BAB 2
TINJAUAN PUSTAKA

2.1 Gambar

Gambar adalah tiruan barang yang meliputi orang, tumbuhan binatang, alam,benda dan sebagainya. Yang dapat dibuat dengan coretan pensil ataupun alat lain dengan media kertas dan sebagainya[13].

2.1.1 Citra

Citra adalah gambar dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinu menjadi gambar diskrit melalui proses sampling[13].

2.1.2 Citra Digital

Citra digital adalah gambar dua dimensi yang bisa ditampilkan pada layar komputer sebagai himpunan/diskrit nilai digital yang disebut pixel/picture elements. Dalam tinjauan matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi[13].

2.2 Pattern Recognition

Pattern Recognition (PR) atau Pengenalan Pola, merupakan salah satu cabang AI *(artificial intelligence)*[14], yang mempelajari pembuatan sebuah sistem untuk dapat mengenali suatu pola tertentu. Misalnya sistem PR untuk mengenali huruf dari tulisan tangan, walaupun terdapat perbedaan penulisan huruf A dari masing-masing orang tetapi PR dapat mengenali bahwa huruf tersebut adalah huruf A. Beberapa aplikasi dari PR antara lain : *Fingerprint Identification, Face Identification, Handwriting Identification, Optical Character Recognition, Biological Slide Analysis, Robot Vision* dan lainnya.

2.2.1 Penerapan *Pattern Recognition*

Seperti yang telah disebutkan di atas bahwa AI merupakan salah satu cabang Ilmu Komputer. Tapi karena kompleksitas area AI maka dibuat sub-sub bagian yang dapat berdiri sendiri dan dapat saling bekerja sama dengan sub bagian lain atau dengan disiplin ilmu lain[14]. Berikut ini beberapa cabang ilmu sub bagian dari AI

1. *Character Recognition*

Salah satu area peengenalan pola yang secara umum menangani permasalahan otomatisasi dan informasi. Sistem OCR mempunyai *front end device* yang terdiri dari lensa scan , document, data transport dan sebuah detector.

2. Speech Recognition

Pengenalan pola yang secara umum telah banyak dikembangkan saat ini. Sistem ini mengijinkan kita untuk berkomunikasi antara manusia dengan memasukan data ke komputer. Meningkatkan efisiensi industri manufaktur, mengontrol mesin , dan sebagainya.

3. *Face Recognition*

Salah satu pengaplikasian sistem pengenalan pola yang ditujukan kepada pengenalan pola wajah , *face recogniton* adalah sebuah sistem yang mengenali image wajah manusia yang biasa digunakan untuk otomatisasi dan security sebuah insdustri.

4. *Machine Vision*

Sebuah sistem mesin yang menggunakan pengenalan pola sebagai dasar dari tahapannya. Mesin ini menangkap sebuah atau sekelompok object dengan kamera dan selanjutnya dianalisa untuk di deskripsikan.

2.3 Pengolahan Citra Digital

Pengolahan citra digital artinya melakukan pemrosesan gambar berdimensi dua melalui komputer digital. pengolahan citra adalah istilah umum untuk berbagai teknik yang keberadaanya untuk memanipulasi dan memodifikasi citra dengan berbagai cara [13].Foto adalah contoh gambar berdimensi dua yang dapat diolah dengan mudah. Setiap foto dalam bentuk citra digital dapat diolah menggunakan perangkat lunak tertentu. Misalnya citra yang gelap dapat diolah supaya menjadi lebih terang, merubah format warna citra menjadi grayscale, resize citra dan lain sebagainya.

2.3.1 Grayscale

Citra grayscale menangani gradasi warna hitam dan putih, yang menghasilkan efek warna abu-abu. Warna gambar dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih [14]. Berikut adalah rumus konversi citra berwana (RGB) menjadi grayscale.

$$I = (0.2989 * R)+(0.5870*G)+(0.1141*B) \quad (2.1)$$

Keterangan :

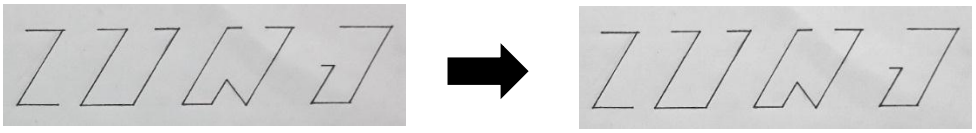
I = Fungsi pencarian nilai skala keabu-abuan

R = komponen nilai merah *(Red)* dari suatu titik *pixel*

G = komponen nilai hijau *(Green)* dari suatu titik *pixel*

B = komponen nilai biru *(Blue)* dari suatu titik *pixel*

Persamaan di atas merupakan salah satu rumus yang digunakan untuk mengkonversi citra berwarna menjadi grayscale. Persamaan 2.1 dipilih dalam penelitian ini karena mata manusia secara alami lebih sensitif terhadap cahaya merah dan hijau. Maka dari itu, warna-warna ini diberi bobot yang lebih tinggi untuk memastikan bahwa keseimbangan intensitas relatif dalam citra grayscale yang dihasilkan mirip dengan citra warna RGB [15].



Gambar 2. 1 Contoh hasil *Grayscale*

2.3.2 Sauvola Threshold

Sauvola threshold merupakan metode *threshold* yang mampu mendeteksi citra tulisan tangan dengan sangat baik, dengan cara menghitung ambang menggunakan rentang nilai dinamis dari nilai standar deviasi citra *grayscale* [16]. *Sauvola* termasuk dalam *local threshold* dimana nilai ambang ditentukan oleh nilai pixel tetangga, tujuan dilakukannya proses *threshold* adalah untuk menyederhanakan bentuk citra.

$$T(x,y) = m(x,y) * (1 + k * (\frac{s(x,y)}{R} - 1)) \quad (2.2)$$

$$m(x,y) = \frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} G_{i,j}}{i*j} \quad (2.3)$$

$$s(x,y) = \sqrt{\frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} (img(i,j)-m(x,y))^2}{(i*j)-1}} \quad (2.4)$$

Keterangan :

R : Nilai maksimum dari standar deviasi (128 untuk citra grayscale)

k : Kernel dengan nilai antara 0.2 – 0.5.

m : Fungsi yang menghasilkan nilai rata-rata dari sejumlah pixel citra.

s : Fungsi yang menghasilkan nilai standar deviasi dari sejumlah pixel citra

T : Fungsi yang menghasilkan nilai threshold (ambang)

x : Nilai koordinat lebar citra dalam rumus (i)

y : Nilai koordinat tinggi citra dalam rumus (j)

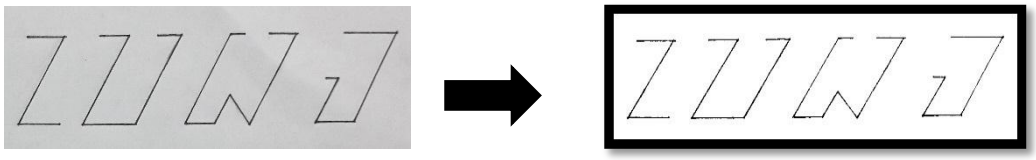
Setelah nilai ambang T(x,y) sudah didapatkan, selanjutnya masukkan ke persamaan di bawah ini.

$$f(x,y) = \begin{cases} 0, & \text{img}(x,y) < \text{T}(x,y) \\ 255, & \text{img}(x,y) \geq \text{T}(x,y) \end{cases} \quad (2.5)$$

Keterangan :

f : Fungsi yang menghasilkan nilai 0 atau 255

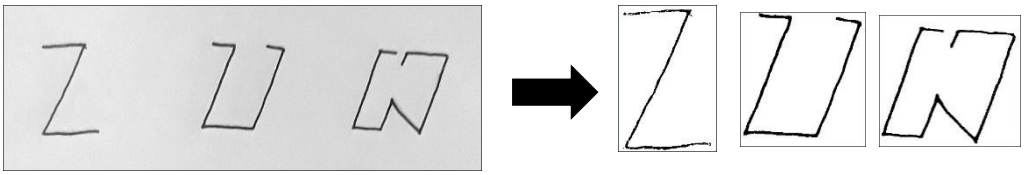
img : Nilai grayscale citra



Gambar 2. 2 Contoh Hasil *Sauvola Threshold*

2.3.3 Segmentasi Citra (*Image Segmentation*)

Segmentasi citra adalah teknik untuk membagi citra menjadi beberapa daerah atau region dimana setiap daerah memiliki kemiripan atribut [16]. Beberapa teknik segmentasi antaralain meliputi : pengambangan, penandaan komponen terhubung, segmentassi berbasis cluster, dan transformasi hough. Fokus pada segmentasi citra pada penelitian ini adalah menggunakan pengambangan dan penandaan komponen terhubung, contoh segmentasi pada citra input dapat dilihat pada gambar 2.4.



Gambar 2. 3 Contoh Gambar Segmentasi Citra

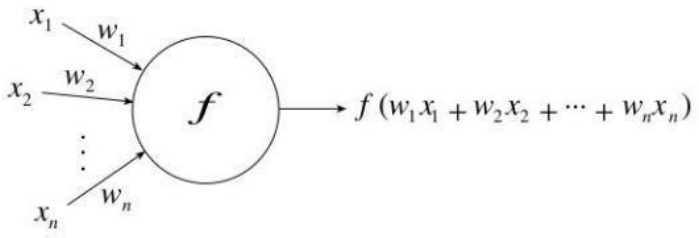
2.4 Teknik *Learning* (Belajar)

Teknik *learning* merupakan salah satu bagian dari ilmu kecerdasan buatan. Dalam learning kita tidak harus tahu aturan yang berlaku dalam sistem yang dibuat, melainkan aturan yang diharapkan bisa secara otomatis ditemukan. Proses belajar dalam teknik learning menggunakan data-data masukan sebagai pengalaman yang baru untuk dapat meningkatkan performasi dari sistem. Program komputer yang sanggup belajar adalah program yang bisa meningkatkan performasinya melalui pengalaman [15]. Ada beberapa metode yang menggunakan teknik *learning* seperti *decision tree*, jaringan syaraf tiruan, algoritma genetika dan lain-lain.

2.5 *Artificial Neural Networks*

Artificial neural networks atau biasa disingkat dengan ANN adalah model dari kemampuan jaringan saraf biologis untuk memproses informasi [17]. Jaringan

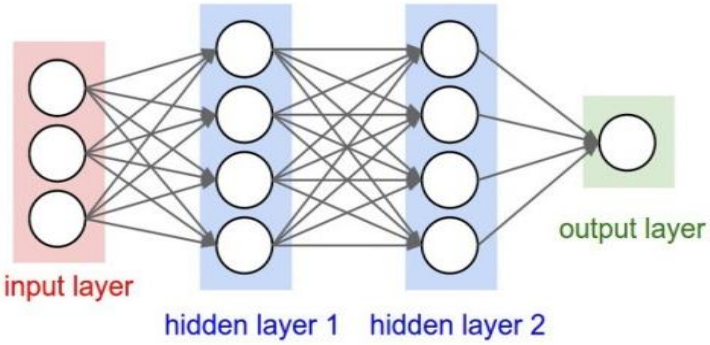
saraf biologis adalah sistem yang mengatur dirinya sendiri dan masing-masing *neuron* juga mengorganisir struktur dirinya dalam kemampuan memproses informasi dalam berbagai cara. Perbedaan utama antara jaringan saraf dan sistem komputer konvensional adalah paralelisme besar dan redudansi yang mereka eksploitasi untuk menangani unit komputasi individual yang tidak dapat diandalkan.



Gambar 2. 4 Ilustrasi Sebuah Neuron pada ANN

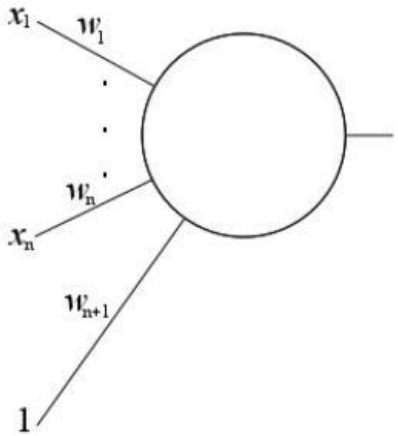
Gambar 2.5 menampilkan ilustrasi dari sebuah neuron pada *artificial neural networks* dengan x_1, x_2, ..., x_n sebagai masukan. Setiap masukan memiliki bobot yang berhubungan, yang berarti bahwa informasi yang masuk x_i dikalikan sesuai dengan bobotnya w_i.

Artificial neural networks terdiri dari beberapa *neuron* yang saling terhubung. Pada umumnya struktur *artificial neural networks* dibagi menjadi tiga, yaitu *input layer*, *hidden layer*, dan *output layer*. Jumlah *neuron* atau unit setiap *layer* disesuaikan dengan informasi yang akan diproses. Jumlah *input layer* dan *output layer* adalah satu, sedangkan jumlah *hidden layer* dapat satu atau lebih. Gambar 2.6 menampilkan ilustrasi dari arsitektur *artificial neural networks.*



Gambar 2. 5 Arsitektur *Artificial Neural Networks*

Setiap unit *neuron* disetiap *layer* tidak terhubung satu sama lain. Setiap unit atau *neuron* mentransmisikan nilai ke setiap *neuron* di *layer* setelahnya, kecuali *layer* terakhir (*output layer*). Dalam kebanyakan kasus pembelajaran dapat dinyatakan dengan lebih mudah dengan menambahkan bias. Gambar 2.7 menunjukan ilustrasi *neuron* dengan bias.



Gambar 2. 6 Neuron dengan Bias

Vektor masukan (x_1, x_2, ..., x_n) harus ditambah satu dan dihasilkan ($_{n+1}$)-dimensi vektor (x_1, x_2, ..., $x_{n,\ 1}$). Bobot yang menghubungkan *neuron* juga ditambahkan (w_1, w_2, ..., $w_{n,}$ w_{n+1}).

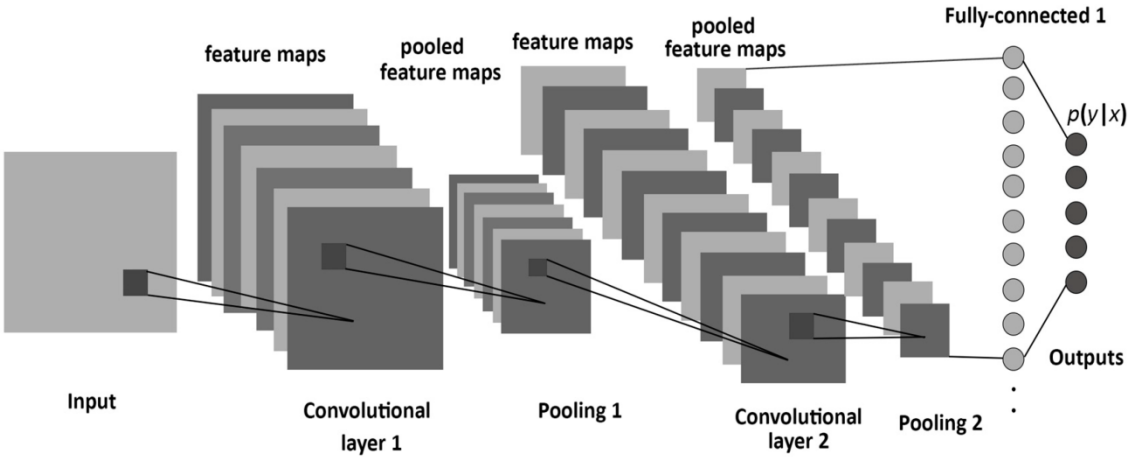
2.5.1 *Deep Learning*

Deep Learning memungkinkan model komputasi yang terdiri dari beberapa lapisan pengolahan untuk mempelajari representasi data dengan berbagai tingkat

abstraksi. Metode ini telah secara dramatis memperbaiki state-of-the-art dalam speech recognition, pengenalan objek visual, deteksi objek dan banyak domain lainnya seperti penemuan obat dan genomik.

Deep learning menemukan struktur yang rumit dalam kumpulan data yang besar dengan menggunakan algoritma backpropagation untuk menunjukkan bagaimana sebuah mesin harus mengubah parameter internalnya yang digunakan untuk menghitung representasi pada setiap lapisan dari representasi pada lapisan sebelumnya. Jaring konvolusi yang dalam telah membawa terobosan dalam memproses gambar, video, ucapan dan audio, sedangkan jala berulang telah menyoroti data sekuensial seperti teks dan ucapan [13].

2.6 *Convolutional Neural Network* (CNN)



Gambar 2. 7 Arsitektur *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu jenis *neural network* yang biasa digunakan pada data image. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada sebuah citra. Arsitektur dari CNN dibagi menjadi 3 bagian besar, *Convolutional Layer*, *Pooling Layer*, dan *Fully-Connected Layer*.

2.6.1 *Convolutional Layer*

Convolutional layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sekumpulan filter dengan panjang dan tinggi (*pixels*). Masing-masing filter ini akan digeser keseluruh bagian dari gambar satu persatu. Setiap pergeseran akan dilakukan operasi “dot” antara input dan nilai dari filter

tersebut sehingga menghasilkan sebuah output atau biasa disebut sebagai *activation map* atau *feature map*.

Ada 3 *hyperparameter* yang harus ditentukan dalam sebuah *convolutional layer*, yaitu *depth, stride* dan *zero-padding*.

1. Depth

   Depth adalah jumlah filter yang digunakan dalam suatu convolutional layer. Setiap filter akan mencari suatu ciri yang berbeda dalam masukan.

2. Stride

   Stride adalah parameter yang menentukan jumlah pergeseran filter. Jika nilai stride adalah 2, maka filter akan bergeser sebanyak 2 pixel secara horizontal lalu vertikal. Semakin kecil stride maka akan semakin detail informasi yang kita dapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan stride yang besar.

3. Zero-Padding

   Zero-Padding adalah parameter yang menentukan jumlah pixel berisi nilai 0 yang akan ditambahkan di setiap sisi dari masukan. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi output dari feature map. Dengan ditambahkannya zero-padding, maka ukuran citra output dan input akan tetap sama sehingga mengurangi informasi yang terbuang.

Pada *convolutional layer* akan dilakukan konvolusi antara matriks citra masukan dengan matriks-matriks filter F. Filter-filter ini akan digeser ke seluruh permukaan gambar sehingga menghasilkan keluaran matriks *feature map* FM.

Dengan *hyperparameter* di atas maka akan dihasilkan *feature map* FM dengan ukuran yang didapat dari rumus berikut [25]:

$$featuresize = \frac{inputsize - filtersize + 2pad}{str} + 1 \quad (2.6)$$

Keterangan:

featurezise = ukuran *feature map*

inputsize = ukuran matriks masukan

filtersize = ukuran filter

pad = zero padding
str = stride

Langkah-langkah yang dilakukan pada *convolutional layer* adalah sebagai berikut. Pertama-tama di setiap sisi matriks citra masukan akan ditambah pixel berisi nilai 0 sesuai dengan nilai *zero-padding*. Setelah itu lakukan operasi konvolusi antara citra masukan dengan filter F, dimulai dari pojok kiri atas, lalu bergeser ke kanan lalu ke bawah. Tambah setiap hasil dengan bias filter bF. Lakukan untuk setiap filter F. Berikut adalah rumusnya.

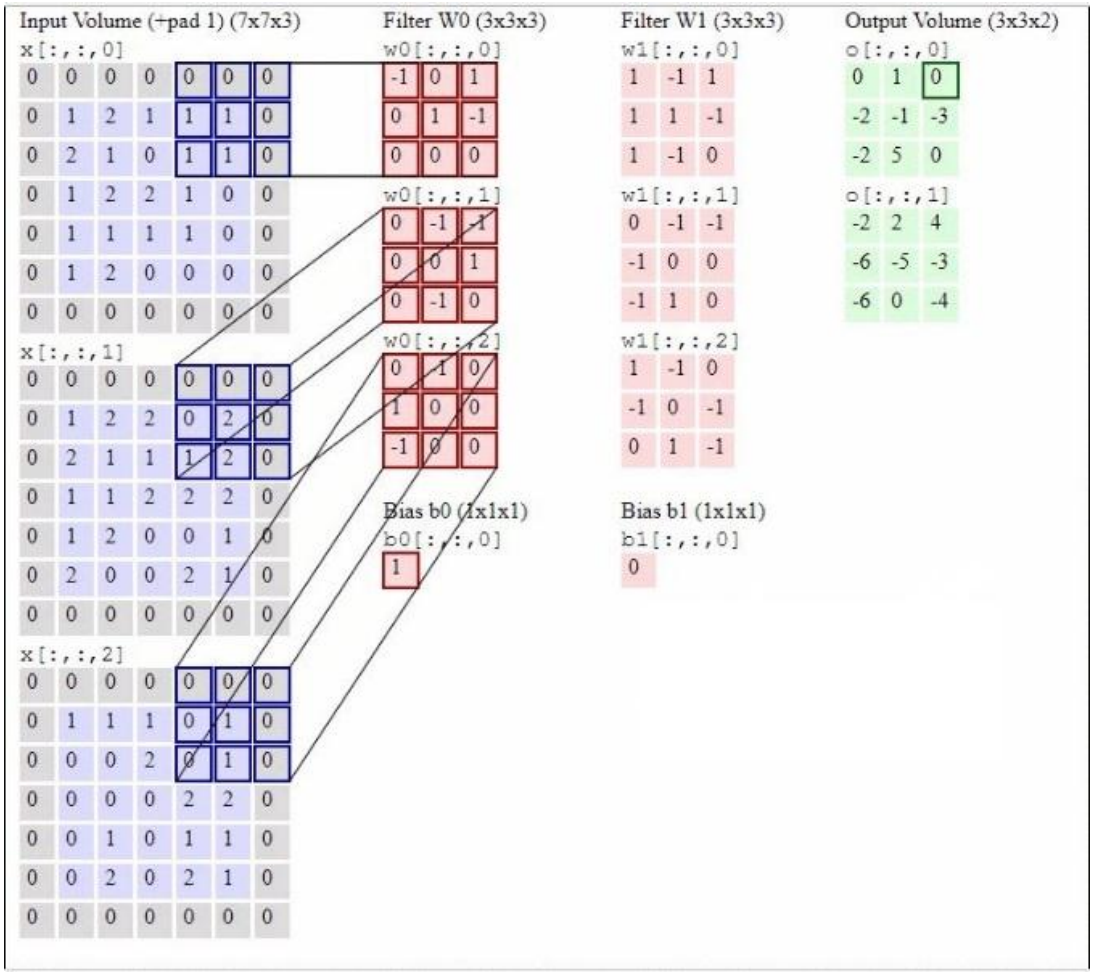
$$\mathrm{FM}_{x',y'} = \sum_{x=1}^{size} \sum_{y=1}^{size} (\mathrm{A_{x,y}} * \mathrm{F_{x,y}}) + \mathrm{bF} \quad (2.7)$$

Keterangan:
$\mathrm{FM}_{x',y'}$ = pixel *feature map* ke x', y'
$\mathrm{A_{x,y}}$ = pixel matriks input yang telah diberi *zero padding* ke x, y
$\mathrm{F_{x,y}}$ = pixel filter ke x, y
size = ukuran filter

Lalu jalankan fungsi aktivasi pada tiap *feature map* FM. Setiap pixel *feature map* FM akan dimasukkan ke dalam fungsi aktivasi ReLU, dimana setiap pixel yang memiliki nilai < 0 akan dijadikan 0, dengan rumus

$$ReLU(x) = \max(0, x) \quad (2.8)$$

dimana x adalah pixel.
Feature map FM yang telah dimasukkan ke dalam fungsi aktivasi ReLU akan menghasilkan *feature map* R. Ulangi untuk tiap *feature map.*



Gambar 2. 8 Cara Kerja *Convolutional Layer*

2.6.2 *Pooling Layer*

Pooling layer terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan bergeser pada seluruh area feature map. *Pooling* yang akan digunakan dalam penelitian ini adalah *Max Pooling*. Tujuan dari penggunaan *pooling layer* adalah mengurangi dimensi dari *feature map* (*downsampling*), sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit.

Max pooling layer bekerja dengan cara memilih nilai maksimum dalam suatu jendela. Pemilihan ini akan diulangi dengan menggeser jendela ke seluruh permukaan citra sehingga menghasilkan keluaran matriks *feature map* P yang berisi nilai-nilai maksimum yang terpilih. Terdapat dua *hyperparameter* dalam *pooling layer*, yaitu ukuran jendela dan *stride*. Dengan *hyperparameter* di atas maka akan dihasilkan *feature map* P dengan ukuran yang didapat dari rumus berikut: [25]

$$featuresize = \frac{inputsize - windowsize}{str} + 1 \quad (2.9)$$

Keterangan:

featurezise = ukuran *feature map*

inputsize = ukuran matriks masukan

windowsize = ukuran jendela

str = *stride*

Cara kerja *pooling layer* adalah sebagai berikut. Letakkan jendela pada pojok kiri atas *feature map.* Berikut adalah rumusnya.

$$P_{x',y'} = \max(R_{x,y}, \dots, R_{x+window,y+window}) \quad (2.10)$$

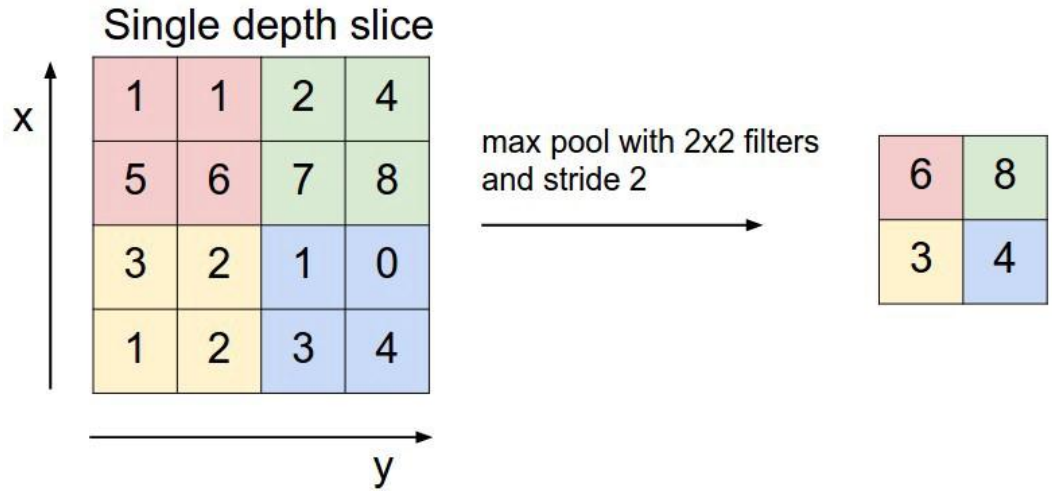
Keterangan:

$P_{x',y'}$ = pixel *feature map pooling* ke x', y'

$R_{x,y}$ = pixel *feature map ReLU* ke x, y

window = ukuran jendela

Gambar 2.9 menggambarkan contoh cara kerja max pooling untuk suatu irisan feature map.



Gambar 2. 9 Cara Kerja *Max Pooling* Untuk Suatu Irisan *Feature Map*

2.6.3 *Fully Connected Layer*

Fully Connected Layer pada intinya adalah sebuah neural netrowk multilayer perceptron (MLP), yang memiliki beberapa *hidden layer, activation function, output layer* dan *loss function*. *Fully connected layer*-lah yang akan berperan untuk mengklasifikasi data masukan.

Output yang dihasilkan dari *pooling layer* masih berbentuk *multidimensional array*, sehingga akan di-*flatten* terlebih dahulu untuk mengubah data menjadi vektor sebelum dijadikan input untuk *fully connected layer* [19].

2.6.3.1 *Input Layer*

Input layer X adalah penggabungan matriks-matriks *feature map* P yang didapat dari *pooling layer* dan direntangkan menjadi sebuah vektor sepanjang jumlah pixel keseluruhan ketiga *feature map*. Total *node input layer* adalah jumlah dari *pixel feature map* P. Nilai-nilai *node input layer* X akan digunakan pada perhitungan *hidden layer* Z.

2.6.3.2 *Hidden Layer*

Setiap *node* di *input layer* X akan mengirimkan sinyal input kepada setiap *hidden layer* Z. Setiap *node* X_i akan dikalikan dengan bobot $V_{j,i}$ lalu ditambahkan dengan bias $V_{0,i}$ untuk menghasilkan nilai masukan z_in_i. Rumus penghitungan masukan *hidden layer* z_in adalah sebagai berikut.

$$z\_in_i = \sum_{i=1}^{n} X_j * V_{j,i} + V_{0,i} \quad (2.11)$$

Keterangan:

z_in_i = masukan untuk *node hidden layer Z* ke-*i* dengan jumlah node *n*

X_j = *node X* ke-*j*

$V_{j,i}$ = *weight V* untuk *node Xj* dan *node Zi*

$V_{o,i}$ = bias *V* untuk *node* z_in_i

Setelah itu hitung nilai keluaran Z dengan mengaktifkan nilai masukan z_in menggunakan fungsi aktivasi ReLU, sebagai berikut.

$Z_i = ReLU(z\_in_i) = \max(0, z\_in_i)$ (2.12)

Z_i = keluaran untuk *node hidden layer* ke-*i*

z_in_i = masukan untuk *node hidden layer* ke-*i*

Nilai-nilai matriks *hidden layer* Z akan digunakan pada perhitungan *output layer* Y.

2.6.3.3 *Output Layer*

Setiap *node* di *output layer* Z_j akan mengirimkan sinyal input kepada setiap *output layer* Y_i. Setiap *node* Z akan dikalikan dengan bobot $W_{j,i}$ lalu ditambahkan dengan bias $W_{0,i}$. Penghitungan masukan *output layer* y_in adalah sebagai berikut.

$$y\_in_i = \sum_{i=1}^{n} Z_j * W_{j,i} + W_{0,i} \quad (2.13)$$

y_in_i = masukan untuk *node hidden layer Z* ke-*i* dengan jumlah node *n* buah

Z_j = *node Z* ke-*j*

$W_{j,i}$ = *weight W* untuk *node Zj* dan *node Yi*

$W_{o,i}$ = bias W untuk *node* y_in_i

Setelah itu aktifkan nilai keluaran Y dengan menggunakan fungsi aktivasi *softmax*.

$$Y_i = \frac{e^{y\_in_i}}{\sum_{i=1}^{m} e^{M}} \quad (2.14)$$

Keterangan:

Y_i = keluaran untuk *node output layer* ke-*i*

y_in_i = masukan untuk *node output layer* ke-*i*

M = semua masukan untuk *node output layer,* berjumlah *m* buah

Dari hasil keluaran Y dapat dipetakan klasifikasi kelas huruf berbentuk matriks *one-hot*.

2.6.3.4 *Backpropagation*

Pada tahap *backpropagation* [14] akan dilakukan pelatihan pada kelas huruf. Pada tahap ini akan dilakukan proses penyesuaian tiap *weight* dan *bias* berdasarkan *error* yang didapat pada tahap *feedforward.* Pertama akan dihitung gradien dari *loss function* terhadap semua parameter (*weight* dan *bias*) yang ada dengan mencari

turunan parsial (*partial derivative*) dari fungsi tersebut. Setelah itu *update* semua parameter dengan menggunakan *Stochastic Gradient Descent (SGD)*.

1. Penghitungan *loss function*

   Loss function adalah sebuah fungsi yang mengukur seberapa bagus performa dari *neural network* dalam melakukan prediksi terhadap target. *Loss function* akan menghitung *loss*, yaitu selisih dari nilai *output* dengan nilai target yang diharapkan. Untuk masalah klasifikasi, *loss function* yang biasa digunakan adalah *cross-entropy loss function* :

   $$L = -\sum_{i=1}^{m} t_i \log(Y_i) \quad (2.15)$$

   Keterangan:

   L = *loss / error*

   t_i = nilai vektor ke-i *one-hot* target yang diharapkan

   Y_i = nilai matriks output pada tahap *feedforward* ke-i dengan jumlah kelas *m* buah

2. Penghitungan gradien kesalahan terhadap parameter bobot W_{ji}

   Setelah itu hitung gradien kesalahan terhadap parameter bobot W_{ji}, dengan menggunakan rumus *chain rule*. [30]

   $$\frac{\partial L}{\partial W_{ji}} = \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y\_in_i}\frac{\partial y\_in_i}{\partial W_{ji}} \quad (2.16)$$

   $$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1-Y_i)} \quad (2.17)$$

   $$\frac{\partial Y_i}{\partial y\_in_i} = Y_i(1 - Y_i) \quad (2.18)$$

   $$\frac{\partial y\_in_i}{\partial W_{ji}} = Z_j \quad (2.19)$$

   Maka untuk filter W_{ji}

   $$\frac{\partial L}{\partial W_{ji}} = Y_i(1 - Y_i)Z_j \quad (2.20)$$

   dan untuk bias $W_{0,i}$

   $$\frac{\partial L}{\partial W_{ji}} = Y_i(1 - Y_i) \quad (2.21)$$

   Keterangan:

$\frac{\partial L}{\partial W_{ji}}$ = gradien bobot W_{ji} terhadap *loss* L

$\frac{\partial L}{\partial W_{0,i}}$ = gradien bias $W_{0,i}$ terhadap *loss* L

$\frac{\partial L}{\partial Y_i}$ = gradien nilai *node* Y_i terhadap *loss* L

$\frac{\partial Y_i}{\partial y\_in_i}$ = gradien nilai matriks y_in_i terhadap nilai *node* Y_i

$\frac{\partial y\_in_i}{\partial W_{ji}}$ = gradien bobot W_{ji} terhadap nilai matriks y_in_i

Y_i = nilai *node output layer* ke-i

y_in_i = nilai masukan *output layer* ke-i

Z_j = nilai *node hidden layer* ke-j

3. Penghitungan gradien kesalahan terhadap parameter bobot V_{kj} dan bias $V_{0,j}$

   Setelah itu hitung gradien kesalahan terhadap parameter bobot V_{kj}, dengan menggunakan rumus *chain rule*.

$$\frac{\partial L}{\partial V_{kj}} = \sum_i^m \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y\_in_i}\frac{\partial y\_in_i}{\partial Z_j}\frac{\partial Z_j}{\partial z\_in_j}\frac{\partial z\_in_j}{\partial V_{kj}} \quad (2.22)$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1-Y_i)} \quad (2.23)$$

$$\frac{\partial Y_i}{\partial y\_in_i} = Y_i(1 - Y_i) \quad (2.24)$$

$$\frac{\partial y\_in_i}{\partial Z_j} = W_{ji} \quad (2.25)$$

$$\frac{\partial Z_j}{\partial z\_in_j} = Z_j\left(1 - Z_j\right) \quad (2.26)$$

$$\frac{\partial z\_in_j}{\partial V_{kj}} = X_k \quad (2.27)$$

   Maka untuk filter V_{kj}

$$\frac{\partial L}{\partial V_{kj}} = \sum_i^m (Y_i - t_i)(W_{ji})(Z_j\left(1 - Z_j\right))(X_k) \quad (2.28)$$

   dan untuk bias $V_{0,j}$

$$\frac{\partial L}{\partial V_{kj}} = \sum_i^m (Y_i - t_i)(W_{ji})(Z_j\left(1 - Z_j\right)) \quad (2.29)$$

4. Penghitungan gradien kesalahan terhadap parameter filter F dan bias bF

Setelah itu hitung gradien kesalahan terhadap parameter filter F. Sesuai aturan *chain rule*, sebelumnya kita harus mencari gradien untuk *layer-layer* sebelumnya terlebih dahulu. Lalu hitung gradien masing-masing filter dengan gradien yang didapat sebelumnya.

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial X}\frac{\partial X}{\partial P}\frac{\partial P}{\partial F} \tag{2.30}$$

Keterangan:

$\frac{\partial L}{\partial F}$ = gradien nilai filter F terhadap *loss* L

$\frac{\partial L}{\partial X}$ = gradien nilai *node* X terhadap *loss* L

$\frac{\partial X}{\partial P}$ = gradien nilai *feature map* P terhadap nilai *node* X

$\frac{\partial P}{\partial F}$ = gradien nilai filter F terhadap nilai *feature map* P

Pertama hitung nilai $\frac{\partial L}{\partial X}$. Hitung gradien kesalahan terhadap parameter bobot X_k, dengan menggunakan rumus *chain rule*.

$$\frac{\partial L}{\partial X_k} = \sum_j^n \sum_i^m \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y\_in_i}\frac{\partial y\_in_i}{\partial Z_j}\frac{\partial Z_j}{\partial z\_in_j}\frac{\partial z\_in_j}{\partial X_k} \tag{2.31}$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1-Y_i)} \tag{2.32}$$

$$\frac{\partial Y_i}{\partial y\_in_i} = Y_i(1 - Y_i) \tag{2.33}$$

$$\frac{\partial y\_in_i}{\partial Z_j} = W_{ji} \tag{2.34}$$

$$\frac{\partial Z_j}{\partial z\_in_j} = Z_j\left(1 - Z_j\right) \tag{2.35}$$

$$\frac{\partial z\_in_j}{\partial X_k} = V_{kj} \tag{2.36}$$

maka

$$\frac{\partial L}{\partial X_k} = \sum_j^n \sum_i^m (Y_i - t_i)(W_{ji})(Z_j\left(1 - Z_j\right))(V_{kj}) \tag{2.37}$$

Keterangan:

$\frac{\partial L}{\partial X_k}$ = gradien nilai *node* X_k terhadap *loss* L

$\frac{\partial L}{\partial Y_i}$ = gradien nilai *node* Y_i terhadap *loss* L

$\frac{\partial Y_i}{\partial y\_in_i}$ = gradien nilai matriks y_in_i terhadap nilai *node* Y_i

$\frac{\partial y\_in_i}{\partial Z_j}$ = gradien bobot W_{ji} terhadap nilai matriks y_in_i

$\frac{\partial Z_j}{\partial z\_in_j}$ = gradien nilai matriks z_in_j terhadap nilai *node* Z_j

$\frac{\partial z\_in_j}{\partial X_k}$ = gradien nilai *node* X_k terhadap nilai matriks z_in_j

Y_i = nilai *node output layer* ke-i

y_in_i = nilai masukan *output layer* ke-i

Z_j = nilai *node hidden layer* ke-j

z_in_j = nilai masukan *hidden layer* ke-j

X_k = nilai *node input layer* ke-k

m = jumlah *node output layer* Y

n = jumlah *node hidden layer* Z

Setelah itu kita hitung hitung nilai $\frac{\partial X}{\partial P}$ dengan menggunakan nilai yang didapat dari perhitungan $\frac{\partial L}{\partial X}$. Nilai ini didapat dari mencari gradien pada layer *pooling layer*. Pertama kita kembalikan lagi bentuk vektor $\frac{\partial L}{\partial X}$ menjadi gradien *feature map*. Untuk operasi *max-pooling feature map* P', gradien $\frac{\partial X}{\partial P}$ melihat kepada nilai-nilai maksimum *feature map* sesuai dengan hasil *max-pooling* pada tahap *feedforward*. Gradien $\frac{\partial X}{\partial P}$ untuk pixel-pixel dengan nilai maksimum adalah $\frac{\partial \mathrm{L}}{\partial \mathrm{X}}$, sedangkan gradien untuk pixel-pixel lainnya adalah 0.

Lalu kita hitung hitung nilai $\frac{\partial P}{\partial F}$ dengan menggunakan nilai yang didapat dari perhitungan $\frac{\partial X}{\partial P}$. Nilai ini didapat dari mencari gradien pada layer *convolution layer*. Untuk menghitungnya, lakukan operasi konvolusi seperti pada tahap

feedforward, dengan masukannya adalah matriks citra masukan (yang sudah diberi *zero-padding*) A, dan filternya adalah gradien *feature map* $\frac{\partial X}{\partial P}$. Hasilnya adalah gradien filter $\frac{\partial P}{\partial F}$.

Sedangkan untuk gradien bias filter, jumlahkan setiap nilai pada gradien filter. Lakukan untuk semua filter.

$$\frac{\partial L}{\partial bF} = \sum_{x}^{size} \sum_{y}^{size} \frac{\partial P_{x,y}}{\partial F_{x,y}} \quad (2.38)$$

Keterangan:

$\frac{\partial L}{\partial bF}$ = gradien bias filter bF terhadap *loss* L

$\frac{\partial P_{x,y}}{\partial F_{x,y}}$ = gradient nilai filter F ke-x,y terhadap nilai *feature map* P ke-x,y

size = ukuran filter

5. *Update* nilai parameter

   Setelah semua gradien parameter dihitung, *update* parameter-parameter dengan menggunakan *Stochastic Gradient Descent* [25]. *Stochastic Gradient Descent* adalah metode optimasi yang digunakan untuk menemukan nilai bobot optimal agar didapatkan nilai *loss* yang kecil, di mana nilai *loss* merepresentasikan tingkat kesalahan (*error*) pada sistem. Pencarian nilai bobot dilakukan dengan cara merubah bobot secara bertahap ketika pelatihan, dengan menggunakan parameter yang disebut *learning rate*, yaitu seberapa cepat *neural network* melakukan pembelajaran.

Berikut adalah perhitungan untuk *update* nilai parameter bobot W'.

$$W_{x,y}' = W_{x,y} - \alpha(\frac{\partial L}{\partial W_{x,y}}) \quad (2.39)$$

Keterangan:

$W_{x,y}'$ = Bobot W ke-x,y yang telah di-*update*

$W_{x,y}$ = Bobot W ke-x,y awal

$\frac{\partial L}{\partial W_{x,y}}$ = Gradien bobot $W_{x,y}$ terhadap *loss* L

α = *learning rate*

Berikut adalah perhitungan untuk *update* nilai parameter bias W_0'.

$$W_{0,y}' = W_{0,y} - \alpha(\frac{\partial L}{\partial W_{x,y}}) \quad (2.40)$$

Keterangan:

$W_{0,y}'$ = Bias W_0 ke-y yang telah di-*update*

$W_{0,y}$ = Bias W_0 ke-y awal

$\frac{\partial L}{\partial W_{0,y}}$ = Gradien bias $W_{0,y}$ terhadap *loss* L

α = *learning rate*

Selanjutnya berikut adalah perhitungan untuk *update* nilai parameter bobot V'.

$$V_{x,y}' = V_{x,y} - \alpha(\frac{\partial L}{\partial V_{x,y}}) \quad (2.41)$$

Keterangan:

$V_{x,y}'$ = Bobot V ke-x,y yang telah di-*update*

$V_{x,y}$ = Bobot V ke-x,y awal

$\frac{\partial L}{\partial V_{x,y}}$ = Gradien bobot $V_{x,y}$ terhadap *loss* L

α = *learning rate*

Berikut adalah perhitungan untuk *update* nilai parameter bias V_0'.

$$V_{0,y}' = V_{0,y} - \alpha(\frac{\partial L}{\partial V_{x,y}}) \quad (2.42)$$

Keterangan:

$V_{0,y}'$ = Bias V_0 ke-y yang telah di-*update*

$V_{0,y}$ = Bias V_0 ke-y awal

$\frac{\partial L}{\partial V_{0,y}}$ = Gradien bias $V_{0,y}$ terhadap *loss* L

α = *learning rate*

Lalu berikut adalah perhitungan untuk *update* nilai parameter filter F'.

$$F_{x,y}' = F_{x,y} - \alpha(\frac{\partial L}{\partial F_{x,y}}) \quad (2.43)$$

Keterangan:

$F_{x,y}'$ = Bobot F ke-x,y yang telah di-*update*

$F_{x,y}$ = Bobot F ke-x,y awal

$\frac{\partial L}{\partial F_{x,y}}$ = Gradien bobot $F_{x,y}$ terhadap *loss* L

α = *learning rate*

Terakhir berikut adalah perhitungan untuk *update* nilai parameter filter bF'.

$$bF' = bF - \alpha(\frac{\partial L}{\partial bF}) \quad (2.44)$$

Keterangan:

bF' = Bias bF yang telah di-*update*

bF = Bias bF awal

$\frac{\partial L}{\partial bF}$ = Gradien bias bF terhadap *loss* L

α = *learning rate*

Hasil *update* parameter-parameter bobot V', bias V_0', bobot W', bias W_0', dan filter F' dan bias filter bF' akan digunakan pada iterasi *feedforward* selanjutnya.

2.7 Pengujian Sistem

Pengujian sistem adalah proses pemeriksaan atau evaluasi sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan-kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaaan-perbedaan antara hasil yang diterapkan dengan hasil yang terjadi [20]. Pengujian seharusnya meliputi tiga konsep berikut.

1. Demonstrasi validitas perangkat lunak pada masing-masing tahap disiklus pengembangan sistem.
2. Penentuan validitas sistem akhir dikaitkan dengan kebutuhan pemakai.
3. Pemeriksaan perilaku sistem dengan mengeksekusi sistem pada data sampel pengujian.

Pada dasarnya pengujian diartikan sebagai aktiitas yang dapat atau hanya dilakukan setelah pengkodean (kode program selesai). Namun, pengujian seharusnya dilakukan dalam skala lebih luas. Pengujian dapat dilakukan begitu spesifikasi kebutuhan telah dapat didefinisikan. Evaluasi terhadap spesifikasi dan perancangan

juga merupakan teknik dipengujian. Kategori pengujian dapat dikategorikan menjadi dua [20] yaitu :

1. Berdasarkan ketersediaan logik sistem, terdiri dari *Black box testing* dan *White box testing*.
2. Berdasarkan arah pengujian, terdiri dari Pengujian *top down* dan Pengujian *bottom up*.

2.7.1 Pengujian *Black Box*

Konsep *black box* digunakan untuk merepresentasikan sistem yang cara kerja di dalamnya tidak tersedia untuk diinspeksi. Di dalam *black box*, item-item yang diuji dianggap "gelap" karena logiknya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari *black box* [20].

Pada pengujian *black box*, kasus-kasus pengujian berdasarkan pada spesifikasi sistem. Rencana pengujian dapat dimulai sedini mungkin di proses pengembangan perangkat lunak. Teknik pengujian konvensional yang termasuk pengujian "black box" adalah sebagai berikut [20].

1. *Graph-based testing*
2. *Equivalence partitioning*
3. *Comparison testing*
4. *Orthogonal array testing*

Pada pengujian *black box*, kita mencoba beragam masukan dan memeriksa keluaran yang dihasilkan. Kita dapat mempelajari apa yang dilakukan kotak, tapi tidak mengetahui sama sekali mengenai cara konversi dilakukan. Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis skenario, dimana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use case* dan informasi analisis yang lain.

2.7.2 Pengujian Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value* atau *reference value*). Tingkat akurasi diperoleh dengan persamaan sebagai berikut.

$$Akurasi = \frac{Jumlah\ Karakter\ Sama}{Jumlah\ Seluruh\ Karakter}\ X\ 100\% \quad (2.45)$$

2.8 Bahasa Pemrograman

Bahasa pemrograman diperlukan untuk menjalankan instruksi-instruksi apa yang harus dilakukan komputer. Komputer tidak bisa memahami bahasa manusia, sehingga diperlukan penggunaan bahasa komputer di dalam program komputer. Penelitian ini menggunakan bahasa pemrograman Java untuk mengembangkan aplikasi dan habasa pemrograman SQL untuk mengelola *database*.

2.8.1 Java

Bahasa pemrograman Java dikembangkan oleh sebuah tim yang diketuai oleh James Gosling di Sun Microsystem. Java awalnya dikenal dengan Oak, yang didesain pada tahun 1991 untuk chip-chip yang tertanam pada peralatan-peralatan elektronik. Pada tahun 1995, diberi nama baru Java yang didesain ulang untuk mengembangkan aplikasi-aplikasi internet. Java telah menjadi sangat populer. Perkembangannya yang sangat cepat dan penerimaannya di kalangan pengguna dapat dijejak dari karakteristik perancangannya, khususnya dari janji pengembang Java bahwa begitu anda menciptakan suatu program maka anda bisa menjalankannya di mana saja [21].

Java memiliki banyak fitur, bahasa pemrograman bertujuan umum yang dapat digunakan untuk mengembangkan aplikasi-aplikasi tingkat tinggi. Saat ini Java tidak lagi hanya digunakan untuk pemrograman web, tetapi juga untuk aplikasi-aplikasi *standalone* bebas *platform* pada *server*, desktop, dan divais-divais bergerak(*mobile*) [21]. Bahasa Java juga telah digunakan untuk mengembangkan kode dalam berkomunikasi dan mengendalikan robot di Mars. Banyak perusahaan yang sebelumnya meremehkan keunggulan Java, namun sekarang malah menggunakannya untuk mengembangkan aplikasi-aplikasi terdistribusi yang dapat diakses oleh banyak konsumen melalui internet.

Java merupakan bahasa pemrograman yang tangguh terbukti handal pada banyak aplikasi. Terdapat tiga edisi Java [21], yaitu :

1. Java SE (*Standard Edition*)

Digunakan untuk mengembangkan aplikasi-aplikasi pada sisi client atau *applet*.

2. Java EE (*Enterprise Edition*)

   Digunakan untuk mengembangkan aplikasi-aplikasi pada sisi *server*, seperti *Java servlets* dan *Java server pages*.

3. Java ME (*Micro Edition*)

   Digunakan untuk mengembangkan aplikasi-aplikasi untuk divais bergerak,seperti telepon genggam.

Penelitian ini menggunakan Java SE untuk membuat aplikasi pengenalan tulisan tangan.

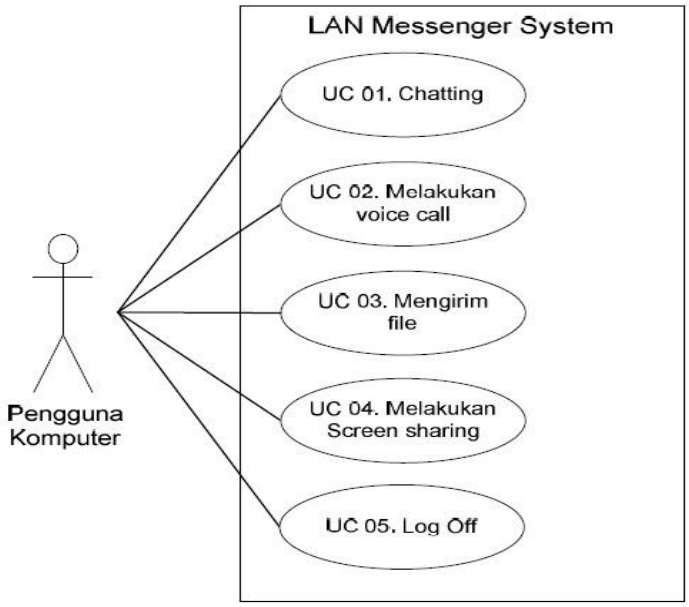
2.9 UML (*Unified Modeling Language*)

UML *(Uniefied Modeling Language)* adalah notasi yang lengkap untuk membuat visualisasi model suatu sistem. Sistem berisi informasi dan fungsi, tetapi secara normal digunakan untuk memodelkan sistem komputer. Di dalam pemodelan objek guna menyajikan sistem yang berorientasi objek kepada orang lain, akan sangat sulit dilakukan dalam bentuk kode bahasa pemrograman [22].

UML disebut sebagai bahasa pemodelan bukan metode. Bahasa pemodelan (sebagian besar grafik) merupakan notasi model yang digunakan untuk mendesain secara cepat. Bahasa pemodelan merupakan bagian terpenting dari metode. UML merupakan bahasa standar untuk penulisan *blueprint software* yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian. alat-alat dari sistem perangkat lunak. UML biasanya disajikan dalam bentuk diagram atau gambar yang meliputi *class* beserta atribut dan operasinya, serta hubunganantar kelas. UML terdiri dari banyak diagram diantaranya *use case, diagram, activity diagram, class diagram, dan sequence diagram.*

2.9.1 *Use Case* Diagram

Dalam konteks UML, tahap konseptualisasi dilakuak dengan pembuatan *use case diagram* yang sesungguhnya merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunannya.



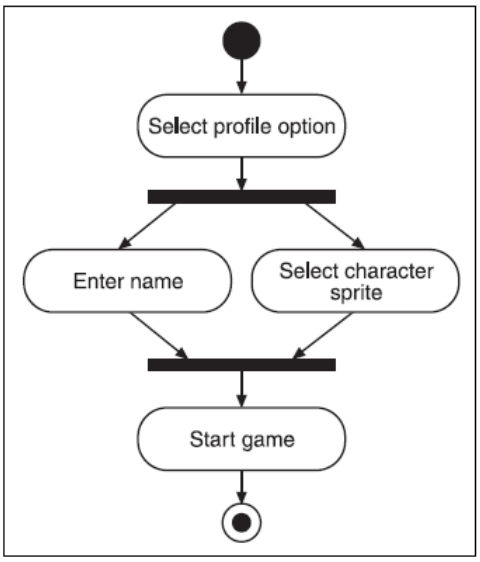
Gambar 2. 10 Contoh Penggunaan *Use case* Diagram

Use case diagram merupakan deskripsi lengkap tentang interaksi yang terjadi antara para aktor dengan sistem [22]. Dalam hal ini, setiap objek yang berinteraksi dengan sistem merupakan aktor untuk sistem, sementara *use case* merupakan deskripsi lengkap tentang bagaimana sistem berprilaku kepada aktornya. Aktor dalam *use case diagram* digambarkan sebagai ikon yang berbentuk manusia, sementara *use case* digambarkan elips yang berisi nama *use case* yang bersangkutan. Untuk mempermudah pemahaman, aktor biasanya dituliskan sebagai kata benda, sementara *use case* biasanya ditulisakan dengan kata kerja.

2.9.2 *Activity* Diagram

Activity Diagram pada dasarnya menggambarkan skenario secara grafis. A*ctivity diagram* cukup serupa dengan diagram alur (*flow chart),* yang membedakan hanya *swimlane* yang menunjukkan suatu *state* berada pada objek/kelas tertentu. Keunggulan dari *activity diagram* adalah lebih mudah

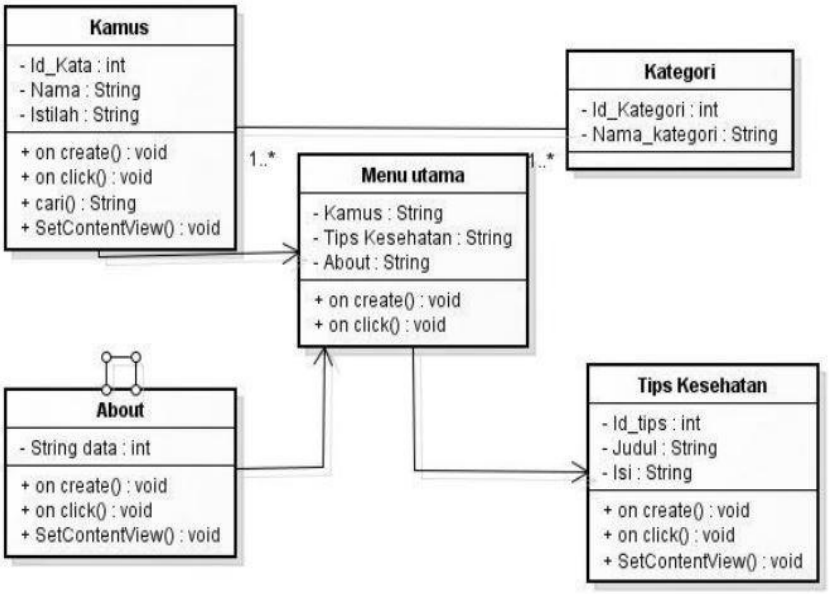
dipahami dibanding skenario. Selain itu, dengan menggunakan *activity diagram,* kita dapat melihat dibagian manakah sistem dari suatu skenario akan berjalan [22].



Gambar 2. 11* Contoh *Activity Diagram

2.9.3 *Class* Diagram

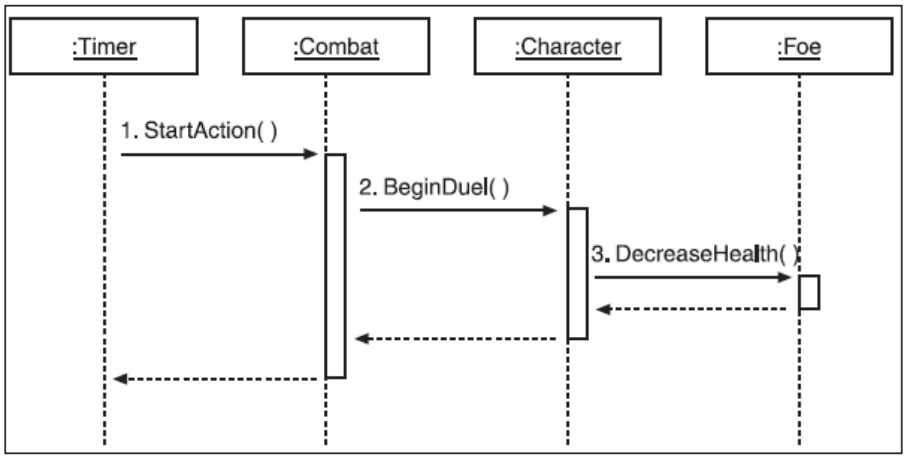
Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendeinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class diagram* menggambarkan struktur dan deskripsi kelas, package, dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi dan sebagainya.



Gambar 2. 12 Contoh *Class Diagram*

2.9.4 *Sequence* Diagram

Diagram sekuen menggambarkan kelakuan/perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek [22]. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* berserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

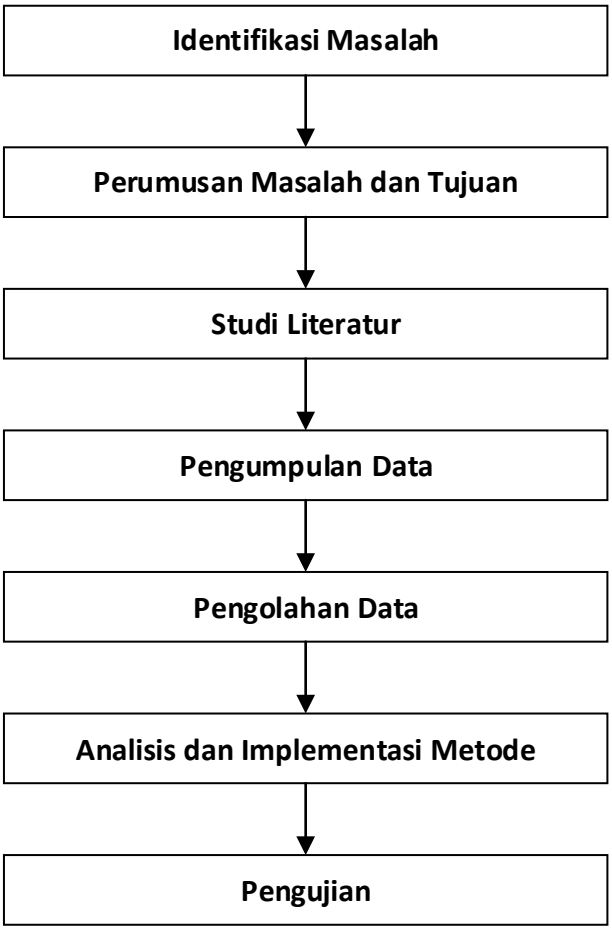


Gambar 2. 13 Contoh *Sequence* Diagram

2.10 Metodologi Penelitian

Metodologi peenelitian adalah sekumpulan peraturan, kegiatan dan prosedur yang digunakan oleh pelaku suatu disiplin (penelitian). Metologi merupakan analisis teoritis mengenai suatu cara atau metode. Penelitian merupakan suatu penyelidikan yang sstematis untuk meeningkatkan sejumlah pengetahuan .

Dalam penelitian ini metode penelitian yang digunakan adalah metode SDLC *(Software Development Life Cycle)* [10], metode penelitian ini digunakan karena tahapan pada penelitiannya sama dengan yang dilakukan oleh peneliti, tahapan penelitian yang dilakukan pada gambar 2.14



Gambar 2. 14 Skema Penelitian Yang Dilakukan

2.10.1 Metode Pengumpulan Data

Studi literatur yaitu pengumpulan data dengan mempelajari masalah yang akan dibahas serta mengumpulkan literatur[10]. Sumber-sumber yang diperlakukan berupa e-book, jurnal, tutorial dan beberapa informasi yang relevan dengan penelitian.

2.10.2 Metode Pembangunan Perangkat Lunak

Berikut adalah langkah-langkah yang dilakukan dalam proses Model Prototipe [12].

1. Pengumpulan Kebutuhan dan Analisis

   Tahap ini merupakan tahap awal dalam Model Prototipe. Kegiatan yang dilakukan yaitu menganalisis dan mengumpulkan kebutuhan data dengan cara membaca buku dan jurnal tentang metode yang digunakan dalam *preprocessing*

citra tulisan tangan, klasifikasi karakter tulisan tangan dan metode dalam pengecekan jawaban esai. Selain itu juga mengumpulkan sampel tulisan tangan dari orang yang berbeda.

2. Perancangan Cepat

   Pada tahap ini melakukan perancangan awal dari aplikasi yang dibangu, seperti perancangan antar muka aplikasi dan *database*. Perancangan awal dibuat berdasarkan dengan data-data yang sudah dikumpulkan pada tahap pengumpulan kebutuhan dan analisis.

3. Membangun Prototipe

   Tahap ini termasuk ke dalam tahap implementasi dari desain yang sudah dibuat pada tahap perancangan cepat. Tahap ini membuat *database* untuk menyimpan karakter tulisan tangan dan juga menulis program dalam bahasa Java.

4. Evaluasi Prototipe

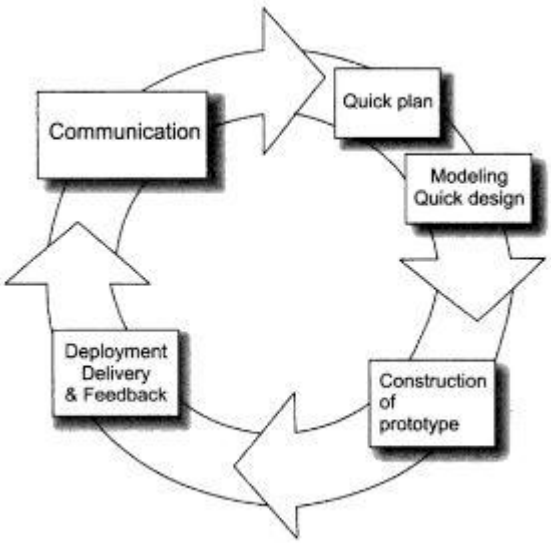
   Pada tahap ini melakukan pengujian terhadap aplikasi yang sudah dibuat oleh peneliti. Melihat apakah metode yang diterapkan pada tahap *preprocessing*, tahap klasifikasi dan tahap evaluasi jawaban ujian sudah berjalan dengan baik serta apakah ada fungsionalitas program yang tidak berjalan dengan baik.

5. Perubahan Desain dan Prototipe

   Tahap ini merupakan tahap memperbaiki aplikasi dari segi fungsionalitas dan juga metode-metode yang digunakan belum berjalan dengan baik pada tahap evaluasi prototipe yang sudah dilakukan.

6. Pengembangan Aplikasi

   Tahap ini merupakan tahap akhir dimana fungsionalitas aplikasi dan metode-metode yang diterapkan sudah berjalan sesuai dengan tujuan yang diharapkan. Sampai tahap ini, program sudah bisa melakukan pengenalan terhadap citra tulisan tangan yang diujikan.



Gambar 2. 15 Model *Prototype*

1. *Communication*

   Tahap ini dilakukan untuk menganalisa fungsional dari kebutuhan sistem.

2. *Quick plan*

   Tahap ini dilakukan untuk merancang fungsional kebutuhan sistem yang telah dianalisa pada tahap *Communication.*

3. *Modeling Quick design*

   Tahap ini dilakukan untuk membuat model basis data dan model antarmuka berdasarkan hasil perancangan dari tahap *Quik plan*.

4. *Contruction of prototype*

   Tahap ini dilakukan untuk pembangunan perangkat lunak yang akan diimplementasikan kedalam *sourcecode* dari hasil *Modeling Quick design.*

5. *Deployment Delivery & Feedback*

   Tahap ini dilakukan untuk pengembangan dan evaluasi terhadap sistem yang telah dibangun berdasarkan *Contruction of prototype*.