BAB 2
TINJAUAN PUSTAKA

2.1 Optical Character Recognition (OCR)

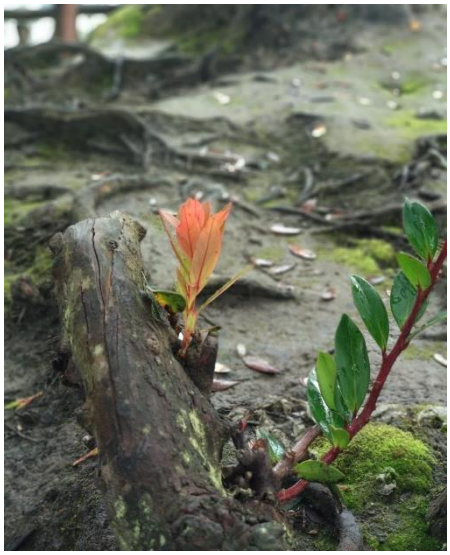
OCR merupakan solusi yang efektif untuk proses konversi dari dokumen cetak ke dalam bentuk dokumen digital. Sistem pengenal huruf ini dapat meningkatkan fleksibilitas atau kemampuan dan kecerdasan komputer. Sistem pengenal huruf yang cerdas sangat membantu usaha digitalisasi informasi dan pengetahuan, misalnya dalam pembuatan koleksi pustaka digital, koleksi sastra kuno, dan lain-lain [9].

2.2 Karya Tulis Ilmiah Skripsi

Salah satu jenis karya tulis tulis ilmiah yaitu skripsi. Skripsi adalah istilah yang digunakan di Indonesia untuk mengilustrasikan suatu karya tulis ilmiah berupa paparan tulisan hasil penelitian sarjana S1 yang membahas suatu permasalahan/fenomena dalam bidang ilmu tertentu dengan menggunakan kaidah-kaidah yang berlaku. Skripsi ditulis berdasarkan kajian pada studi pustaka, penyelidikan, observasi, atau penelitian lapangan. Penelitian yang akan dibuat ini menggunakan hasil *scan* beberapa bagian dari karya tulis ilmiah skripsi dalam bentuk gambar/citra untuk dikenali setiap karakter yang terdapat didalamnya.

2.3 Citra

Citra terbagi menjadi dua jenis, yaitu citra analog dan citra digital, namun dalam studi tentang pengolahan citra, citra yang biasa digunakan adalah citra digital yang merupakan hasil dari alat elektronik yang dapat menangkap gambar. Citra digital dibentuk oleh kumpulan titik yang dinamakan piksel (*pixel* atau "*picture element*"). Setiap piksel digambarkan sebagai satu kotak kecil. Dengan sistem koordinat yang mengikuti asas pemindaian pada layar TV standar, sebuah piksel mempunyai koordinat berupa (i, j), dimana i menyatakan posisi kolom, dan j menyatakan posisi baris. Piksel pojok kiri-atas mempunyai koordinat (0,0) dan piksel pada pojok kanan mempunyai koordinat (i-1, j-1) [10]. Salah satu contoh citra digital dapat dilihat pada Gambar 2.1.



Gambar 2.1 Contoh Citra Digital

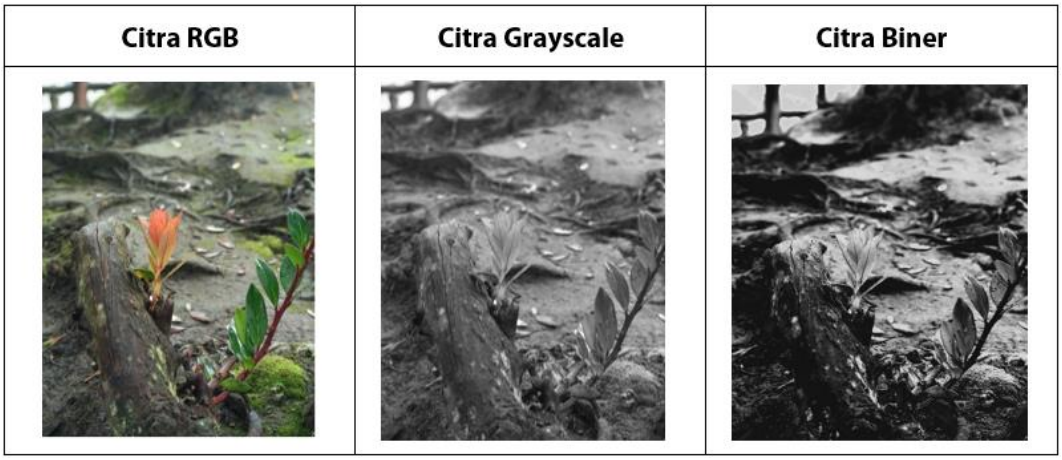
Pada penelitian ini, citra berfungsi sebagai data masukan yang akan berpengaruh terhadap proses dan hasil akhir. Maka dari itu, kualitas citra masukan sangat menentukan tingkat akurasi pada proses pengujian. Oleh karena itu, dilakukan proses pengolahan citra untuk memperoleh kualitas citra yang baik.

2.3.1 Jenis-jenis Citra

Jenis-jenis citra yang biasa digunakan diantaranya yaitu [11]:

a. Citra Berwarna: Citra yang memiliki 3 buah kanal warna didalamnya, pada umumnya terbentuk dari komponen merah/*red* (R), hijau/*green* (G), dan biru/*blue* (B) yang dimodelkan kedalam ruang warna RGB. RGB adalah standar yang digunakan untuk menampilkan citra berwarna pada layar televisi maupun layar komputer.
b. Citra *Grayscale*: Citra yang hanya memiliki 1 buah kanal sehingga yang ditampilkan hanyalah nilai intensitas atau dikenal juga dengan istilah derajat keabuan. Jenis citra ini disebut juga sebagai 8-bit *image* karena untuk setiap nilai pikselnya memerlukan penyimpanan sebesar 8-bit.
c. Citra Biner: Citra yang hanya memiliki 2 kemungkinan nilai untuk setiap pikselnya, yaitu 0 atau 1. Nilai 0 akan tampil sebagai warna hitam sedangkan nilai 1 akan tampil sebagai warna putih.

Penelitian ini akan menggunakan ketiga jenis citra yang telah disebutkan melalui proses pengolahan citra. Perbedaan ketiga jenis citra tersebut dapat dilihat pada Gambar 2.2.



Gambar 2.2 Perbedaan Citra RGB, Grayscale, dan Biner

2.3.2 Pengolahan Citra Digital

Pengolahan citra adalah istilah umum untuk berbagai teknik yang keberadaanya untuk memanipulasi dan memodifikasi citra dengan berbagai cara. Foto adalah contoh gambar berdimensi dua yang dapat diolah dengan mudah. Setiap foto dalam bentuk citra digital (misalnya berasal dari kamera digital) dapat diolah melalui perangkat lunak tertentu [10]. Pada penelitian ini pengolahan citra dilakukan untuk memproses citra masukan sehingga kualitas citra masukan menjadi lebih baik sebelum masuk ke proses berikutnya.

2.4 *Preprocessing* Citra

Preprocessing merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak perlu pada gambar. *Preprocessing* adalah tahap pertama yang harus dilakukan sebelum proses utama dari pengenalan karakter dilakukan. Tahap ini sangat penting untuk menentukan keberhasilan suatu proses pengenalan pola [12]. Inti dari tahap *preprocessing* adalah mengubah hasil gambar *scan* dokumen cetak menjadi kumpulan karakter yang akan dikenali dalam proses OCR [13]. Tahap *preprocessing* yang dilakukan dalam penelitian ini yaitu konversi citra RGB ke

grayscale, *thresholding*, segmentasi citra menggunakan *profile projection*, binerisasi, *resize*, serta ekstraksi fitur zoning.

2.4.1 Konversi Citra ke *Grayscale*

Tahapan awal dalam *preprocessing* pada penelitian ini adalah konversi citra RGB ke citra *grayscale*. *Grayscale* adalah istilah untuk menyebutkan satu citra yang memiliki warna abu-abu, hitam, dan putih. Pada jenis gambar ini, warna dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih [10]. Karena jenis citra ini hanya memiliki 1 kanal saja, maka citra *grayscale* memiliki tempat penyimpanan yang lebih hemat. Jenis citra ini disebut juga sebagai citra 8-bit [11]. Pada penelitian ini menggunakan metode *grayscale* dengan Persamaan 2.1 sebagai berikut.

$$GS_{(i,j)} = (0{,}299 * R_{(i,j)}) + (0{,}587 * G_{(i,j)}) + (0{,}114 * B_{(i,j)}) \quad (2.1)$$

Keterangan :

$GS_{(i,j)}$ = Citra *grayscale*

$R_{(i,j)}$ = Nilai dari piksel berwarna merah (*Red*)

$G_{(i,j)}$ = Nilai dari piksel berwarna hijau (*Green*)

$B_{(i,j)}$ = Nilai dari piksel berwarna biru (*Blue*)

2.4.2 *Thresholding*

Thresholding yaitu mengubah citra *grayscale* menjadi citra berwarna hitam putih dengan nilai ambang yang sudah ditentukan. Terdapat dua metode *thresholding*, yaitu *global thresholding* dan *local thresholding* [10]. Apabila nilai ambang (*threshold*) bergantung hanya pada satu nilai aras keabuan, pengambangan disebut global, dimana semua piksel dalam citra akan ditentukan oleh satu nilai ambang yang sudah ditentukan. Pengambangan ini biasa digunakan untuk memisahkan tulisan hitam yang berada di atas secarik kertas putih. Namun, jika nilai ambang bergantung pada beberapa nilai sesuai dengan aras keabuan pada citra maka disebut pengambangan lokal. Dimana nilai ambang untuk setiap piksel ditentukan oleh nilai piksel tetangga, maka nilai ambang untuk masing-masing

piksel belum tentu sama. Perbedaan ciri kedua jenis *thresholding* ini yaitu sebagai berikut.

a. Ciri-ciri *global thresholding*

Ciri-ciri dari *global thresholding* yaitu [10]:

1) Tidak memperhatikan hubungan spasial antar piksel.
2) Sensitif terhadap pencahayaan tidak seragam.
3) Hanya berlaku untuk keadaan ideal (misalnya, latar belakang berwarna putih dan objek berwarna hitam).
4) Bergantung kepada pemilihan nilai ambang yang tepat.

b. Ciri-ciri *local thresholding*

Ciri-ciri dari *local thresholding* yaitu [10]:

1) Memperhatikan hubungan spasial antarpiksel.
2) Mampu beradaptasi dengan pencahayaan yang tidak seragam.
3) Berlaku untuk keadaan ideal atau tidak.

Berdasarkan ciri-ciri yang telah disebutkan, maka dalam penelitian ini akan menggunakan metode *global thresholding*, yang dilakukan dengan cara mengubah nilai citra *grayscale* menjadi nilai 0 atau 255 berdasarkan nilai *threshold* yang telah ditentukan sebelumnya. Metode ini digunakan karena data latih dan data uji yang digunakan merupakan citra yang memiliki latar belakang berwarna putih dan objek berwarna hitam, serta memiliki tingkat pencahayaan yang seragam.

Pemilihan nilai ambang yang tepat sangat berpengaruh pada *global thresholding.* Karena hal tersebut bisa mempengaruhi berhasil atau tidaknya proses *thresholding* pada citra. Berdasarkan penelitian sebelumnya [14], terdapat 5 nilai ambang batas di dalam pengujiannya, yaitu nilai tengah, nilai minimum, nilai maksimum, nilai modulus, dan nilai *minimax between*. Nilai tengah adalah 128, nilai minimum diambil dari nilai tengah dibagi 2 sehingga nilai minimum adalah 64, nilai maksimum adalah 128 ditambah dengan 64 sehingga hasilnya adalah 192, nilai modulus yaitu jika nilai *grayscale* dimodulus dengan 2 = 1 maka akan diubah menjadi 255 sedangkan bila hasilnya sama dengan 0 maka tidak diubah, sementara nilai *minimax between* hampir sama dengan nilai modulus tetapi jika nilai *grayscale* < 64 maka diubah menjadi 0, dan jika > 192 maka diubah menjadi 255. Penelitian

tersebut menyatakan bahwa nilai tengah mempunyai hasil warna yang lebih jelas ketika dibandingkan dengan nilai *threshold* lainnya [14].

Oleh karena itu, nilai ambang yang digunakan pada penelitian ini yaitu sebesar 128. Nilai tersebut didapatkan dari total intensitas derajat keabuan dibagi 2. Dimana nilai total intensitas derajat keabuan yaitu sebesar 255 dibagi 2, hasilnya adalah 127.5 dibulatkan menjadi 128. Apabila nilai citra *grayscale* kurang dari 128, maka nilainya akan diubah menjadi 0 (hitam/objek), dan apabila nilai citra *grayscale* lebih dari 128, maka nilainya akan diubah menjadi 255 (putih/*background*). Seperti pada persamaan berikut.

$$T_{(i,j)} = \begin{cases} 255, GS_{(i,j)} \geq 128 \\ 0, GS_{(i,j)} < 128 \end{cases} \quad (2.2)$$

Keterangan :

$T_{(i,j)}$: Nilai *threshold*

$GS_{(i,j)}$: Nilai citra *grayscale* suatu piksel

2.4.3 Segmentasi Citra Menggunakan *Profile Projection*

Metode *Profile Projection* digunakan untuk memisahkan tulisan per baris, per kata dan per karakter. Berikut ini masing-masing penjelasan proses segmentasi.

2.4.3.1 Segmentasi Baris

Proses segmentasi baris bertujuan untuk memisahkan baris-baris pada citra yang mengandung karakter. Segmentasi baris dilakukan secara horizontal (terhadap sumbu j) dengan cara menjumlahkan nilai piksel hitam pada setiap baris. Jika ditemukan jumlah piksel hitam pada suatu baris <5 atau = 0 maka akan dijadikan acuan untuk pemotongan per baris[15]. Baris yang sudah ditemukan selanjutnya akan dipotong dan disimpan untuk proses segmentasi kata. Segmentasi baris akan terus dilakukan hingga semua baris yang mengandung karakter pada citra dipotong dan disimpan.

2.4.3.2 Segmentasi Kata

Citra hasil segmentasi baris selanjutnya akan masuk ke tahap segmentasi kata untuk memisahkan setiap kata yang terdapat pada baris. Proses segmentasi ini

dilakukan secara vertikal (terhadap sumbu i). Proses yang dilakukan yaitu dengan melihat proyeksi piksel hitam secara vertikal. Namun acuan untuk pemisahannya yang agak berbeda dengan pemisahan baris, karena rentang piksel yang tidak mengandung piksel hitam (berwarna putih) untuk memisahkan kata harus lebih besar agar pemisahan benar-benar memisahkan kata bukan karakter. Acuan yang digunakan yaitu dengan cara menentukan ambang batas spasi untuk memisahkan setiap kata. Ambang batas yang ditentukan yaitu 80% dari tinggi citra baris [16] seperti menggunakan persamaan berikut.

$$S_k = 0{,}3 * t \tag{2.3}$$

Keterangan :

S_k : Nilai ambang batas spasi kata

t : Tinggi citra hasil segmentasi baris

Dimana jika ditemukan jarak kolom dengan jumlah piksel hitam = 0 berjumlah lebih dari nilai ambang spasi kata, maka kolom tersebut akan dianggap sebagai batas antar kata. Lakukan semua tahap tersebut pada setiap baris sehingga kata-kata yang terdapat pada setiap baris dapat terpisah.

2.4.3.3 Segmentasi Karakter

Proses segmentasi karakter hampir sama dengan proses segmentasi kata, hanya berbeda pada penentuan nilai ambang batas. Batas ambang yang digunakan dalam pemisahan karakter yaitu tidak menggunakan batas ambang khusus. Dimana jika ditemukan jarak kolom dengan jumlah piksel hitam = 0, maka kolom tersebut akan dianggap sebagai batas antar huruf. Lakukan semua tahap tersebut pada setiap baris sehingga huruf-huruf yang terdapat pada setiap baris dapat terpisah.

2.4.4 *Resize*

Resize adalah proses yang digunakan untuk mengubah ukuran citra digital dalam piksel, baik menjadi lebih kecil atau lebih besar dari ukuran sebenarnya. Proses resize pada penelitian ini tidak memerlukan metode khusus, caranya hanya

dengan dilakukan perbandingan ukuran antara citra hasil *thresholding* (pada tahap latih) dan hasil segmentasi (pada tahap uji) dengan menggunakan persamaan 2.4 untuk mendapatkan posisi koordinat dari titik x yang baru dan persamaan 2.5 untuk mendapatkan posisi koordinat dari titik y yang baru. Dimana kedua persamaan tersebut adalah sebagai berikut.

$$i_r = \frac{pb * pp}{pa} \tag{2.4}$$

Keterangan:

i_r = Posisi koordinat i baru

pb = Ukuran panjang dari matriks baru

pp = Posisi koordinat i lama

pa = Ukuran panjang dari matriks lama

Untuk mencari koordinat j yang baru, digunakan Persamaan 2.5 sebagai berikut.

$$j_r = \frac{lb * pp}{la} \tag{2.5}$$

Keterangan:

j_r = Posisi koordinat j baru

lb = Ukuran lebar dari matriks baru

pp = Posisi koordinat j lama

la = Ukuran lebar dari matriks lama

Pada penelitian ini *resize* digunakan untuk mengubah ukuran *pixel* sesuai dengan kebutuhan dalam melakukan proses ekstraksi ciri. Hal ini dilakukan agar semua citra karakter dari hasil segmentasi mempunyai ukuran yang sama (normalisasi ukuran citra) sebelum masuk ke tahap ekstraksi fitur.

2.4.5 Binerisasi

Setelah melakukan proses *resize,* selanjutnya yaitu melakukan binerisasi. Proses binerisasi untuk mengubah citra *hasil resize* menjadi citra biner, artinya mengubah warna tiap-tiap piksel pada citra bernilai 0 dan 255 ke dalam piksel

bernilai 0 dan 1. Sehingga citra hanya berwarna hitam dan putih. Namun dalam penelitian ini nilai biner diubah dengan cara membalik nilai, jika nilai piksel citra hasil *resize* $= 255$ maka akan diubah menjadi 0 (hitam/*background*), dan apabila nilai citra *grayscale* $= 1$ akan diubah menjadi 1 (putih/objek). Hal ini dilakukan karena nilai objek akan dilakukan untuk proses ekstraksi fitur. Persamaan 2.6 digunakan untuk proses binerisasi, jika citra yang akan diubah berasal dari proses *thresholding*.

$$B_{(i,j)} = \begin{cases} 1, T_{(i,j)} = 0 \\ 0, T_{(i,j)} = 255 \end{cases} \quad (2.6)$$

Keterangan :

$B_{(i,j)}$: Nilai biner

$T_{(i,j)}$: Nilai citra suatu piksel hasil proses *thresholding*

Jika citra yang akan diubah berasal dari proses *resize* maka pengubahan menjadi citra biner akan dilakukan dengan menggunakan Persamaan 2.7.

$$B_{(i,j)} = \begin{cases} 1, R_{(i,j)} = 0 \\ 0, R_{(i,j)} = 255 \end{cases} \quad (2.7)$$

Keterangan :

$B_{(i,j)}$: Nilai biner

$R_{(i,j)}$: Nilai citra suatu piksel hasil proses *resize*

2.4.6 Ekstraksi Fitur Zoning

Ekstraksi fitur zoning merupakan salah satu metode ekstraksi ciri dimana inti dari metode ini adalah citra dibagi pada sejumlah zona yang sama untuk dikenali ciri dari setiap karakter huruf yang selanjutnya menghasilkan sebuah nilai untuk diproses pada tahapan klasifikasi [17].

Ada beberapa algoritma untuk metode ekstraksi ciri *zoning*, diantaranya metode ekstraksi ciri jarak metrik ICZ *(image centroid zone)*, metode ekstraksi ciri jarak metrik ZCZ *(Zone centroid and zone)*. Kedua algoritma tersebut menggunakan citra digital sebagai *input* dan menghasilkan fitur untuk klasifikasi dan pengenalan

sebagai *output*-nya. Pada penelitian ini digunakan metode ekstraksi ciri jarak metrik ICZ. Berikut merupakan tahapan dalam proses ekstraksi ciri ICZ [18].

1. Hitung centroid dari citra masukan menggunakan persamaan berikut.

$$c_i = \frac{(P_{i1}.p_1 + P_{i2}.p_2 + \cdots + P_{in}.p_n)}{(p_1 + p_2 + \cdots + p_n)} \quad (2.8)$$

$$c_j = \frac{(P_{j1}.p_1 + P_{j2}.p_2 + \cdots + P_{jn}.p_n)}{(p_1 + p_2 + \cdots + p_n)} \quad (2.9)$$

Keterangan:

c_i = *centroid* koordinat i

c_i = *centroid* koordinat j

P_{in} = koordinat i dari piksel ke-n

P_{jn} = koordinat j dari piksel ke-n

p_n = nilai piksel ke-n

2. Bagi citra masukan ke dalam n zona yang sama. Contoh pembagian 3 Zona pada citra biner dapat dilihat pada Tabel 2.1.

Tabel 2.1 Contoh Pembagian Zona Pada Citra Biner

0	0	1	0	0	**Zona 1 (atas)**
0	1	0	1	0	
1	0	0	0	1	
0	1	0	0	0	**Zona 2 (Tengah)**
0	0	1	0	0	
0	0	0	1	0	
1	0	0	0	1	**Zona 3 (Bawah)**
0	1	0	1	0	
0	0	1	0	0	

3. Hitung jarak antara centroid citra dengan masing-masing piksel yang ada dalam zona.

$$d(P_{i,j}, c_{i,j}) = \sqrt{(P_i - c_i)^2 + (P_j - c_j)^2} \quad (2.10)$$

Keterangan:

d = jarak antara koordinat *centroid* (i,j) dengan koordinat objek
P_i = koordinat i objek
P_j = koordinat j objek
c_i = centroid koordinat i
c_j = centroid koordinat j

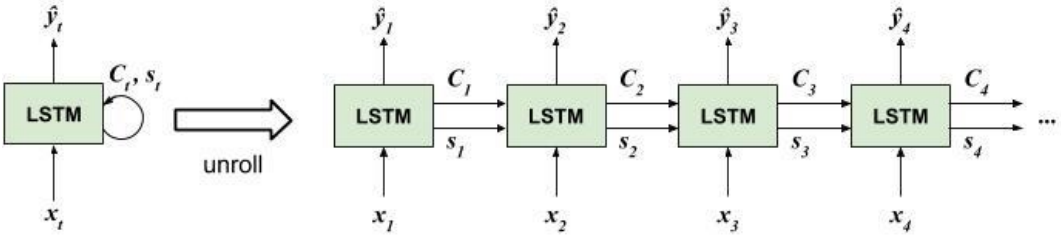
4. Ulangi langkah ke 3 untuk setiap piksel yang ada di zona
5. Hitung rata-rata jarak antara titik-titik tersebut.
6. Ulangi langkah-langkah tersebut untuk keseluruhan zona,
7. Hasilnya adalah n fitur yang akan digunakan dalam klasifikasi dan pengenalan.

2.5 *Long Short-Term Memory* (LSTM)

Long Short-Term Memory (LSTM) adalah sebuah pengembangan dari RNN (*Recurrent Neural Network*), diperkenalkan oleh Hochreiter dan Schmidhuber pada 1997. LSTM dapat dikatakan sebagai variasi dari RNN, dimana RNN merupakan bagian dari jaringan syaraf tiruan untuk pemrosesan data sekuensial. Pada RNN, memori yang tersimpan sebelumnya tidak berguna karena dengan seiringnya waktu, memori tersebut akan ditimpa dengan memori yang baru. Berbeda dengan RNN, LSTM melakukan pembelajaran jangka panjang. LSTM dapat bekerja dengan sangat baik pada berbagai macam masalah dan dapat mengingat informasi untuk jangka waktu yang lama [19].

Modul LSTM memiliki pemrosesan yang berbeda dengan modul RNN biasa. Selain itu, pada modul LSTM terdapat tambahan sinyal yang diberikan dari satu langkah waktu ke langkah waktu berikutnya, yaitu konteks, direpresentasikan dengan simbol $\boldsymbol{C}_t$ [20]. Konteks adalah sebuah vektor, yang jumlah elemennya kita tentukan sebagai desainer jaringan LSTM. Ide kunci dari LSTM adalah jalur yang menghubungkan konteks lama ($\boldsymbol{C}_{t-1}$) ke konteks baru ($\boldsymbol{C}_t$) dibagian atas modul LSTM. Dengan adanya jalur tersebut, suatu nilai dikonteks yang lama akan dengan

mudah diteruskan ke konteks yang baru dengan sedikit sekali modifikasi, jika diperlukan. Untuk lebih jelasnya, jaringan LSTM dapat dilihat pada Gambar 2.3.



Gambar 2.3 Jaringan LSTM

Arsitektur LSTM terdiri dari *input layer*, *output layer*, dan *hidden layer*. *Hidden layer* terdiri dari *memory cell*, dimana satu *memory cell* memiliki tiga *gate* yaitu *input gate*, *forget gate*, *output gate* [21]. Sebuah *memory cell* di LSTM menyimpan sebuah nilai atau keadaan (*cell state*), baik untuk periode waktu yang panjang atau singkat. Berikut adalah penjelasan untuk setiap *gate* (gerbang) yang ada pada satu *memory cell* LSTM :

a. *Input Gate* (***i_t***)

Input gate berperan mengambil *output* sebelumnya dan *input* baru serta melewatkan mereka melalui lapisan *sigmoid*. *Gate* ini mengembalikan nilai 0 atau 1. Rumus dari i_t adalah [22]:

$$i_t = \sigma(w_{xi} \,.\, x_t + w_{hi} \,.\, h_{t-1} + b_i) \tag{2.11}$$

$$\sigma = \frac{1}{1 + \exp(-x)} \tag{2.12}$$

Keterangan:

σ = Fungsi aktivasi *sigmoid*.
w_{xi} = Bobot matriks dari *input gate*.
w_{hi} = Bobot matriks dari *input gate*.
h_{t-1} = *State* sebelumnya atau *state* pada waktu t-1.
x_t = *Input* pada waktu t.
b_i = Unit bias pada *input gate*.

Nilai *input gate* dikalikan dengan *output* dari lapisan kandidat (v_t). Rumus dari (v_t) adalah [22]:

$$v_t = tanh(w_{xv} . x_t + w_{hv} . h_{t-1} + b_v) \quad (2.13)$$

$$C_t = f_t * C_{t-1} + i_t * v_t \quad (2.14)$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (2.15)$$

Keterangan:

v_t = *Intermediate cell state*.
w_{xv} = Bobot dari *intermediate cell state*.
h_{t-1} = *State* sebelumnya atau *state* pada waktu t-1.
x_t = *Input* pada waktu t.
$tanh$ = Fungsi tangen hiperbolik.

State sebelumnya dikalikan dengan forget gate dan kemudian ditambahkan ke fungsi kandidat baru yang diizinkan oleh *output gate*.

b. *Forget Gate* (f_t)

Forget gate adalah lapisan *sigmoid* yang mengambil *output* pada waktu t – 1 dan *input* pada waktu t dan menggabungkannya serta menerapkan fungsi aktivasi *sigmoid*. Karena *sigmoid, output* dari *gate* ini adalah 0 atau 1. Jika f_t = 0 maka keadaan (*state*) sebelumnya akan dilupakan, sementara jika f_t = 1 state sebelumnya tidak berubah. Rumus dari f_t adalah [22]:

$$f_t = \sigma(w_{xf} . x_t + w_{hf} . h_{t-1} + b_f) \quad (2.16)$$

Keterangan:

σ = Fungsi aktivasi *sigmoid*.
w_{xf} = Bobot matriks dari *forget gate*.
w_{hf} = Bobot matriks dari *forget gate*.
h_{t-1} = *State* sebelumnya atau *state* pada waktu t-1.
x_t = *Input* pada waktu t.
b_f = Unit bias pada *forget gate.*

Lapisan ini menerapkan tangen hiperbolik ke campuran *input* dan *output* sebelumnya. Mengembalikan vektor kandidat yang akan ditambahkan ke state.

c. *Output Gate* (o_t)

Output gate mengontrol seberapa banyak *state* yang lewat ke *output* dan bekerja dengan cara yang sama dengan *gate* lainnya, dan terakhir menghasilkan *cell state* yang baru (h_t). Rumus dari o_t dan h_t adalah [22]:

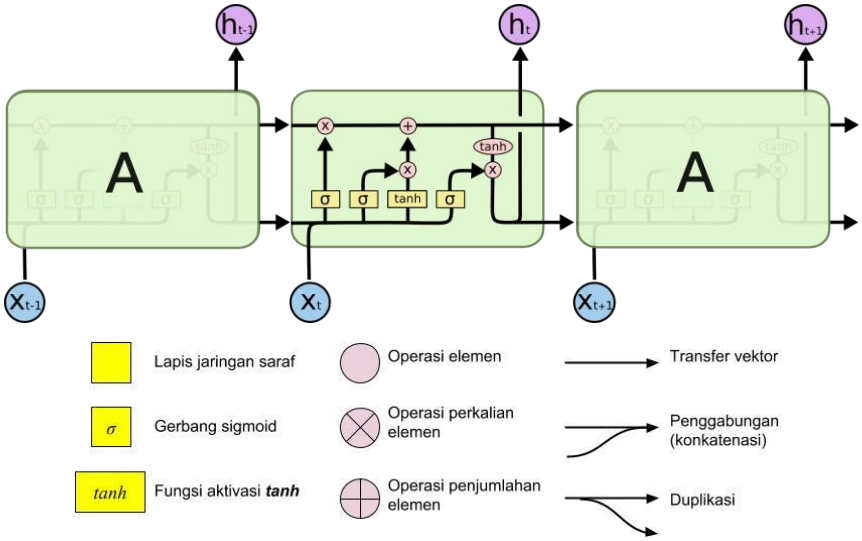
$$o_t = \sigma(w_{xo} \, . \, x_t + w_{ho} \, . \, h_{t-1} \, + b_o) \tag{2.17}$$

$$h_t = o_t * \, tanh(C_t) \tag{2.18}$$

Keterangan:

σ = Fungsi aktivasi *sigmoid*.
w_{xo} = Bobot matriks dari *output gate*.
w_{ho} = Bobot matriks dari *output gate*.
h_{t-1} = *State* sebelumnya atau *state* pada waktu t-1.
x_t = *Input* pada waktu t.
b_o = Unit bias pada *output gate*.

Untuk lebih jelasnya, arsitektur jaringan LSTM dapat dilihat pada Gambar 2.4 di bawah ini.



Gambar 2.4 Arsitektur Jaringan LSTM

2.6 Pengujian Sistem

Pada penelitian ini, proses pengujian sistem terbagi menjadi dua, yaitu pengujian fungsionalitas sistem atau pengujian *black box* dan pengujian metode untuk mengukur tingkat akurasi yang dihasilkan metode dalam mengenali karakter pada citra. Berikut ini masing-masing penjelasan dari kedua jenis pengujian.

2.6.1 Pengujian *Black Box*

Menurut Pressman [23] *black-box testing* berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineers* untuk memperoleh set kondisi *input* yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program. *Black-Box testing* berusaha untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang tidak benar atau fungsi yang hilang.
2. Kesalahan antarmuka.
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan perilaku atau kesalahan kinerja

Inisialisasi dan perumusan kesalahan.

2.6.2 Pengujian Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value* dan *reference value*). Tingkat akurasi diperoleh dengan persamaan sebagai berikut [24].

$$\text{Akurasi} = \frac{\text{Jumlah karakter yang dikenali dengan tepat}}{\text{Total semua karakter inputan}} \times 100\% \quad (2.17)$$

2.7 Bahasa Pemrograman Python

Python adalah bahasa pemrograman yang bersifat *open source*. Bahasa pemrograman ini dioptimalisasikan untuk *software quality*, *developer productivity*, *program portability*, dan *component integration*. Python telah digunakan untuk mengembangkan berbagai macam perangkat lunak, seperti *internet scripting*, *systems programming*, *user interfaces*, *product customization*, *numberic programming*, dan lain-lain [25].

2.8 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak-artifak dari sistem. UML digunakan dalam pengembangan sistem perangkat lunak yang menggunakan pendekatan berorientasi objek [26]. Adapun yang menjadi pegangan dalam pembuatan UML ini berdasarkan *Ebook* oleh Rules Miles [27].

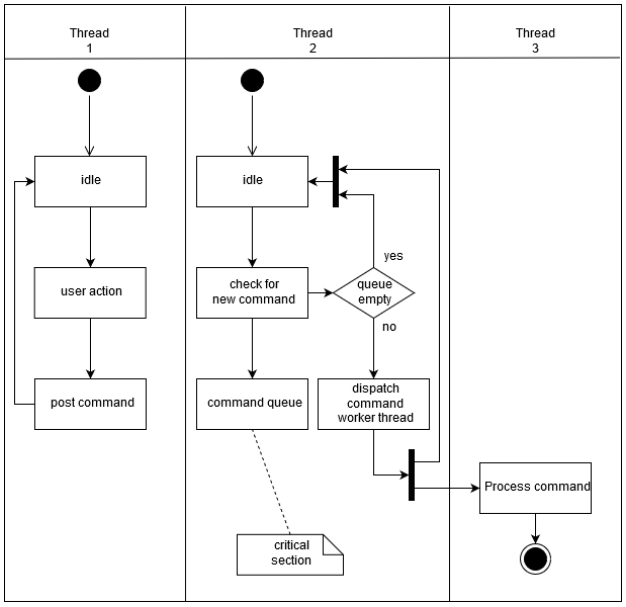
2.8.1 *Use Case Scenario*

Use-case scenario dijelaskan secara tekstual dalam beberapa format tergantung kebutuhannya, yaitu singkat (*brief*), *informal* (*casual*), atau lengkap (*fully dressed*), yang dapat dijelaskan dalam satu atau dua kolom [26]. Pada penelitian ini digunakan format yang diperkenalkan oleh Russ Miles dan Kim Hamilton [27] dengan bagian yang terlibat sebagai berikut.

1. Nama *Use Case* (*Use case name*)
2. Persyaratan Terkait (*Related Requirements*), kondisi spesifik yang harus terpenuhi sebelum *use-case* dieksekusi oleh aktor.
3. Tujuan (*Goal in Context*), penggunaan *use case* dan menjelaskan mengapa *use case* ini menjadi penting digunakan.
4. Kondisi Awal (*Preconditions*), kondisi awal sebelum *use case* dieksekusi.
5. Kondisi Akhir Berhasil (*Successful End Condition*), kondisi ketika *use case* berhasil dieksekusi.
6. Kondisi Akhir Gagal (*Failed End Condition*), kondisi ketika *use case* gagal dieksekusi.
7. Aktor Utama (*Primary Actors*), Aktor utama yang menggunakan *use case*.
8. Aktor Sekunder (*Secondary Actors*), Aktor sekunder/lainnya yang menggunakan *use case*.
9. Pemicu (*Trigger*), pemicu oleh aktor sehingga *use case* dieksekusi.
10. Alur Utama (*Main Flow*), penjelasan terkait alur utama dari *use case* apabila berjalan secara normal.
11. Ekstensi (*Ekstensions*), yaitu jalur alternatif dari interaksi yang terjadi antara aktor dan sistem yang mencakup percabangan (pilihan) maupun skenario yang gagal sehingga tujuan aktor tidak terpenuhi.

2.8.2 *Activity Diagram*

Activity diagram adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain berdasarkan use case diagram [26]. Berikut adalah contoh dari *activity diagram* pada Gambar 2.5.



Gambar 2.5 Contoh *Activity Diagram*

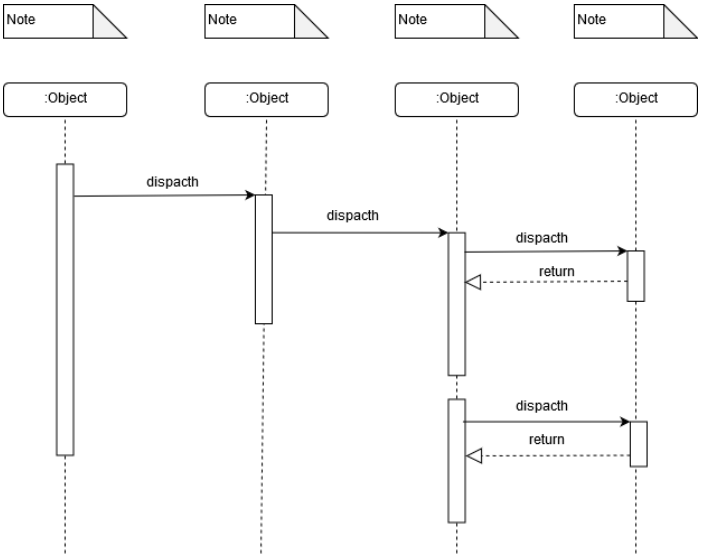
2.8.3 *Class Diagram*

Class diagram merupakan inti dari setiap sistem berorientasi objek, oleh karena itu kaitannya sangat erat dengan *use-case diagram*. *Class diagram* bertujuan untuk menjelaskan berbagai jenis objek dan hubungannya dengan kelas lainnya yang dimiliki oleh sistem. Hal ini lah yang menjadikan *class diagram* memiliki dua macam informasi, yaitu informasi tentang kondisi awal objek dan bagaimana berperilaku dalam lingkungannya [26].

2.8.4 *Sequence Diagram*

Sequence diagram atau dikenal juga sebagai diagram interaksi bertujuan untuk memodelkan interaksi secara *runtime* antara bagian – bagian tertentu pada sistem yang membentuk alur perpindahan sebuah objek [27].

Diagram ini akan meyampaikan urutan dari setiap interaksi pada suatu proses atau bagian – bagian sistem. Dengan menampilkan apa yang menjadi pemicu dari sebuah proses dan menunjukkan informasi lain berupa peristiwa dalam suatu interaksi [27]. Berikut adalah contoh dari *sequence diagram* pada gambar 2.6.



Gambar 2.6 Contoh *Sequence Diagram*