian *normalized central moments*, didefinisikan sebagai berikut :

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{pq}^{\gamma}} \; , \; \gamma = \frac{p+q}{2} + 1 \quad (2.20)$$

Sehingga ketujuh nilai *invariant moments* dapat diturunkan dari moment kedua dan ketiga sebagai berikut.

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^{\;4}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} + \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - (3\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[(3\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\phi_6 = (\eta_{20} - 3\eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} - \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (2.21)$$

2.11. *Neural Network*

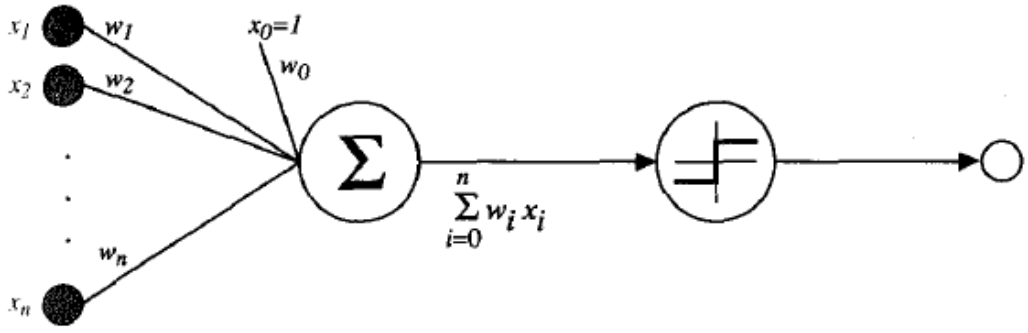
Konsep *Neural Network* sebagian besar terinspirasi oleh pengamatan sistem pembelajaran biologi yang dibangun oleh jaring yang sangat kompleks *neuron* yang saling berhubungan. *Neuron* dalam *Neural Network* dianalogikan sebagai unit yang memungkinkan saling terhubung secara sederhana dengan unit lainnya. Suatu unit membutuhkan beberapa masukan nilai desimal (yang memungkinkan dari unit lain)

dan keluaran dari unit adalah nilai tunggal desimal (yang mungkinkan menjadi masukan unit lain) [7]. Kemampuan pemprosessan disimpan di dalam kekuatan koneksi interunit atau bobot (*weights*), kemampuan tersebut diperoleh dari proses "adaptasi terhadap" atau "belajar dari" suatu set pola pelatihan [1].

Neural Network tercipta sebagai suatu generalisasi model matematis dari pemahaman manusia (*human cognition*) yang didasarkan atas asumsi sebagai berikut :

1. Pemrosesan informasi terjadi pada elemen sederhana yang disebut neuron
2. Sinyal mengalir diantara unit neuron melalui suatu penghubung
3. Setiap sambungan penghubung memiliki bobot yang bersesuaian. Bobot ini akan digunakan untuk menggandakan / mengalikan sinyal yang dikirim melaluinya.
4. Setiap sel syaraf akan menerapkan fungsi aktivasi terhadap sinyal hasil penjumlahan berbobot yang masuk kepadanya untuk menentukan sinyal keluarannya.

Model struktur neuron pada neural network dijelaskan pada Gambar 2.11.



Gambar 2.11 Model Unit Pengolah Neural Network

2.12. Backpropagation

Backpropagation merupakan algoritma pembelajaran *neural network* yang cukup populer dan banyak digunakan. Secara umum pada arsitektur *Neural Network Backpropagation*, *neuron-neuron* akan dikumpulkan dalam lapisan-

lapisan yang disebut dengan *layer*. Neuron-neuron pada satu *layer* akan dihubungkan. Bila *neuron-neuron* pada suatu *layer* (misal *hidden layer*) akan dihubungkan dengan neuron-neuron pada *layer* lain (misal *output layer*) maka setiap neuron pada *layer* tersebut (*hidden layer*) juga harus dihubungkan dengan setiap neuron pada *layer* lainnya (*output layer*), ini sebut dengan terhubung penuh (*full connected*). Informasi yang diberikan pada *Neural Network* ini akan merambat dari lapisan ke lapisan, mulai dari lapisan masukan sampai ke lapisan keluaran melalui *hidden layer*. Faktor terpenting dalam menentukan perlakuan suatu *neuron* adalah fungsi aktivasi dan pola bobotnya [7]. Umumnya *neuron-neuron* yang terletak pada *layer* yang sama akan memiliki keadaan yang sama sehingga pada setiap *layer* yang sama neuron-neuron memiliki fungsi aktivasi yang sama.

Backpropagation memiliki dua konsep dasar yakni feed foward dan back propagation dari error-nya. Algoritma neural network backpropagation adalah sebagai berikut :

1. Menentukan Arsitektur Neural Network Struktur backpropagation terdiri dari tiga tipe lapisan yang terhubung penuh (full connected), yaitu: input layer, hidden layer dan output layer, dimana setiap layer di tambah satu bias.
2. Inisialisasi iterasi n = 1
3. Inisialisasi bobot $w(i,j)$ dengan nilai random.
4. Masukan pola input dan hitung respons jaringan neural network. Untuk setiap neuron mengunakan rumus :

   $$output = f\left(\sum_i \ i_i w_i\right) \qquad (2.22)$$

   Dimana $f(x)$ adalah fungsi sigmoid :

   $$f(x) = \frac{1}{(1+ e^{-x})} \qquad (2.23)$$

5. Hitung nilai error output menggunakan Root Mean Square Error :

   $$rms = \sqrt{\frac{1}{n}\sum_i^n (target_i - output_i)^2} \qquad (2.24)$$

6. Jika Error lebih kecil dari target error atau jumlah iterasi lebih besar sama dengan maksumum iterasi stop proses learning.
7. Hitung gradient ouput layer :

   $$\delta = \ (target - ouput) \ . f'(output) \qquad (2.25)$$

   Dimana :

$f'(output)$ = turunan fungsi sigmoid.

$$f'(x) = f(x)\ (1 - f(x))$$

$$f'(x) = \left(\frac{1}{(1 + e^{-x})}\right)\left(1 - \left(\frac{1}{(1 + e^{-x})}\right)\right)$$

8. Hitung gradient hidden layer :

$$\delta = \left(\sum_i (w_i * \delta_{n_i})\right).f'(output) \qquad (2.26)$$

Dimana :

δ_{n_i} = keluaran dari neuron selanjutnya

$f'(output)$ = turunan fungsi sigmoid.

$$f'(x) = \left(\frac{1}{(1 + e^{-x})}\right)\left(1 - \left(\frac{1}{(1 + e^{-x})}\right)\right)$$

9. Update bobot $w(i,j)$ dengan rumus :

$$\Delta w_{baru} = \beta\,.\delta\,.output + \alpha.\Delta w_{lama} \qquad (2.27)$$

$$\Delta w_{lama} = \Delta w_{baru}$$

$$w = w + \Delta w_{baru}$$

Dimana :

β = laju pembelajaran (learning rate)

α = parameter momentum

10. Naikan harga n, kemudian kembali ke langkah 4 hingga langkah 6 terpenuhi.

2.13. C / C++

Tahun 1969, laboratorium Bell AT&T di Muray Hill, New Jersey menggunakan bahasa *assembly* ini untuk mengembangkan sistem operasi UNIX. Maksudnya adalah untuk membuat sistem operasi yang dapat bersifat *'programmer-friendly'*. Setelah UNIX berjalan, Ken Thomson, seorang pengembang sistem di laboratorium tersebut mengembangkan bahasa baru dengan nama bahasa B. Huruf B ini diambil dari BCPL. Bahasa B ini kemudian digunakan untuk menulis ulang atau merevisi sistem operasi UNIX. Oleh karena bahasa B ini masih bersifat interpret dan lambat, maka pada tahun 1971, sistem operasi UNIX kemudian ditulis ulang dengan menggunakan bahasa C, yaitu bahasa pemrograman

yang dikembangkan oleh Dennis Ritchie, seorang pengembang sistem di laboratorium yang sama.

Sampai sekarang bahasa C masih digunakan untuk melakukan pengembangan-pengembangan program dan sistem-sistem operasi, diantaranya sistem operasi Windows. Alasan itulah yang menjadikan bahasa C sangat populer di dunia pemrograman khususnya untuk industri perangkat lunak. Namun sayangnya basa C merupakan bahasa yang masih tergolong susah untuk dipelajari karena masih bersifat prosedural murni. Untuk membentur satu objek, kita harus melakukan banyak sekali penulisan kode. Hal ini tentu dapat dikatakan sebagai sebuah kelemahan. Untuk mengatasi masalah ini, pada tahun 1983, seorang doktor bernama Bjane Stroustrup yang juga bekerja di laboratorium yang sama menciptakan bahasa baru yaitu bahasa C++ yang meripakan *hibryd* dari bahasa C. Bahasa C++ didasarkan atas bahasa C sehingga kita dapat melakukan kompilasi program-program yang ditulis dalam bahasa C dengan menggunakan kompiler C++. Keistimewaan dari bahasa C++ adalah karena bahasa ini mendukung pemrograman berarah objek atau lebih sering dikenal dengan istilah *Object Oriented Programming* (OOP).

2.14. K-Fold Cross Validation

Pada metode k-fold cross validation, data dibagi menjadi k subset (fold) yang saling bebas secara acak, yaitu S1, S2....Sk, yang diasumsikan memiliki ukuran yang sama. Pelatihan dan pengujian dilakukan sebanyak k kali. Pada iterasi ke-i subset S1 diperlakukan sebagai data pengujian dan subset lainnya sebagai data pelatihan. Tingkat akurasi dihitung dengan membagi jumlah keseluruhan klasifikasi yang benar dengan jumlah semua *instance* pada data awal [8].

2.15. Confusion Matrix

Sebuah *Confusion Matrix* (Kohavi and Provost, 1998) berisi informasi tentang klasifikasi atual dan hasil perkiraan yang dilakukan oleh sistem klasifikasi. Kinerja sistem dievaluasi menggunakan tabel matrix. Tabel ini diperlukan untuk menentukan kinerja suatu sistem klasifikasi.