BAB 2
LANDASAN TEORI

2.1 Pengenalan Tulisan Tangan

Pengenalan tulisan tangan *Handwriting recognition* adalah kemampuan computer untuk menerima dan menafsirkan input tulisan tangan yang dapat dimengerti dari sumber seperti dokumen kertas, foto, layar sentuh dan perangkat lainnya[8].

2.2 Pengolahan Citra Digital

Pengolahan citra digital artinya melakukan pemrosesan gambar berdimensi dua melalui komputer digital. Pengolahan citra adalah istilah umum untuk berbagai teknik yang keberadaannya untuk memanipulasi dan memodifikasi citra dengan berbagai cara [9].

Suatu citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitude f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut.

Citra digital dapat ditulis dalam bentuk matrik seperti pada persamaan (2.1).

$$f(x,y) = \begin{matrix} f(0,0) & \dots & f(0,N-1) \\ \dots & \dots & \dots \\ f(M-1,0) & \dots & f(M-1,N-1) \end{matrix} \qquad (2.1)$$

Nilai pada suatu irisan antara baris dan kolom (pada posisi x,y) disebut dengan picture elements, image elements, pels, atau pixels. Istilah terakhir (pixel) paling sering digunakan pada citra digital.

2.2.1 Operasi Pengolahan Citra

2.2.1.1 RGB

RGB adalah suatu model warna yang terdiri atas 3 buah warna yaitu: merah *(Red)*, hijau *(Green)*, dan biru *(Blue)*, yang ditambahkan sebagai cara untuk menghasilkan bermacam-macam warna[10].

2.2.1.2 Grayscale

Citra Grayscale menangani gradasi warna hitam dan putih, yang tentu saja 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 [9]. Berikut adalah rumus konversi citra berwarna (RGB) menjadi grayscale dapat dilihat pada rumus (2.2).

$$I = (0.2989*R) + (0.5870*G) + (0.1141 * B) \quad (2.2)$$

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 mengonversi citra berwarna menjadi grayscale. Persamaan (2.2) 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 [10].

2.2.1.3 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* [12]. *Sauvola* termasuk dalam *local threshold* dimana nilai ambang ditentukan oleh nilai

pixel tetangga, tujuan dilakukannya proses *threshold* adalah untuk menyederhanakan bentuk citra.

$$T(x,y) = m(x,y) * (1 + k * (\frac{s(x,y)}{R} - 1)) \quad (2,3)$$

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

$$s(x,y) = \sqrt{\frac{\sum_{i=min}^{i=max} \sum_{j=min}^{j=max} (img(i,j)-m(x,y))^2}{(i*j)-1}} \quad (2.5)$$

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.

$$f(x,y) = \begin{cases} 0, & \text{img}(x,y) < \text{T(x,y)} \\ 255, & \text{img}(x,y) \geq \text{T(x,y)} \end{cases} \quad (2.6)$$

Keterangan :

f : Fungsi yang menghasilkan nilai 0 atau 255

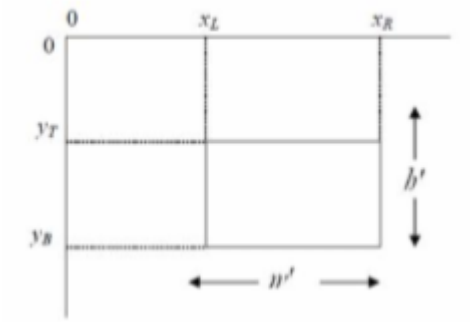
img : Nilai grayscale citra

2.2.1.4 Cropping Citra

Cropping adalah memotong satu bagian dari citra sehingga diperoleh citra yang berukuran lebih kecil. Proses ini bertujuan untuk memisahkan objek yang satu dengan objek yang lain dalam suatu gambar untuk mempercepat proses selanjutnya. Rumus yang digunakan untuk menggunakan operasi ini adalah sebagai berikut:

$$x' = x - x_L \quad \text{untuk} \quad x = x_L \quad sampai \quad x_R$$
$$y' = y - y_T \quad \text{untuk} \quad y = y_T \quad sampai \quad y_B \qquad (2,7)$$

(xL,yT) dan (xR,yB) masing-masing adalah koordinat titik pojok kiri atas dan pojok
kanan bawah bagian citra yang hendak dicrop (Gambar 2.1)



Gambar 2. 1 Koordinat titik pojok bagian citra yang akan dicrop

Dengan h'(high) merupakan tinggi citra dan w' (width) merupakan lebar citra. Ukuran citra berubah menjadi :

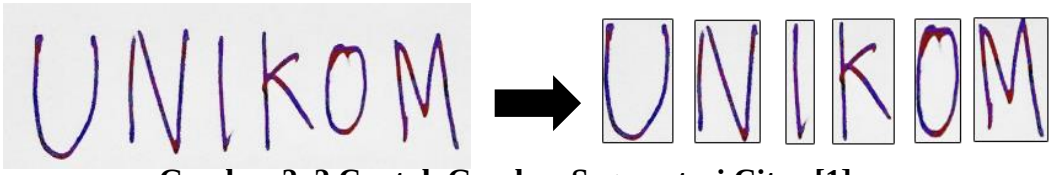
$$w' = x_R - x_L$$
$$h' = y_b - y_T \qquad (2.8)$$

Dan transformasi baliknya adalah :

$$x = x' + x_L \quad \text{untuk} \quad x' = 0 \quad sampai \quad w' - 1$$
$$y' = y - y_T \quad \text{untuk} \quad y' = 0 \quad sampai \quad h' - 1$$

2.2.1.5 Segmentasi Citra

Segmentasi citra adalah teknik untuk membagi citra menjadi beberapa daerah atau region dimana setiap daerah memiliki kemiripan atribut[11]. 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 2.2.



Gambar 2. 2 Contoh Gambar Segmentasi Citra [1].

2.2.1.6 Resize/Normalisasi

Resize citra artinya mengubah besarnya ukuran citra digital dalam pixel. Adakala ukurannya berubah menjadi lebih kecil dari ukuran aslinya dan ada kalanya sebaliknya. Proses Resize citra dapat mempengaruhi size citra lebih besar atau lebih kecil dan juga kualitas citra menjadi lebih baik atau lebih buruk. Resize/Normalisasi dapat dilihat pada Gambar 2.3.

Gambar 2. 3 Resize/Normalisasi [1]

Contoh resize di atas mengubah citra yang awalnya memiliki ukuran 35x32 pixel menjadi 20x40 pixel.

2.3 Feature Extraction

Feature Extraction adalah proses transformasi data masukan menjadi kumpulan fitur untuk mengambil representasi minimal dari data masukan. Feature Extraction merupakan proses mengambil ciri-ciri yang terdapat pada objek di dalam citra. Pada proses ini objek di dalam citra perlu dideteksi seluruh tepinya, lalu menghitung properti-properti objek yang berkaitan sebagai ciri [12].

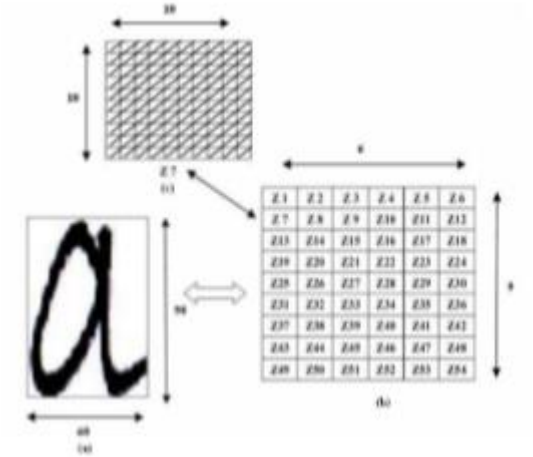
Karakteristik fitur yang baik sebisa mungkin memenuhi persyaratan sebagai berikut[12] :

a. Dapat membedakan suatu objek dengan yang lainnya.
b. Kompleksitas komputasi yang tidak terlalu rumit
c. Tidak terikat invarian terhadap transformasi
d. Jumlahnya sedikit

2.3.1 Diagonal Based Feature Extraction

Diagonal Based Feature Extraction adalah algoritma ekstraksi ciri yang membagi ukuran piksel gambar menjadi piksel-piksel yang lebih kecil dan sama rata. Misal character image berukuran 100x100 pixels dibagi menjadi 100 zona yang sama rata. Setiap zona berukuran 10x10 pixels (Gambar 2.3). Ciri diextraksi dari tiap zona dengan bergerak diagonal dari masing-masing 10x10 pixels. Tiap zona memiliki 19 garis diagonal dan foreground pixels yang ada di setiap baris diagonal dijumlahkan untuk mendapatkan satu sub-ciri. 19 nilai sub-ciri ini akan dirataratakan untuk mendapatkan nilai ciri tunggal dan ditempatkan di zona yang sesuai (Gambar 2.3). Prosedur ini diulangi untuk semua zona [6].

Akan ada beberapa zona yang diagonalnya kosong dari foreground pixels. Nilai ciri untuk zona tersebut adalah nol. 100 ciri telah di ekstraksi utuk masing-masing karakter. Selain itu, 100 ciri didapatkan dari merata-ratakan nilai yang ditempatkan pada tiap zona baris dan kolom [6].



Gambar 2. 4 Diagonal Feature Extraction [6].

2.4 Machine Learning

Machine learning, cabang dari kecerdasan buatan, adalah disiplin ilmu yang mencakup perancangan dan pengembangan algoritma yang memungkinkan computer untuk mengembangkan perilaku yang didasarkan pada data empiris, seperti dari sensor data basis data. System pembelajar dapat memanfaatkan contoh

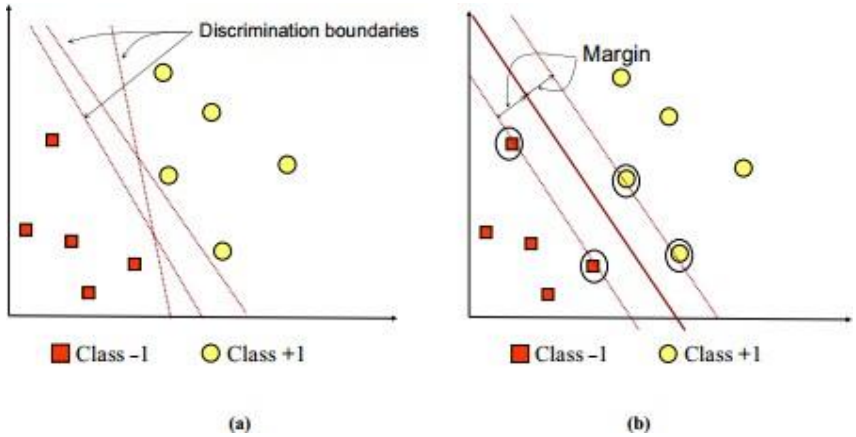
(data) untuk menangkap ciri yang diperlukan dari probalitas yang mendasarinya (yang tidak diketahui).

Data dapat dilihat sebagai contoh yang mengambarkan hubungan antara variable yang diamati. Focus besar penelitian pemelajaran mesin adalah bagaimana mengenali secara otomatis pola kompleks dan membuat keputusan cerdas berdasarkan data. Kesukarannya terjadi karena himpunan semua perilaku yang mungkin, dari semua masukan yang dimungkinkan, terlalu besar untuk diliput oleh himpunan contoh pengamatan (data pelatihan). Karena itu pembelajar harus merampatkan (generalisasi) perilaku dari contoh yang ada untuk menghasilkan keluaran yang berguna dalam kasus-kasus baru [13].

2.4.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) diperkenalkan oleh Vapnik pada tahun 1992 sebagai suatu teknik klasifikasi yang efisien untuk masalah nonlinier. SVM berbeda dengan teknik klasifikasi di era 1980-an, seperti *decision tree* dan ANN, yang secara konsep kurang begitu jelas dan seringkali terjebak pada optimum lokal. SVM memiliki konsep yang jauh lebih matang, lebih jelas secara matematis, dibanding teknik – teknik klasifikasi sebelum era 1990-an. SVM berusaha menemukan *hyperplane* dengan memaksimalkan jarak antar kelas. Dengan cara ini, SVM dapat menjamin kemampuan generalisasi yang tinggi untuk data – data yang akan datang [13].

Menurut penelitian yang dilakukan oleh Anto Satriyo Nugroho, Arief Budi Witarto, dan Dwi Handoko [13] konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah *class* pada *input space*. Gambar 2.5-a memperlihatkan beberapa *pattern* yang merupakan anggota dari dua buah class : +1 dan –1. *Pattern* yang tergabung pada *class* –1 disimbolkan dengan warna merah (kotak), sedangkan *pattern* pada *class* +1, disimbolkan dengan warna kuning(lingkaran).



Gambar 2. 5 SVM mencari Hyperplane terbaik

Masalah klasifikasi dapat diterjemahkan dengan usaha menemukan garis (*hyperplane*) yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif garis pemisah (*discrimination boundaries*) ditunjukkan pada Gambar 2.5-a. *Hyperplane* pemisah terbaik antara kedua *class* dapat ditemukan dengan mengukur *margin hyperplane* tsb. dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* tersebut dengan pattern terdekat dari masing-masing *class*. *Pattern* yang paling dekat ini disebut sebagai *support vector*.

Garis solid pada Gambar 2.5-b menunjukkan *hyperplane* yang terbaik yaitu yang terletak tepat pada tengah-tengah kedua *class*, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM. Data yang tersedia dinotasikan sebagai $\vec{x_i} \in \Re^d$ sedangkan label masing-masing dinotasikan yi ∈ {-1,+1}untuk i = 1,2,...n ,yang mana n adalah banyaknya data. Diasumsikan kedua *class* –1 dan +1 dapat terpisah secara sempurna oleh *hyperplane* berdimensi *d* , yang didefinisikan sebagai berikut:

$$\vec{w} \,.\, x + b = 0 \qquad (2.8)$$

Pattern $\vec{x_i}$ yang termasuk *class* -1 (sampel negatif) dapat dirumuskan sebagai:

$$\vec{w} \,.\, x + b \leq -1 \qquad (2.9)$$

Sedangkan *pattern* $\vec{x_i}$ yang termasuk *class* +1 (sampel positif) adalah sebagai berikut:

$$\vec{w} \cdot x + b \geq +1 \qquad (2.10)$$

Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara *hyperplane* dan titik terdekatnya, yaitu $1 / \|\vec{w}\|$. Hal ini dapat dirumuskan sebagai *Quadratic Programming (QP) problem*, yaitu mencari titik minimal persamaan (2.11), dengan memperhatikan *constraint* persamaan (2.12)

$$\min \tau(w) = \frac{1}{2} \|\vec{w}\|^2 \qquad (2.11)$$

$$w^{\rightarrow\rightarrow}$$

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0, \forall i \qquad (2.12)$$

Problem ini dapat dipecahkan dengan berbagai teknik komputasi, di antaranya *Lagrange Multiplier*

$$L(w^{\rightarrow\rightarrow}, b, \alpha) = 1/2 \|w^{\rightarrow}\|^2 - \Sigma \alpha i (yi((x\ i\ .w^{\rightarrow\rightarrow} + b) - 1)), i = 1,2 \ldots lli = 1 \qquad (2.13)$$

α_i adalah *Langrange multipliers*, yang bernilai nol atau positif ($\alpha_i \geq 0$). Nilai optimal dari persamaan (2.12) dapat dihitung dengan meminimalkan L terhadap $\vec{w}$ dan b, dan memaksimalkan L terhadap α_i. Dengan memperhatikan sifat bahwa pada titik optimal *gradient* L = 0, persamaan (2.12) dapat dimodifikasi sebagai maksimalisasi *problem* yang hanya mengandung α_i, sebagaimana persamaan (2.14) dibawah.

Maximize :

$$\Sigma \alpha i - 1/2\ \Sigma \alpha i \alpha j yi yj x\ i.x.j \qquad (2.14)$$

Dimana $\alpha i \geq 0$ (=1,2,….l) dan $\Sigma \alpha i yi = 0 li = 1$. Dari hasil perhitungan ini diperoleh αi yang kebanyakan bernilai positif. Data yang berkorelasi dengan αi yang positif inilah yang disebut *support vector*.

Untuk mendapatkan nilai αi, langkah pertama adalah mengubah setiap *feature* menjadi nilai vektor (*support vector*) = xy. Kemudian vektor akan dimasukkan kedalam persamaan *kernel trick phi* φ yaitu sebagai berikut:

$$\varphi\begin{bmatrix}x\\y\end{bmatrix} = \begin{Bmatrix}\sqrt{Xn^2+Yn^2} > 2, \begin{bmatrix}4-x+ & |x-y|\\ 4-x+ & |x-y|\end{bmatrix}\\ \sqrt{Xn^2+Yn^2} \le 2, maka\ \begin{bmatrix}x\\y\end{bmatrix}\end{Bmatrix} \quad (2.15)$$

Kemudian untuk mencari nilai αi didapatkan dari persamaan sebagai berikut:

$$\sum_{i=1,=1}^{n} a_i T^T T_j \quad (2.16)$$

Dan selanjutnya menggunakan persamaan sebagai berikut:

$$W = \sum_{i=1}^{n} a_i T_i \quad \text{dan} \quad \sum_{i=1,=1}^{n} a_i T_i = y_i \quad (2.17)$$

Terakhir akan dicari nilai w dan b untuk menemukan *hyperplane* sebagai patokan proses klasifikasi dengan persamaan sebagai berikut:

$y = wx + ,= \Sigma a\ i\ s\ i$ (2.18)

Nilai s_i merupakan nilai *support vector* yang telah dihitung sebelumnya. Dengan demikian proses klasifikasi selesai dengan memperhatikan *hyperplane* nya.

2.4.1.1 Kernel Trick

Feature space dalam prakteknya biasanya memiliki dimensi yang lebih tinggi dari vektor input (*input space*). Hal ini mengakibatkan komputasi pada *feature space* mungkin sangat besar, karena ada kemungkinan *feature space* dapat memiliki jumlah *feature* yang tidak terhingga. Selain itu, sulit mengetahui fungsi transformasi yang tepat. Untuk mengatasi masalah ini, pada SVM digunakan ”*kernel trick*”. Fungsi kernel yang umum digunakan adalah sebagai berikut [15].

1. *Kernel Linear*

   $(x_i, x) = x^T x$ (2.19)

2. *Polynomial kernel*

$K(x_i,x) = (\gamma . x^T x + r)^p, \gamma > 0$ (2.20)

3. *Radial basis function (RBF)*

   $K(x_i,x) = \exp(-\gamma |x_i - x|^2), \gamma > 0$ (2.21)

4. *Sigmoid kernel*

   $K(x_i,x) = \tanh(\gamma x^T x + r)$.(2.22)

2.4.2 Smooth Support Vector Machine (SSVM)

SSVM adalah pengembangan SVM dengan menggunakan teknik smoothing. Metode ini pertama kali diperkenalkan oleh Lee pada tahun 2001. SVM memanfaatkan optimasi dengan quadratic programming, sehingga untuk data berdimensi tinggi dan data jumlah besar SVM menjadi kurang efisien. Oleh karena itu dikembangkan smoothing technique yang menggantikan plus function SVM dengan integral dari fungsi sigmoid neural network yang selanjutnya dikenal dengan *Smooth Support Vector Machine* (SSVM).

Diberikan masalah klasifikasi dari *n* objek dalam ruang dimensi R^p sehingga susunan data berupa matriks *A* berukuran *n x p* dan keanggotaan tiap titik terhadap kelas {+1} atau {-1} yang didefinisikan pada diagonal matriks D berukuran ,problem optimasinya adalah

$min_{w,b,\xi} \ \frac{c}{2}\xi'\xi + \frac{1}{2}(w'w + b^2)$ (2.23)

dengan kendala

$D\ (Aw + eb) + \ \xi \ \geq e, \xi \ \geq 0$. (2.24)

Solusi problem adalah

$\xi \ = \big(e - D(Aw + eb)\big)$ (2.25)

Dimana ξ adalah variabel *slack* yang mengukur kesalahan klasifikasi. Kemudian dilakukan substitusi dan konversi, sehingga persamaan dapat ditulis sebagai berikut:

$min_{w,b} \ \frac{c}{2}||(e - D(Aw - eb)||_2^2 + \frac{1}{2}(w'w + b^2)$ (2.26)

Fungsi objektif dalam persamaan tidak memiliki turunan kedua. Teknik *smoothing* yang diusulkan dilakukan dengan mengganti fungsi plus dengan p(x,a)

yaitu integral dari fungsi sigmoid *neural network* atau dapat dituliskan sebagai berikut:

$$p\,(x,a) = x + \frac{1}{a}\log(1 + \varepsilon^{-ax}), a > 0 \qquad (2.27)$$

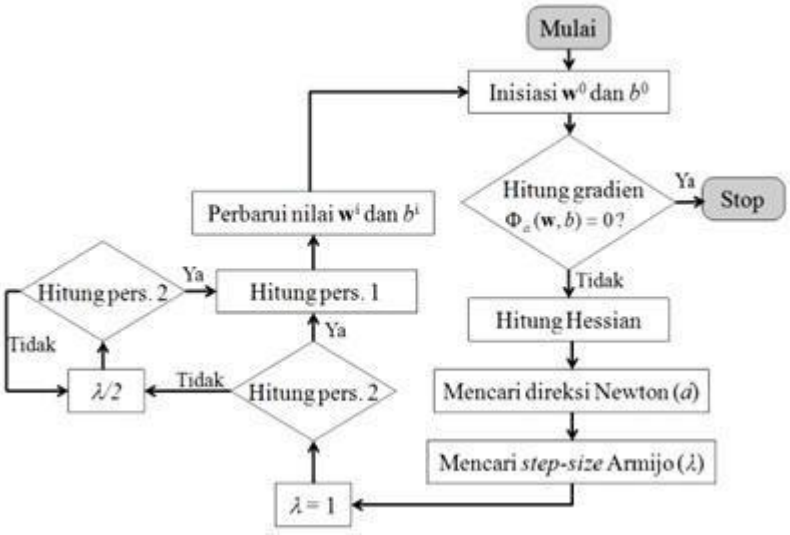
dimana α adalah parameter *smoothing*. Dengan menggantikan fungsi plus dengan p (x,a) maka diperoleh model SSVM sebagai berikut:

$$min_{(w,b)\in Rp+1}\phi a(w,b) = min_{(w,b)\in Rp+1}\ \frac{c}{2}||p(e - D(Aw - eb)||_2^2 + \frac{1}{2}\,(w'w + b^2) \qquad (2.28)$$

Secara umum, problem optimasi SSVM dapat ditulis sebagai berikut:

$$min_{(w,b)\in Rp+1}\phi a(w,b) = min_{(w,b)\in Rp+1}\ \frac{c}{2}||p(e - D\big(K\big(x_i,x_j\big)Dw - eb\big),a)||_2^2 + \frac{1}{2}\,(w'w + b^2) \qquad (2.29)$$

Yang diselesaikan dengan iterasi Newton Armijo (Gambar 2.6) dan *K(Xi,Xj)* merupakan fungsi kernel yang dalam penelitian ini digunakan kernel Gaussian atau bisa dirumuskan berikut *K(Xi,Xj) = exp (-y(*||Xi – Xj||^2) dengan parameter kernel *y*.



Gambar 2. 6 Diagram Alir Algoritma Newton-Armijo

Persamaan 1 :

$$\phi_a(w_i,b_i) - \phi_a((w_i,b_i) + (\lambda_i d_i) \geq\ -\delta\lambda_i\nabla\phi_a(w_i,b_i)d_i \qquad (2.30)$$

Persamaan 2 :

$$w_{i+1,}b_{i+1} = (w_i,b_i) +\ (\lambda_i d_i) \qquad (2.31)$$

Saat iterasi pada algoritma Newton-Armijo berhenti, diperoleh nilai *w* dan *b* yang konvergen. Dengan demikian fungsi pemisah yang diperoleh untuk kasus klasifikasi linier adalah

F(x) = *sign* (w'x + b) (2.32)

Sedangkan fungsi permisah untuk kasus klasifikasi nonlinier adalah sebagai berikut

F(x) = *sign* (w'x + b) = *sign* (u'*D*'*K*(Xi,Xj) + b) (2.33)

Perumusan program linier SVM 1-norm adalah salah satu cara untuk memilih atribut (*feature selection*) di antara varian-varian norm SVM, problem linier tersebut adalah sebagai berikut:

$$\min_{(w,b,s,\xi)\in R^{(2p)+1+n}} Ce'\xi + e's \qquad (2.34)$$

Dengan kendala

$$D(Aw + eb) + \xi \geq e \qquad (2.35)$$

$$-s \leq w \leq s$$

$$\xi \geq 0 \qquad (2.36)$$

Solusi dari ***w*** mampu menghasilkan model yang parsimoni dan bersifat *sparsity*. Jika nilai dari elemen vektor ***wp*** = 0, maka variabel *p* tidak berkontribusi dalam penentuan kelas. Kontribusi atribut atau variabel prediktor dapat dinilai dari besarnya nilai ***wi*** untuk masing-masing atribut, dengan *l*=1,2, ..., *p*.

Support Vector Machine (SVM) adalah suatu sistem pembelajaran yang menggunakan ruang hipotesis dari suatu fungsi linear dalam suatu ruang dimensi berfitur tinggi yang dikembangkan oleh Boser, Guyon, Vapnik, dan pertama kali dipresentasikan pada tahun 1992 di Annual Workshop on Computational Learning Theory.

2.5 Pengujian Sistem

Pengujian sistem adalah proses pemeriksaan atau evaluasi sistem atau komponen sistem secaa manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan-kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaaan-

perbedaan antara hasil yang diterapkan dengan hasil yang terjadi. 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, 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*.

2.5.1 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*.

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.

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.

2.5.2 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.

$$Akurasi = \frac{Jumlah\ Karakter\ Sama}{Jumlah\ Seluruh\ Karakter}\ X\ 100\% \qquad (2.42)$$

2.6. 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.

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*). 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, 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.

2.7. UML (Unified Modeling Language)

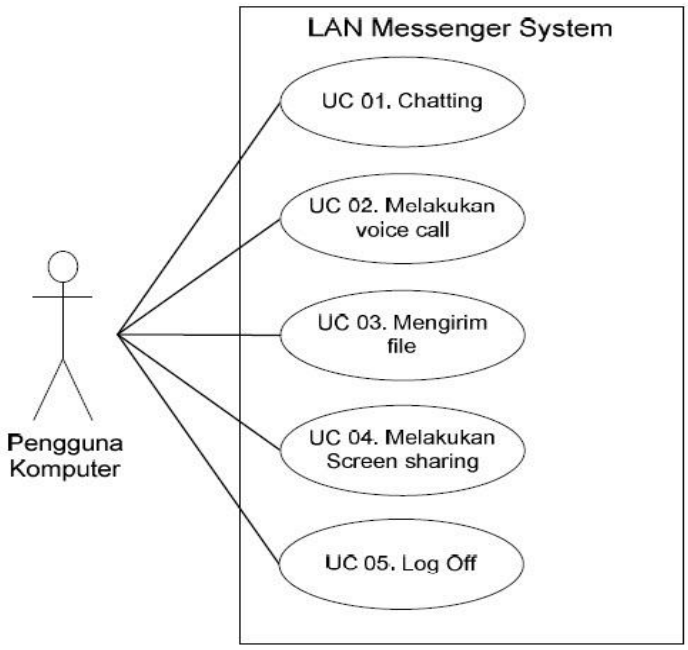
UML *(Uniefied Modeling Language)* adalah notasi yang lengkap untuk membuat visualisasi model suatu sistem. Sistem berisi informasi dan fungsi, tetapi secara normal digunakan untuk memodelkan sistem komputer. Di dalam pemodelan objek guna menyajikan sistem yang berorientasi objek kepada orang lain, akan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

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.*

2.7.1 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.

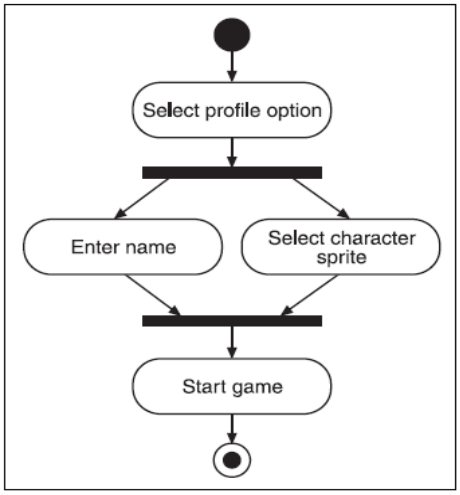
Use case diagram merupakan deskripsi lengkap tentang interaksi yang terjadi antara para aktor dengan sistem. Dalam hal ini, setiap objek yang berinteraksi dengan sistem merupakan aktor untuk sistem, sementara *use case* merupakan deskripsi lengkap tentang bagaimana sistem berprilaku kepada aktornya. 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.



Gambar 2. 7 Use Case Diagram

2.7.2 Activity Diagram

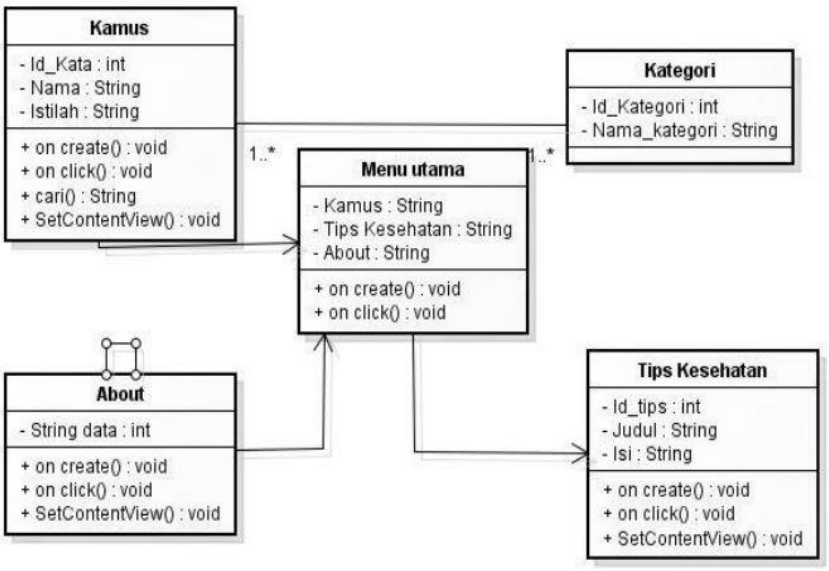
Activity Diagram pada dasarnya menggambarkan skenario secara grafis. A*ctivity diagram* cukup serupa dengan diagram alur (*flow chart),* yang membedakan hanya *swimlane* yang menunjukkan suatu *state* berada pada objek/kelas tertentu. Keunggulan dari *activity diagram* adalah lebih mudah dipahami dibanding skenario. Selain itu, dengan menggunakan *activity diagram,* kita dapat melihat dibagian manakah sistem dari suatu skenario akan berjalan.



Gambar 2. 8 Activity Diagram

2.7.3 Class Diagram

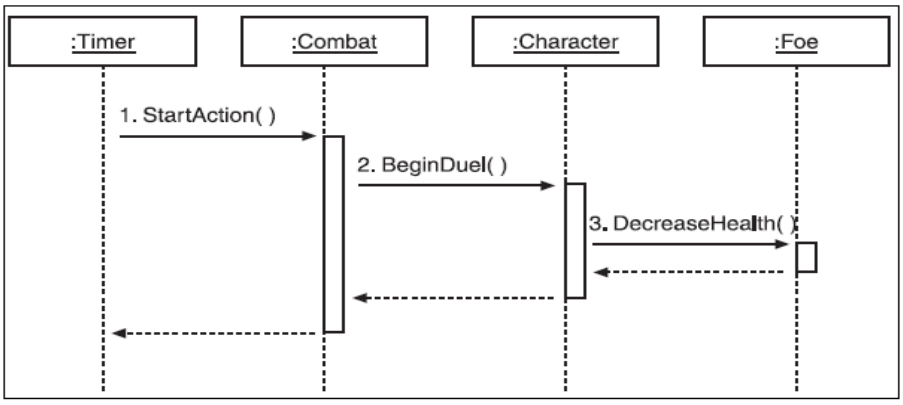
Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendeinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class diagram* menggambarkan struktur dan deskripsi kelas, package, dan objek beserta hubungan satu sama lain seperti *containment,* pewarisan, asosiasi dan sebagainya.



Gambar 2. 9 Class Diagram

2.7.4 Sequence Diagram

Diagram sekuen menggambarkan kelakuan/perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. 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. 10 Sequence Diagram