BAB 2
TINJAUAN PUSTAKA

2.1 Pengenalan Tulisan Tangan

Pengenalan tulisan tangan merupakan salah satu tantangan dalam bidang AI. Banyak kegiatan yang melibatkan kegiatan menulis yang dapat dilakukan pengenalan tulisan tangan, seperti digitalisasi dokumen, menyimpan, mengambil dan mengindeks dokumen, penyortiran surat otomatis, pemrosesan cek bank, dan pemrosesan formulir [1]. Hal ini dilihat dari sisi efektifitas dan kemudahan dalam menggunakan tulisan tangan daripada memanfaatkan komputer. Meski demikian, penggunaan tulisan tangan terbatas pada kegiatan – kegiatan *simple* dan tidak begitu kompleks, namun tetap saja tidak sedikit kegiatan ini masih berlangsung meski di era moderen saat ini.

Pengenalan tulisan tangan biasa disebut juga *Intelligent Character Recognition* (ICR), yaitu sebuah kegiatan untuk mengonversi dokumen bertulisan tangan ke versi digital atau elektronik. Dalam hal serupa, ada juga istilah yang dikenal *Optical Character Recognition* (OCR) untuk kasus dokumen hasil cetak atau *print*. Dalam banyak kasus, ICR jauh lebih sulit diimplementasikan karena karakteristik tulisan tangan yang berbeda – beda dan memiliki cakupan yang luas [1].

Menulis artinya sebuah kegiatan menggoreskan pena atau alat tulis lainnya ke sebuah media, terutama media cetak seperti kertas, buku, dll. Dengan penggunaan media tersebut, hasil dari tulisan tangan memiliki keterbatasan secara fisik, baik ditinjau dari waktu dan juga kondisi. Sebagai contoh, tinta pada kertas akan memudar bila terkena tetesan air, atau kertas yang terkena air akan mudah sobek dan berlubang. Kelemahan fisik inilah yang akan merusak keberadaan dari data atau tulisan tangan tersebut.

Salah satu penelitian yang sedang populer saat ini adalah bagaimana membangun sebuah sistem yang mampu mengenali tulisan tangan, sehingga banyak dilakukan pengujian metode – metode untuk memperoleh hasil yang diharapkan. Begitupun pada penelitian ini yang bertujuan untuk menguji kemampuan dari metode CNN.

2.2 Pengolahan Citra Digital

2.2.1 Pengertian Citra

Manusia merupakan makhluk visual yang mengandalkan pengelihatan untuk memahami sekelilingnya, dengan kemampuan tersebut manusia dapat merasakan dan mengetahui perbedaan sesuatu hal dengan cepat. Citra sendiri didefinisikan sebagai fungsi dari dua variabel misalnya $a_{x,y}$ di mana a merupakan amplitudo atau dalam hal ini ialah kecerahan, sedangkan x, y ialah koordinat yang membentuk citra [13].

Citra merupakan istilah lain untuk gambar pada salah satu komponen multimedia dengan karakteristik yang tidak dimiliki oleh data teks. Secara harfiah citra ialah gambar atau sebuah terusan cahaya pada bidang dwimatra (dua dimensi) yang ditinjau dari sudut pandang matematis. Cahaya yang menyinari suatu objek akan memberikan gambaran tentang objek tersebut, hal ini dikarenakan adanya pantulan cahaya saat sumber cahaya menyinari objek. Pantulan cahaya tersebut akan ditangkap oleh alat – alat optik, misalnya mata manusia, kamera, *scanner*, dan sebagainya [14].

Citra secara umum terbagi menjadi dua jenis, citra analog dan citra digital, dalam studi tentang pengolahan citra, citra yang digunakan merupakan citra digital yang dihasilkan dari mesin atau alat elektronik yang mampu menangkap gambar.

Pada penelitian ini, citra berperan sebagai data masukan yang akan berpengaruh terhadap proses dan hasil. Citra dengan kualitas yang buruk akan menurunkan akurasi pada klasifikasi. Oleh karena itu, guna memperoleh kualitas citra yang lebih baik, dilakukanlah pengolahan citra.

2.2.2 Pengolahan Citra

Citra atau gambar merupakan salah satu media yang dapat menyampaikan informasi lebih detil dan jelas bila dibandingkan dengan teks, penggunaannya pun dapat merepresentasikan suatu objek secara lebih nyata, namun tidak semua citra memiliki kualitas yang baik [13].

Dalam pengolahan citra digital terutama citra yang digunakan sebagai data masukan suatu proses, perlu memiliki kualitas yang baik. Hal – hal yang

mengganggu kualitas citra contohnya adalah adanya bercak hitam atau putih yang bukan bagian dari objek pada citra.

Meskipun demikian, untuk memperoleh citra yang baik memerlukan alat optik yang bagus dengan harga yang cukup mahal. Alternatif dari penggunaan alat optik yang mahal adalah dilakukannya pengolahan citra dengan beragam macam metode, salah satunya ialah menggunakan konsep *smoothing*.

2.2.2.1 Konversi Citra RGB ke *Grayscale*

Tahapan awal dalam *preprocessing* dari proses pengolahan citra pada penelitian ini diawali dengan pengkonversian citra RGB ke citra *grayscale*. Citra *grayscale* ialah citra yang hanya memiliki satu buah kanal sehingga yang ditampilkan hanya nilai intensitas atau derajat keabuannya saja. Citra *grayscale* disebut juga sebagai citra 8 bit karena ukurannya yang jauh lebih hemat daripada citra RGB. Perbedaan citra RGB dan *grayscale* dapat dilihat pada Gambar 2.1.

Gambar 2.1 Perbedaan Citra RGB dan *Grayscale*

Pembangunan program pada penelitian ini menggunakan metode *grayscale* dengan persamaan (2.1) sebagai berikut.

$$I_{x,y} = (0.299 \times R) + (0.587 \times G) + (0.114 \times B) \qquad (2.1)$$
$$x = 0,1,..,M-1, \qquad y = 0,1,..,N-1$$

Keterangan:

$I_{x,y}$ = Nilai derajat keabuan yang dicari

R = Nilai dari piksel berwarna merah (*Red*)

G = Nilai dari piksel berwarna hijau (*Green*)

B = Nilai dari piksel berwarna biru (*Blue*)

M = Dimensi baris matriks

N = Dimensi kolom matriks

2.2.2.2 Citra Integral

Citra integral merupakan konsep menjumlahkan setiap piksel citra dengan nilai tetangganya dalam aturan tertentu dengan tujuan untuk menyimpan informasi jumlah keseluruhan intensitas citra. Proses ini terbukti meningkatkan efisiensi dan efektifitas dalam mengurangi waktu proses [15]. Adapun langkah pembentukan citra integral ialah sebagai berikut.

a. Pembuatan data matriks citra integral $g_{x,y}$ dari citra *grayscale* $I_{i,j}$ dengan ukuran $M \times N$ menggunakan persamaan 2.2.

$$g_{x,y} = \sum_{i=0}^{x}\sum_{j=0}^{y} I_{i,j}\,,\ \ x = 0,1,2,..,M-1, y = 0,1,2,..,N-1 \qquad (2.2)$$

Persamaan 2.2 dapat dilakukan penyederhanaan lagi untuk komputasi yang lebih efektif, penyederhanaan tersebut dapat dilihat pada persamaan 2.3 sampai 2.5.

$$g_{0,y} = I_{0,y} + g_{0,y-1} \qquad y = 1,2,..,N-1 \qquad (2.3)$$

$$g_{x,0} = I_{x,0} + g_{x-1,0} \qquad x = 1,2,..,M-1 \qquad (2.4)$$

$$g_{x,y} = I_{x,y} + g_{x,y-1} + g_{x-1,y} - g_{x-1,y-1}$$
$$x = 1,2,..,M-1, \quad y = 1,2,..,N-1 \qquad (2.5)$$

Keterangan:

g = citra integral.

x = koordinat baris pada citra integral.

y = koordinat kolom pada citra integral.

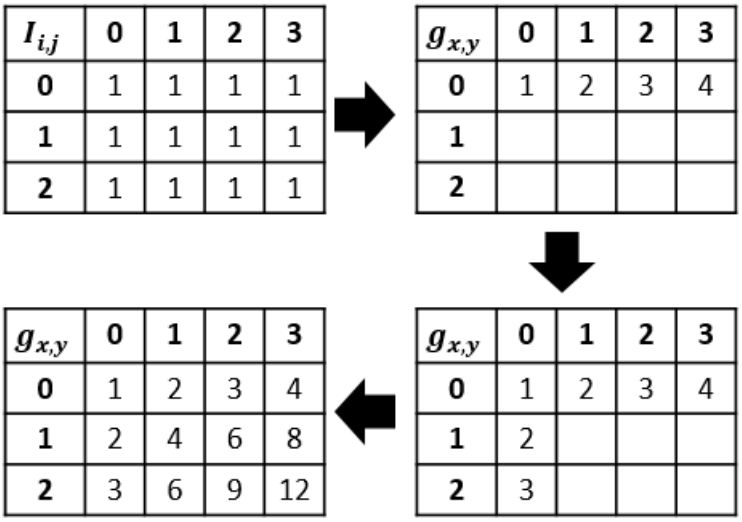
I = Intensitas citra *grayscale.*

M = Jumlah baris pada matriks.

N = Jumlah kolom pada matriks.

Nilai citra integral pada koordinat $g_{0,0}$ akan sama dengan nilai pada koordinat matriks $I_{0,0}$. Persamaan 2.3 digunakan untuk perhitungan pada baris pertama matriks. Sedangkan persamaan 2.4 digunakan untuk perhitungan pada kolom pertama, dan persamaan 2.5 digunakan untuk

perhitungan citra $g_{x,y}$ dengan $x, y \geq 1$. Ilustrasi dari perhitungan ini dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi Citra Integral

b. Menentukan dimensi *mask* yang akan mempengaruhi jumlah intensitas yang diperoleh dari area *mask* tersebut. Ilustrasi penggunaan *mask* dapat dilihat pada Gambar 2.3.

$g_{x,y}$	0	1	2	3
0	1	2	3	4
1	2	4	6	8
2	3	6	9	12

Gambar 2.3 Ilustrasi *Mask* Berukuran 3 × 3

Dengan dimensi atau ukuran *mask* 3 × 3, diperoleh jumlah piksel tetangga sebanyak 8 piksel dari titik koordinat (1,1). Koordinat yang menjadi tetangga adalah (0,0), (0,1), (0,2), (1,0), (1,2), (2,0), (2,1), dan (2,2).

c. Perhitungan nilai rata – rata citra integral pada rentang dimensi *mask* dengan persamaan 2.6.

$$mean(x,y) = \frac{s(x,y)}{wx \times wy}, \quad x = 0,1,2..M-1, \quad y = 0,1,2..N-1 \tag{2.6}$$

Keterangan:

$mean(x, y)$ = fungsi *mean* atau rata – rata dari nilai piksel tetangga.

$s(x, y)$ = fungsi hasil penjumlahan piksel tetangga.

wx = *window* atau total piksel tetangga pada posisi baris.

wy = *window* atau total piksel tetangga pada posisi kolom.

Dimana $s(x, y)$ dapat dihitung dengan persamaan 2.7 berikut ini.

$$s(x,y) = \left(g_{x+dx-1,y+dy-1} + g_{x-dx,y-dy}\right) - \left(g_{x-dx,y+dy-1} + g_{x+dx-1,y-dy}\right) \quad (2.7)$$

Dengan $dx = round\left(\frac{wx}{2}\right)$, $dy = round\left(\frac{wy}{2}\right)$,

Persamaan 2.7 di atas memiliki kelemahan, yaitu tidak mampu menangani kondisi dimana hasil perhitungan koordinat menghasilkan nilai negatif, sehingga lebih kecil daripada ukuran matriks, atau bernilai lebih besar dari ukuran matriks. Untuk menangani persoalan tersebut, maka digunakanlah aturan – aturan sebagai berikut.

1. Jika koordinat $x < 0$ atau koordinat $y < 0$, maka nilai $g_{x,y}$ tersebut diganti dengan nol (0)
2. Jika koordinat $x > (M - 1)$ dan koordinat $0 \leq y \leq (N - 1)$, maka koordinat x diubah menjadi $(M - 1)$.
3. Jika koordinat $0 \leq x \leq (M - 1)$ dan koordinat $y > (N - 1)$, maka koordinat y diubah menjadi $(N - 1)$.
4. Jika koordinat $x > (M - 1)$ dan koordinat $y > (N - 1)$, maka koordinat x diubah menjadi $M - 1$ dan y diubah menjadi $N - 1$.
5. Jika koordinat $0 \leq x \leq (M - 1)$ dan koordinat $0 \leq y \leq (N - 1)$, maka tidak ada perubahan nilai dan koordinat.

Hasil akhir perhitungan rata – rata citra integral akan sama dengan hasil perhitungan menggunakan metode *mean filtering*, yaitu sebuah metode untuk menghitung rata – rata dari nilai ketetanggan berdasarkan ukuran *mask*. Hanya saja konsep perhitungan dari citra integral tidak melalui tahap pencacahan setiap *cell* yang terlibat pada area *mask*, namun berdasarkan

rumus dengan nilai hasil akumulasi. Oleh karena itu waktu prosesnya jauh lebih cepat daripada *mean filtering*.

Meskipun metode ini terbukti mampu mengurangi waktu proses, namun terdapat kekurangan yang cukup berdampak bagi citra pada persamaan 2.6, yaitu hasil perhitungan rata – rata yang kurang tepat karena adanya pengabaian nilai pada kondisi matriks $g_{x,y}$ ketika koordinat (x, y) berada diluar ruang lingkup koordinat matriks *grayscale* pada perhitungan fungsi $s(x, y)$. Sebagai contoh diberikan matriks *grayscale* $(M \times N)$ berukuran (5×5) pada Tabel 2.1 berikut dengan *mask* $(wx \times wy)$ berukuran (3×3).

Tabel 2.1 Contoh Matriks *Grayscale* pada Perhitungan Rata - Rata Citra Integral

$I_{i,j}$	**0**	**1**	**2**	**3**	**4**
0	253	253	254	50	50
1	250	26	40	55	55
2	250	46	50	53	54
3	255	45	55	54	54
4	255	46	55	60	255

Ilustrasi jika dilakukan perhitungan rata – rata pada koordinat $(0,0)$ menggunakan matriks pada Tabel 2.1 dapat dilihat pada Tabel 2.2 berikut.

Tabel 2.2 Ilustrasi Perhitungan Rata – Rata Pada Koordinat $(\mathbf{0}, \mathbf{0})$

$I_{i,j}$	**-1**	**0**	**1**	**2**	**3**	**4**
-1						
0		253	253	254	50	50
1		250	26	40	55	55
2		250	46	50	53	54
3		255	45	55	54	54
4		255	46	55	60	255

Kondisi pada Tabel 2.2 dapat menjadi kekurangan pada persamaan 2.6 apabila kondisi proses yang diharapkan sebagai berikut.

1. Perhitungan rata – rata dimulai dari koordinat $(0,0)$ hingga koordinat akhir dengan tujuan agar matriks yang dihasilkan di akhir proses

berukuran sama dengan matriks citra sebelum perhitungan rata – rata citra integral.

2. Tidak dilakukan penambahan *padding* pada citra *grayscale* dengan asumsi bahwa proses tersebut akan memakan waktu apabila ukuran *mask* yang digunakan cukup besar. Sebagai contoh digunakan ukuran *mask* (20×20) maka perlu dilakukan penambahan *padding* setidaknya setengah dari ukuran *mask* untuk setiap tepi matriks. Hal ini akan mengurangi efektifitas dari penggunaan citra integral.
3. *Background* dari citra *input* berwarna putih dengan objek data berwarna hitam.

Ketiga kondisi tersebut digunakan pada penelitian ini, sehingga apabila ilustrasi pada Tabel 2.2 tersebut dihitung secara manual dengan menganggap *cell* yang tidak memiliki nilai namun terkena *mask* bernilai 255, maka hasil perhitungan rata – ratanya sebagai berikut.

$$mean = \frac{255 + 255 + 255 + 255 + 253 + 253 + 255 + 250 + 26}{9}$$
$$= \frac{2057}{9}$$
$$= 228.55556$$
$$\approx 229$$

Hasil perhitungan rata – rata yang diharapkan dari contoh pada Tabel 2.2 adalah 229, namun jika menggunakan persamaan 2.6 hasil yang diperoleh akan berbeda. Sebagai contoh, dibentuk citra integral dari Tabel 2.1 sebagai berikut.

Tabel 2.3 Contoh Matriks Citra Integral $g_{x,y}$ dari Matriks *Grayscale* $I_{i,j}$

$g_{x,y}$	**0**	**1**	**2**	**3**	**4**
0	253	506	760	810	860
1	503	782	1076	1181	1286
2	753	1078	1422	1580	1739
3	1008	1378	1777	1989	2202
4	1263	1679	2133	2405	2873

Selanjutnya digunakan fungsi pada persamaan 2.7 sebagai langkah awal untuk menghitung rata – rata citra integral. Proses ini diawali dengan perhitungan nilai dx dan dy.

$$dx = round\left(\frac{wx}{2}\right)$$
$$= round\left(\frac{3}{2}\right)$$
$$= round(1.5)$$
$$\approx 2$$

Karena nilai dari $wx = wy$, maka nilai dari $dx = dy$.

Selanjutnya dihitung fungsi $s(x, y)$ pada koordinat $(0,0)$ sebagai berikut.

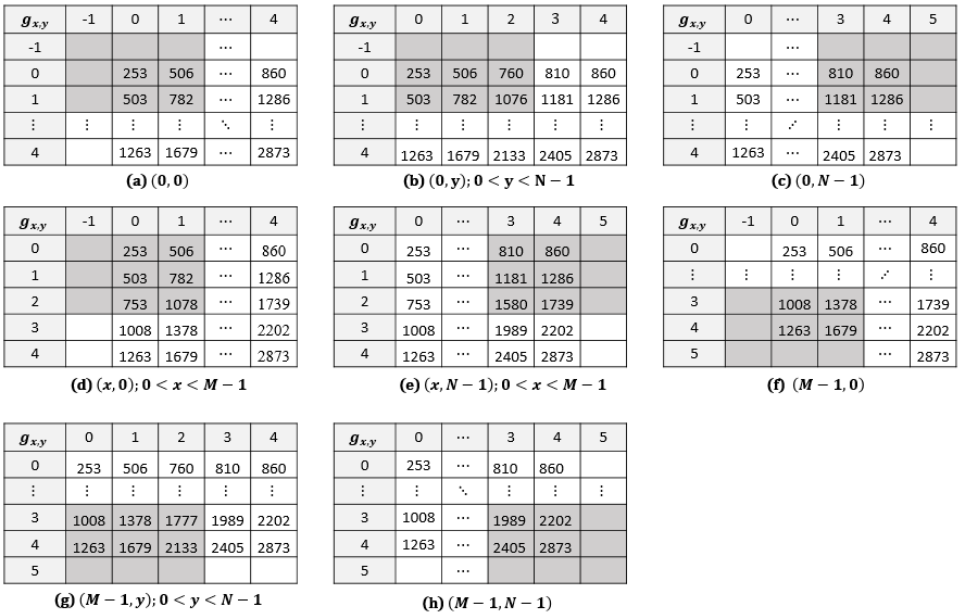
$$s(x, y) = \left(g_{x+dx-1,y+dy-1} + g_{x-dx,y-dy}\right) - \left(g_{x-dx,y+dy-1} + g_{x+dx-1,y-dy}\right)$$
$$s(0,0) = \left(g_{0+2-1,0+2-1} + g_{0-2,0-2}\right) - \left(g_{0-2,0+2-1} + g_{0+2-1,0-2}\right)$$
$$= \left(g_{1,1} + g_{-2,-2}\right) - \left(g_{-2,1} + g_{1,-2}\right)$$
$$= (782 + 0) - (0 + 0)$$
$$= 782$$

Dari perhitungan tersebut, dapat dilihat ada beberapa koordinat dari matriks citra integral $g_{x,y}$ yang lebih kecil dari nol (0) sehingga diimplementasikan syarat pertama dari perhitungan $s(x, y)$, yaitu jika koordinat $x < 0$ atau koordinat $y < 0$, maka nilai $g_{x,y}$ tersebut diganti dengan nol (0).

Tahap selanjutnya adalah perhitungan rata – rata citra integral dengan persamaan 2.6 pada koordinat $(0,0)$ sebagai berikut.

$$mean(x, y) = \frac{s(x, y)}{wx \times wy}$$
$$mean(0,0) = \frac{s(0,0)}{3 \times 3}$$
$$= \frac{782}{9}$$
$$= 86.888889$$
$$\approx 87$$

Hasil dari perhitungan menggunakan persamaan 2.6 berbeda dengan perhitungan manual, hal ini dikarenakan persamaan tersebut tidak melibatkan area *cell* yang tidak memiliki nilai, oleh karena itu akan ada beberapa persoalan dalam penerapan persamaan tersebut. Persoalan – persoalan tersebut di representasikan pada Gambar 2.4.



Gambar 2.4 Permasalahan Perhitungan Rata – Rata Citra Integral

Dari Gambar 2.4, dapat dilihat bahwa terdapat delapan kondisi yang akan mengalami kekeliruan dalam perhitungan rata – rata citra integral. Berikut penjelasan dari setiap kondisinya.

Tabel 2.4 Penjelasan Kondisi Kekeliruan

Huruf	Kondisi	Keterangan
a	$(0,0)$	Pada perhitungan $s(x,y)$ di koordinat $(0,0)$
b	$(0,y); 0 < y < N-1$	Pada perhitungan $s(x,y)$ di baris pertama dimulai dari kolom kedua hingga satu kolom sebelum kolom terakhir
c	$(0,N-1)$	Pada perhitungan $s(x,y)$ di baris pertama kolom terakhir
d	$(x,0); 0 < x < M-1$	Pada perhitungan $s(x,y)$ di kolom pertama baris kedua hingga satu kolom sebelum baris terakhir
e	$(x,N-1); 0 < x < M-1$	Pada perhitungan $s(x,y)$ di kolom terakhir baris kedua hingga satu kolom sebelum baris terakhir
f	$(M-1,0)$	Pada perhitungan $s(x,y)$ di baris terakhir dan kolom pertama
g	$(M-1,y); 0 < y < N-1$	Pada perhitungan $s(x,y)$ di baris terakhir dan kolom kedua hingga satu kolom sebelum kolom terakhir
h	$(M-1,N-1)$	Pada perhitungan $s(x,y)$ di baris terakhir dan kolom terakhir

Oleh karena itu pada penelitian ini, diusulkan suatu teknik pencarian *cell* atau piksel pada *mask* yang tidak memiliki nilai dengan menambahkan satu fungsi artifisial $a(x,y)$ pada persamaan 2.6 sehingga menjadi seperti pada persamaan 2.8.

$$mean(x,y) = \frac{s(x,y)}{wx \times wy} + a(x,y),$$
$$x = 0,1,2..M-1, \quad y = 0,1,2..N-1 \tag{2.8}$$

Nilai dari fungsi $a(x,y)$ dapat dihitung dengan persamaan 2.9.

$$a(x,y) = \begin{cases} \frac{nw(x,y)}{wx \times wy}, & if \begin{pmatrix} (x-nr<0 \; or \; x+nr \geq M) \\ or \; (y-nc<0 \; or \; y+nc \geq N) \end{pmatrix} \\ 0, & sebaliknya \end{cases} \tag{2.9}$$

Dengan $nr = trunc\left(\frac{wx}{2}\right)$ dan $nc = trunc\left(\frac{wy}{2}\right)$

Keterangan:

$a(x,y)$ = Rata – rata nilai intensitas piksel berwarna putih yang tidak ada pada rentang dimensi citra.

$nw(x,y)$ = Jumlah nilai intensitas piksel berwarna putih yang tidak ada pada citra.

nr = Jarak terluar dari *mask* terhadap titik pusat *mask* dalam baris

nc = Jarak terluar dari *mask* terhadap titik pusat *mask* dalam kolom

Dimana nilai fungsi $nw(x,y)$ dapat dihitung dengan persamaan 2.10.

$$nw(x,y) = \begin{cases} \Big(rp(x) \times wy + cp(y) \times \big(wx - rp(x)\big)\Big) \times 255, & if(x - nr < 0 \; and \; y - nc < 0) \\ & or \; (x - nr < 0 \; and \; y + nc \geq N) \\ & or \; (x + nr \geq M \; and \; y - nc < 0) \\ & or \; (x + nr \geq M \; and \; y + nc \geq N) \\ rp(x) \times wy \times 255, & if \begin{pmatrix} (x - nr < 0 \; or \; x + nr \geq M) \\ and \; (y > 0 \; and \; y < N) \end{pmatrix} \\ cp(y) \times wx \times 255, & if \begin{pmatrix} (y - nc < 0 \; or \; y + nc \geq N) \\ and \; (x > 0 \; and \; x < M) \end{pmatrix} \end{cases} \tag{2.10}$$

Keterangan:

$rp(x)$ = *Row padding*, jumlah baris pada *mask* yang berada di luar dimensi citra dihitung berdasarkan jarak terluar.

$cp(y)$ = *Column padding*, jumlah kolom pada *mask* yang berada di luar dimensi citra dihitung berdasarkan jarak terluar

Dengan fungsi $rp(x)$, dan $cp(y)$ yang dapat dilihat pada persamaan 2.11.

$$rp(x) = \begin{cases} |x - nr|, & if(x - nr < 0) \\ x + nr \; - M + 1\,, & if(x + nr \geq M) \end{cases}$$

$$cp(y) = \begin{cases} |y - nc|, & if(y - nc < 0) \\ \mathrm{y} + nc \; - N + 1\,, & if(y + nc \geq N) \end{cases} \tag{2.11}$$

Pada persamaan 2.10, terdapat tiga kondisi yang dapat menentukan nilai dari fungsi $nw(x,y)$ yang bertujuan untuk menyelesaikan permasalahan

pada Gambar 2.4. Berikut adalah letak penggunaan fungsi $nw(x, y)$ berdasarkan persoalan pada gambar tersebut.

Tabel 2.5 Penggunaan Fungsi $nw(x, y)$ dengan Berbagai Kondisi

NO	Rumus dan kondisi	Referensi Gambar
1	$\left(rp(x) \times wy + cp(y) \times \left(wx - rp(x)\right)\right) \times 255$ Ketika nilai koordinat $x - nr < 0$ dan $y - nc < 0$ Atau koordinat $x - nr < 0$ dan $y + nc \geq N$ Atau koordinat $x + nr \geq M$ dan $y - nc < 0$ Atau koordinat $x + nr \geq M$ dan $y + nc \geq N$	2.5(a), 2.5(c), 2.5(f), 2.5(h)
2	$rp(x) \times wy \times 255$ Ketika nilai koordinat $x - nr < 0$ atau $x + nr \geq M$ dan nilai koordinat $0 < y < N$	25(b) dan 25(g)
3	$cp(y) \times wx \times 255$ Ketika nilai koordinat $y - nc < 0$ atau $y + nc \geq N$ dan nilai koordinat $0 < x < M$	2.5(d) dan 2.5(e)

Pada contoh perhitungan rata – rata citra integral menggunakan persamaan 2.6 sebelumnya, diperoleh nilai sebesar 87. Dan apabila hasil perhitungan tersebut dilanjut menggunakan persamaan 2.8, maka nilai yang dihasilkan dari serangkaian fungsi $a(x, y)$ adalah 142, sehingga hasil akhir perhitungan rata – rata sebagai berikut.

$$mean(x, y) = \frac{s(x, y)}{wx \times wy} + a(x, y)$$

$$mean(0,0) = 87 + 142$$

$$= 229$$

Jika diperhatikan perhitungan menggunakan persamaan 2.8 dengan perhitungan manual yang telah dilakukan sebelumnya memiliki nilai yang sama. Oleh karena itu penambahan fungsi $a(x, y)$ bertujuan untuk menghitung rata – rata total nilai piksel yang berada pada rentang *mask* namun tidak memiliki nilai dengan tetap

menjaga performa dari penggunaan citra integral. Penjelasan secara mendetail terkait perhitungan fungsi $a(x, y)$ akan dibahas pada bab 3 subbab 3.2.3 dalam tahap *smoothing* dan *thresholding*.

2.2.2.3 Citra Biner

Citra biner dapat diperoleh melalui *thresholding*, yaitu sebuah teknik yang memanfaatkan ambang batas sebagai batasan dan penentu suatu nilai. Pada kasus pengenalan tulisan tangan dengan latar *background* putih, *thresholding* pada matriks *grayscale* $IG_{x,y}$ dilakukan dengan persamaan berikut.

$$IG_{x,y} \begin{cases} 1, & IG_{x,y} \leq T \\ 0, & IG_{x,y} > T \end{cases} \quad (2.12)$$

Keterangan:

T = Nilai ambang batas

Metode yang digunakan dalam *thresholding* ini adalah *sauvola*. Metode ini termasuk ke dalam salah satu dari metode *Locally Adaptive Thresholding*. Yaitu sebuah metode yang dapat menentukan nilai ambang batas melalui nilai piksel lokal atau nilai disekitar piksel yang bersangkutan [15].

Penggunaan metode *sauvola* pada penelitian ini mengadopsi teknik yang disarankan oleh [15] dengan memanfaatkan citra integral untuk mempercepat proses *thresholding*. Perhatikan tabel perbandingan waktu eksekusi berikut ini.

Tabel 2.6 Perbandingan Waktu Proses Beberapa Metode Pada Citra Berukuran 512×512 [15]

Ukuran *Mask*	Citra integral Dan sauvola	Bernsen	Niblack	Sauvola
3	0.2496	0.8112	7.176	7.1448
7	0.234	1.3728	7.3944	7.3944
11	0.234	2.2152	7.9093	7.9561
15	0.234	3.4164	8.5177	8.5489
19	0.234	4.7892	9.2509	9.2821
23	0.234	6.3336	10.0777	10.0621
27	0.2184	8.1277	11.1073	11.0605
31	0.2028	9.9997	12.0745	12.1369
35	0.1872	12.0589	13.2913	13.3225

Adapun rumus metode *sauvola* yang digunakan untuk memenuhi kriteria citra integral dapat dilihat pada persamaan 2.13.

$$T(x,y) = mean(x,y)\left[1 + k\left(\frac{\partial(x,y)}{1-\partial(x,y)} - 1\right)\right]$$

$$\text{Dimana } \partial(x,y) = I(x,y) - mean(x,y), \quad k = [0.1,0.9] \qquad (2.13)$$

$$\partial(x,y) = \begin{cases} -1, & \partial(x,y) = 1 \\ \partial(x,y), & \partial(x,y) \neq 1 \end{cases}$$

Keterangan:

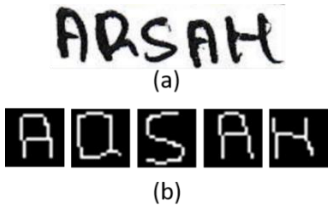
∂ = Standar deviasi

k = Nilai bias dengan rentang [0.1,0.9], yang digunakan dalam penelitian ini ialah 0.1

2.2.2.4 Segmentasi Citra Menggunakan *Connected Component Labeling*

Segmentasi citra digunakan untuk memisahkan objek – objek yang ada dalam sebuah citra. Proses pemisahan objek dalam sebuah citra akan bermanfaat untuk proses klasifikasi. Dengan dilakukannya segmentasi dan pemotongan, nilai – nilai yang tidak diperlukan dapat dihilangkan sesuai kebutuhan.

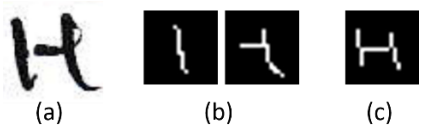
Sekelompok piksel akan membentuk satu objek selama ia tidak terputus, dan apabila objek lainnya terdeteksi, maka objek tersebut berbeda dengan hasil deteksi sebelumnya [16], sebagai contoh dapat dilihat pada Gambar 2.5.



Gambar 2.5 Citra Sebelum Dilakukan Segmentasi (a), Citra Telah Disegmentasi (b)

Connected Component Labeling (CCL) adalah salah satu dari metode segmentasi yang menghubungkan antar piksel satu sama lain (*connected component*). Penggunaan pada citra biner memungkinkan metode ini untuk mencari piksel dengan warna putih dalam rentang dimensi *window* atau *mask*. Ukuran *mask* yang digunakan akan berpengaruh pada hasil akhir pemisahan objek. Contoh

perbandingan hasil segmentasi dengan penggunaan *mask* 3×3 dengan 5×5 dapat dilihat pada Gambar 2.6



Gambar 2.6 Perbedaan Penggunaan Ukuran *Mask* Pada CCL

Pada Gambar 2.6a merepresentasikan citra asli yang akan dilakukan segmentasi, sedangkan perbedaan penggunaan *mask* dapat dilihat pada Gambar 2.6b dan 2.6c yang masing – masing menggunakan *mask* dengan ukuran 3×3 dan 5×5.

2.2.2.5 Scaling

Scaling merupakan tahap dimana citra hasil segmentasi akan dikonversi ukurannya sesuai dengan kebutuhan. Tahap *scaling* pada penelitian ini tidak menggunakan metode tertentu, hanya berdasarkan aturan untuk menghasilkan citra yang sesuai.

Pada tahap *scaling*, data yang akan digunakan berasal dari tahapan sebelumnya, yaitu segmentasi. Dari tahap segmentasi akan diperoleh dua variabel yang akan digunakan ditahap *scaling* ini, yaitu sebuah matriks yang berisi koordinat – koordinat piksel berwarna putih yang diberi nama *wh*, dan *matriks* bernama *br* yang berisi koordinat – koordinat batas terluar dari citra hasil segmentasi. Berikut ialah contoh isian dari variabel *wh* .

Tabel 2.7 Contoh Isian Matriks $wh_{0,j}$

$wh_{0,j}$	**Nilai** (x, y)
$wh_{0,0}$	1,5
$wh_{0,1}$	1,6
$wh_{0,2}$	2,5
$wh_{0,3}$	2,6
$wh_{0,4}$	2,7
⋮	⋮
$wh_{0,409}$	61,6

Sedangkan contoh isian dari matriks br adalah sebagai berikut.

Tabel 2.8 Contoh Isian Matriks $br_{0,j}$

$br_{0,j}$	Koordinat	Keterangan
$br_{0,0}$	1	Atas
$br_{0,1}$	5	Kanan
$br_{0,2}$	1	Bawah
$br_{0,3}$	5	Kiri

Berikut adalah langkah – langkah pada tahap *scaling*.

1. Menghitung lebar (w) dan tinggi (h) citra hasil segmentasi menggunakan persamaan berikut.

$$\begin{aligned} w &= br_{i,1} - br_{i,3} + 1 \\ h &= br_{i,2} - br_{i,0} + 1 \\ i &= 1,2..,Jumlah\ Objek \end{aligned} \tag{2.14}$$

2. Menghitung nilai perbandingan ukuran citra hasil segmentasi (s) dengan ukuran yang telah ditetapkan (u).

$$s = \begin{cases} \dfrac{u}{w}, & if(w \geq h) \\ \dfrac{u}{h}, & if(w < h) \end{cases} \tag{2.15}$$

3. Menghitung jumlah pergeseran setiap piksel pada koordinat x dan y untuk citra hasil *scaling*.

$$\begin{aligned} shx &= \begin{cases} round\left(\dfrac{u}{2}\right) - round\left(\dfrac{s \times h}{2}\right), & if(w \geq h) \\ 0, & if(w < h) \end{cases} \\ shy &= \begin{cases} round\left(\dfrac{u}{2}\right) - round\left(\dfrac{s \times w}{2}\right), & if(w < h) \\ 0, & if(w \geq h) \end{cases} \end{aligned} \tag{2.16}$$

Keterangan:

shx = Nilai penambah pada koordinat x dihitung dari titik tengah citra berukuran $u \times u$

shy = Nilai penambah pada koordinat y dihitung dari titik tengah citra berukuran $u \times u$

4. *Scaling* koordinat dari variabel $wh_{i,j}$.

$$sx = (x - br_{i,0}) \times s + shx$$
$$sy = (y - br_{i,3}) \times s + shy \quad (2.17)$$

Keterangan:

sx = Koordinat x hasil *scaling*

sy = Koordinat y hasil *scaling*

x = Koordinat x yang diperoleh dari nilai pada $wh_{i,j}$

y = Koordinat y yang diperoleh dari nilai pada $wh_{i,j}$

5. Menghitung jumlah *plotting* piksel berwarna putih dimulai dari sx dan sy.

$$nx = \begin{cases} 2, & if(sx \bmod 1 > 0 \; and \; sx > 1) \\ 1, & lainnya \end{cases}$$
$$ny = \begin{cases} 2, & if(sy \bmod 1 > 0 \; and \; sy > 1) \\ 1, & lainnya \end{cases} \quad (2.18)$$

2.3 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST) atau biasa disebut *Neural Network* (NN) merupakan usaha manusia untuk memodelkan cara kerja atau fungsi dari agen AI layaknya sistem syaraf manusia dalam melaksanakan tugas tertentu. Dengan meniru model tersebut sistem yang mengimplementasikan NN mampu melakukan proses belajar, sehingga disebut sebagai *learning agent*. Dalam hal ini, NN memiliki tingkat efektivitas yang tinggi dalam mengenali pola dan permasalahan – permasalahan terkait klasifikasi, optimasi, ataupun untuk kebutuhan *forecasting* [17]. NN yang digunakan dalam penelitian ini ialah *Convolutional Neural Network* (CNN).

2.3.1 Convolutional Neural Networks

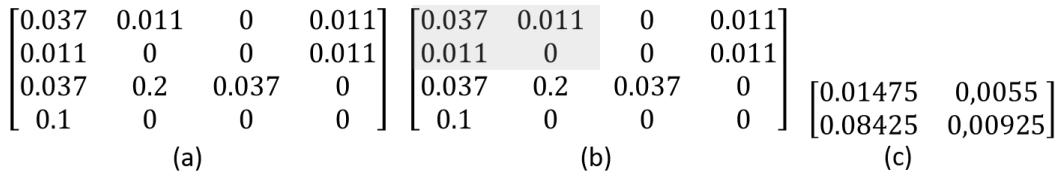
Convolutional Neural Networks (CNN) ialah salah satu dari NN yang sudah berkembang pesat dewasa ini, ia merupakan variasi lain dari *Multi-Layer Perceptron* (MLP). CNN pada dasarnya mampu menangani proses klasifikasi citra lebih baik dari MLP meski tanpa ekstraksi fitur [7].

CNN ialah metode yang memiliki susunan *layer* 3D (lebar, tinggi, dan kedalaman). Lebar dan tinggi ini mengacu pada ukuran *layer*, sedangkan kedalaman merepresentasikan jumlah *layer*. Secara umum jumlah layer pada CNN terbagi menjadi dua macam, yaitu: *layer* ekstraksi dan *layer* klasifikasi.

Layer ekstraksi pada CNN terdiri dari *layer* konvolusi dan *layer pooling*. Setiap *layer*-nya tersusun atas sejumlah *neuron* yang terkoneksi pada daerah lokal (*local region*) [7].

Pada *layer* konvolusi dilakukan proses ekstraksi ciri menggunakan fitur atau kernel dengan nilai awal yang diperoleh dari fungsi *random*. Nilai fitur ini akan dilatih pada tahap *backpropagation*. Seperti langkah pada NN secara umum, CNN pun memerlukan variabel bias dalam prosesnya, dan nilai awal yang ditetapkan pada variabel bias ini keseluruhan ialah nol (0).

Pada *layer pooling* digunakan *average pooling* sebagai metode untuk memperoleh nilai rata – rata dari piksel tetangganya. Sebagai contoh dapat dilihat pada Gambar 2.7.



Gambar 2.7 Contoh Matriks Citra (a), Dilakukan *Average Pooling* dengan Ukuran 2 x 2 (b), Hasil Akhir Setelah Dilakukan *Average Pooling* (c)

Arsitektur pada metode CNN dapat disesuaikan tergantung kedalaman yang ingin digunakan, serta ukuran matriks konvolusi, ukuran *pooling*, dan penggunaan fungsi aktivasi.

Pada penelitian ini, fungsi aktivasi yang digunakan diantara tahap konvolusi dan *pooling* adalah jenis dari *Rectified Units* (RU). RU memiliki empat jenis metode, yaitu:

1. *Rectified Linear Unit* (ReLU)

   Fungsi aktivasi ReLU dapat dilihat pada persamaan 2.19.

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ 0 & if\ x_i < 0 \end{cases} \tag{2.19}$$

   Dimana y_i ialah matriks citra pada koordinat tertentu, dan x_i merupakan hasil hitung dari tahap konvolusi. Metode ini bekerja dengan menginisialisasikan nilai nol (0) pada hasil perhitungan yang bernilai negatif, selain dari pada itu tidak dilakukan perubahan apapun [18].

2. *Leaky Rectified Linear Unit* (*Leaky* ReLU)

   Fungsi aktivasi *Leaky* ReLU dapat dilihat pada persamaan 2.20.

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ \dfrac{x_i}{a_i} & if\ x_i < 0 \end{cases} \tag{2.20}$$

   Dimana a_i merupakan sebuah nilai dengan rentang antara satu (1) sampai tak hingga (∞), dalam penelitian ini digunakan nilai $a_i = 5.5$ yang diperoleh dari penelitian oleh Bing Xu et al. [18].

3. *Parametric Rectified Linear Unit* (PReLU)

   Dalam sebuah peneilitan dikatakan bahwa performa dari metode PReLU jauh lebih baik daripada ReLU dalam menangani klasifikasi citra yang mengalami *scaling* [18]. Meode ini mirip dengan *Leaky* ReLU, hanya saja nilai dari variabel a_i diperoleh dari proses *backpropagation*.

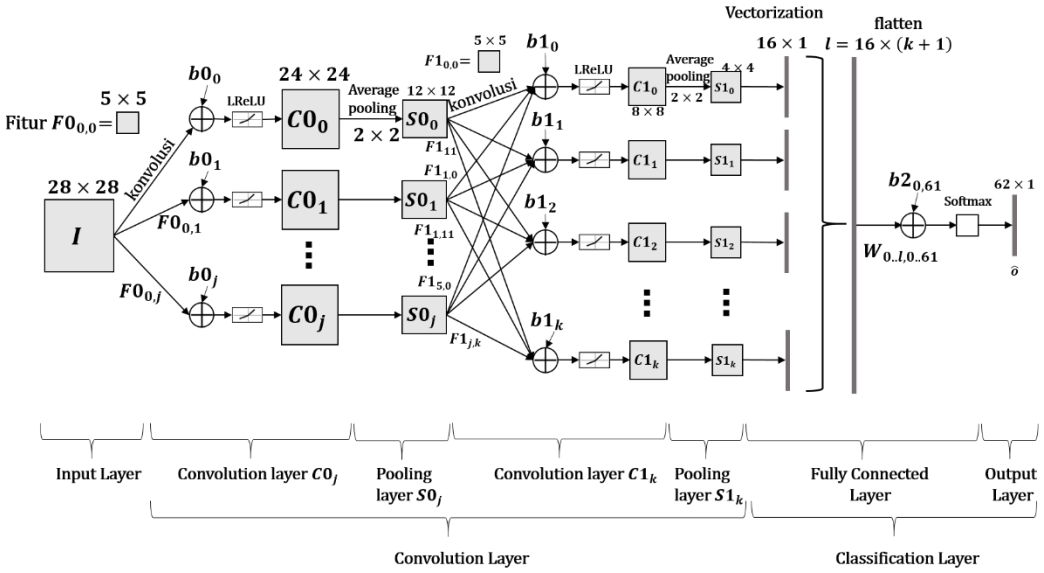
4. *Randomized Leaky Rectified Linear Unit* (RReLU)

   Fungsi aktivasi dari RReLU dapat dilihat pada persamaan 2.21.

$$y_i = \begin{cases} x_{ji} & if\ x_{ji} \geq 0 \\ a_{ji}xj_i & if\ x_{ji} < 0 \end{cases}$$
$$a_{ji}xj_i = \frac{x_{ji}}{\frac{l+u}{2}} \tag{2.21}$$

   Dimana $a_{ji} \sim U(l,u), l < u$, dalam sebuah penelitian digunakan nilai $U(3{,}8)$ berdasarkan saran dari pemenang kompetisi NDSB [18].

Dari hasil penelitian oleh Bing Xu et al. [18] diperoleh kesimpulan bahwa penggunaan metode RReLU dengan $U(3,8)$ memperoleh *error test* yang lebih kecil. Namun pada penelitian ini, digunakan metode *Leaky* ReLU dengan nilai $a_i = 5.5$. Metode ini digunakan untuk memberikan nilai yang konstan ketika proses *backpropagation* pada CNN. Adapun arsitektur CNN yang digunakan dalam penelitian ini dapat dilihat pada Gambar 2.8.



Gambar 2.8 Arsitektur CNN

Arsitektur CNN yang digunakan terdiri dari dua *convolution layer*, dua *pooling layer* dan satu *fully connected layer*. Pada tahap konvolusi digunakan fitur berukuran 5×5, dengan konvolusi pertama sebanyak j dan konvolusi kedua sebanyak k. Selanjutnya dilakukan proses *average pooling* berukuran 2×2 dengan langkah perpindahan atau *stride* $(stx \times sty)$ sebesar (2×2).

Tahap klasifikasi diawali dengan mengkonversi matriks hasil *pooling* kedua menjadi *array* satu dimensi, tahap ini biasa disebut *vectorization*. Setelah diperoleh vektor untuk setiap output pada layer konvolusi, selanjutnya dilakukan proses penyatuan seluruh ciri yang disebut sebagai *flatten*. *Array* dari hasil *flatten* akan dihubungkan satu sama lain terhadap 62 *neuron* yang mewakili setiap karakter pada alfabet, proses ini disebut sebagai *fully connected*. Selanjutnya, hasil dari *fully connected* akan dihitung menggunakan fungsi aktivasi *softmax* untuk memperoleh nilai hasil normalisasi yang akan digunakan untuk pengambilan keputusan.

Metode CNN merupakan metode yang mengadopsi konsep MLP dan *backpropagation*. Sehingga proses akan diawali dengan inisialisasi tahap awal. Tahap ini dilakukan apabila sistem sama sekali belum melakukan proses *training*, sehingga bobot dan bias yang digunakan perlu diinisialisasi terlebih dahulu. Adapun bobot yang digunakan pada penelitian ini ialah fitur $F0_{0,j}$, $F1_{j,k}$, dan $W_{l,m}$ diinisialisasikan dengan nilai antara 0 hingga 1 dalam pecahan dua desimal. Sedangkan bias yang digunakan ialah $b0_j$, $b1_k$, dan $b2_m$ dan setiap bias diinisialisasikan dengan nilai nol (0).

Tahap selanjutnya adalah tahap utama CNN, yaitu *feedforward*, pada tahap ini dilakukan langkah – langkah sebagai berikut.

1. Tahap konvolusi $C0_j$ dapat dilihat pada persamaan 2.22 menggunakan fitur $F0_{0,j}$ berukuran $fx \times fy$ (5×5) terhadap matriks citra *input I* berukuran $M \times N$ (28×28) dimana *j* sejumlah *neuron* lapisan konvolusi pertama.

$$C0_{j,x,y} = \sigma(m_{j,x,y}) \tag{2.22}$$

$$\sigma(m_{j,x,y}) = \begin{cases} \frac{m_{j,x,y}}{5.5}, & if(m_{j,x,y} < 0) \\ m_{j,x,y}, & lainnya \end{cases}$$

$$m_{j,x,y} = \sum_{u=0}^{fx-1} \sum_{v=0}^{fy-1} I_{u+x,v+y} \times F0_{0,j,u,v} + b0_j$$

$$x, y = 0,1,..,M-1,$$

Keterangan:

$C0_j$	=	Matriks Konvolusi pertama
$\sigma(m_{j,x,y})$	=	Fungsi aktivasi *Leaky* ReLU
m_j	=	Matriks hasil konvolusi tanpa fungsi aktivasi
I	=	Matriks masukan yang merepresentasikan citra biner untuk setiap karakter
$F0_j$	=	Matriks fitur untuk konvolusi pertama
$b0_j$	=	*Array* bias untuk konvolusi pertama

2. Tahap *average pooling* $S0_j$ berukuran $M \times N$ (12×12) dapat dilihat pada persamaan 2.23 menggunakan *mask* berdimensi 2×2 dengan *stride* $stx \times sty$ (2×2) dimana j sejumlah *neuron* lapisan *pooling* pertama.

$$S0_{j,x,y} = \frac{1}{4} \sum_{u=x\times stx}^{x\times stx+1} \sum_{v=y\times sty}^{y\times sty+1} C0_{j,u,v} \tag{2.23}$$

$$x, y = 0,1,..,M-1$$

Keterangan:

$S0_j$ = Matriks *pooling* pertama

$C0_j$ = Matriks konvolusi pertama

3. Tahap konvolusi $C1_k$ berukuran $M \times N$ (8×8) dapat dilihat pada persamaan 2.24 menggunakan fitur $F1_{j,k}$ berukuran $fx \times fy$ (5×5) terhadap matriks $S0_j$ berukuran (12×12) dimana j sejumlah neuron pada lapisan *pooling* pertama dan k sejumlah *neuron* pada lapisan konvolusi kedua.

$$C1_{k,x,y} = \sigma(n_{k,x,y}) \tag{2.24}$$

$$\sigma(n_{k,x,y}) = \begin{cases} \frac{n_{k,x,y}}{5.5}, & if(n_{k,x,y} < 0) \\ n_{k,x,y}, & lainnya \end{cases}$$

$$n_{k,x,y} = \sum_{j=0}^{11} \sum_{u=0}^{fx-1} \sum_{v=0}^{fy-1} S0_{j,u+x,v+y} \times F1_{j,k,u,v} + b1_k$$

$$x, y = 0,1,..,M-1$$

Keterangan:

$C1_k$ = Matriks konvolusi kedua

$\sigma(n_{k,x,y})$ = Matriks hasil fungsi aktivasi *L*eaky ReLU pada konvolusi kedua

n_k = Matriks hasil konvolusi tanpa fungsi aktivasi

$S0_j$ = Matriks *pooling* pertama

$F1_{j,k}$ = Matriks fitur untuk konvolusi kedua

$b1_k$ = *Array* bias untuk konvolusi kedua

4. Tahap *average pooling* $S1_k$ dapat dilihat pada persamaan 2.25 menggunakan *mask* berdimensi 2×2 dengan *stride* $stx \times sty$ (2×2).

$$S1_{k,x,y} = \frac{1}{4} \sum_{u=x\times stx}^{x\times stx+1} \sum_{v=y\times sty}^{y\times sty+1} C1_{k,u,v} \qquad (2.25)$$

$$x, y = 0,1,2,3$$

Keterangan:

$S1_k$ = Matriks hasil *average pooling* kedua

$C1_k$ = Matriks konvolusi kedua

5. Tahap selanjutnya ialah *vectorization*, tahap ini mengubah matriks menjadi *array* satu dimensi. Selanjutnya dilakukan *flatten* yaitu penggabungan setiap *array* hasil *vectorization.* Tahap ini dapat dilakukan dengan menggunakan persamaan 2.26.

$$f = FC(\{S1_k\}) \qquad (2.26)$$

6. Selanjutnya ialah tahap *fully connected*, tahap ini menghubungkan setiap *neuron* pada *array flatten* (f_l) dengan l adalah jumlah nilai pada *array flatten* terhadap sejumlah *neuron output* (o_m) dengan $m = 0,1,..,61$. Dilanjutkan dengan menghitung normalisasi menggunakan *softmax* $(\hat{o})$. Tahapan ini dapat dilakukan dengan menggunakan persamaan 2.27.

$$\hat{o}_m = \frac{e^{o_m}}{\sum_{q=0}^{n} e^{o_q}} \qquad (2.27)$$

$$o_m = \sum_{l=0}^{191} W_{l,m} \times f_l + b2_m\,, \qquad m, n = 0,1,..,61,$$

Keterangan:

$\hat{o}_m$ = Matriks hasil *average pooling* kedua

o_m = Keluaran dari proses *fully connected* sebelum dihitung dengan fungsi aktivasi *softmax*

$W_{l,m}$ = Matriks bobot pada lapisan *fully connected*

f_l = *Arrat flatten*

$b2_m$ = *Array* bias pada lapisan *fully connected*

7. Pada pelatihan, dilakukan pembentukan *array* target t_m yang disebut *one hot label*. *Array* ini akan berjumlah 62 data sesuai dengan jumlah *neuron* pada *output layer*. Setiap data pada *array* ini akan merepresentasikan suatu kelas. Urutan dari kelas tersebut dimulai dari angka 0 – 9, dilanjutkan alfabet A – Z, dan dilanjutkan alfabet a – z. *Array* ini terdiri dari nilai nol atau satu, dimana nilai satu hanya dimiliki oleh urutan *array* yang kelasnya sesuai dengan kelas data latih. Selain daripada itu *array* akan bernilai nol.

   Perhitungan *loss* untuk mereduksi error yang diperoleh dari proses *feedforward* menggunakan *Cross Entropy Error Function*. Perhitungan ini dapat dilakukan menggunakan persamaan 2.28 dengan n merupakan jumlah *neuron* pada lapisan *output*.

$$E = -\frac{1}{n}\sum_{m=0}^{n-1} t_m \times \ln \hat{o}_m + (1 - t_m) \times \ln(1 - \hat{o}_m) \tag{2.28}$$

$$n = 62$$

Keterangan:

$\hat{o}_m$ = Hasil *softmax* pada output ke- m.

t_m = *One hot label* atau target sebenarnya yang ingin dicapai. Pada proses *training*, t_m akan bernilai 1 apabila sesuai dengan kelas data citra masukan, dan akan bernilai nol (0) untuk kelas lainnya. Sebagai contoh diberikan matriks citra masukan dengan kelas 'A', maka $t_{10} = 1$ sedangkan t_m lainnya akan bernilai 0.

n = Jumlah neuron *output.*

Pada kasus klasifikasi *multiclass* menggunakan *one hot label*, persamaan 2.28 tersebut dapat disederhanakan berdasarkan kondisi nilai target yang ada. Sebagai contoh diberikan dua kasus perhitungan dengan dua target dan nilai probabilitas yang disimbolkan $\hat{o}$. Kasus pertama menggunakan target bernilai 1 sebagai berikut.

$$E = 1 \times \ln \hat{o} + (1 - 1) \times \ln(1 - \hat{o})$$
$$= 1 \times \ln \hat{o} + 0 \times \ln(1 - \hat{o})$$
$$= 1 \times \ln \hat{o} + 0$$
$$= \ln \hat{o}$$

Kasus kedua menggunakan target bernilai 0 sebagai berikut.

$$E = 0 \times \ln \hat{o} + (1 - 0) \times \ln(1 - \hat{o})$$
$$= 0 \times \ln \hat{o} + 1 \times \ln(1 - \hat{o})$$
$$= 0 + \ln(1 - \hat{o})$$
$$= \ln(1 - \hat{o})$$

Dari kedua kasus tersebut, maka persamaan 2.28 dapat disederhanakan sebagai berikut.

$$E = -\frac{1}{n} \times \sum_{m=0}^{n-1} ce(m) \tag{2.29}$$

$$ce(m) = \begin{cases} \ln \hat{o}_m, & if(t_m = 1) \\ \ln(1 - \hat{o}_m), & lainnya \end{cases}$$

$$n = 62$$

Keterangan:

$ce(m)$ = Perhitungan *cross entropy* berdasarkan nilai target pada *one hot label*

t_m = *One hot label* atau target sebenarnya yang ingin dicapai.

n = Jumlah neuron *output.*

Tahap *backpropagation* akan menggunakan konsep *chain rule*, yaitu hubungan antara turunan dari suatu rumus. Berikut ialah tahapannya.

1. Menghitung turunan fungsi *cross entropy* terhadap *fully connected*. Dari hasil penelitian oleh [19] digunakan rumus turunan pada persamaan 2.30.

$$\Delta o_m = \frac{\partial E}{\partial o_m} = \hat{o}_m - t_m \quad (2.30)$$

$$m = 0,1,..,61$$

Keterangan:

Δo_m = Turunan fungsi *cross entropy* terhadap *fully connected*

$\hat{o}_m$ = *Softmax*

t_m = *One hot label* atau target sebenarnya yang ingin dicapai.

m = Jumlah neuron lapisan *output.*

2. Hitung turunan dari fungsi *error* terhadap bobot $W_{l,m}$ yang direpresentasikan dengan matriks $\Delta W_{l,m}$ menggunakan persamaan 2.31.

$$\Delta W_{l,m} = \frac{\partial E}{\partial W_{l,m}} = \frac{\partial E}{\partial o_m} \times \frac{\partial o_m}{\partial W_{l,m}} \quad (2.31)$$

$$\Delta W_{l,m} = \Delta o_m \times f_l$$

Keterangan:

$\Delta W_{l,m}$ = Turunan fungsi *error* terhadap bobot $W_{l,m}$

Δo_m = Turunan fungsi *cross entropy* terhadap *fully connected*

f_l = *Array flatten*

l = Jumlah *neuron* pada *fully connected layer*.

m = Jumlah neuron lapisan *output.*

3. Hitung turunan bias $b2_m$ yang direpresentasikan dengan matriks $\Delta b2_m$ menggunakan persamaan 2.32 berikut.

$$\Delta b2_m = \frac{\partial E}{\partial b2_m} = \frac{\partial E}{\partial o_m} \times \frac{\partial o_m}{\partial b2_m} \quad (2.32)$$

$$\Delta b2_m = \Delta o_m \times 1$$

$$\Delta b2_m = \Delta o_m$$

4. Hitung turunan fungsi error terhadap *flatten* Δf_l menggunakan persamaan 2.33 dengan $m = 0,1,..,61$.

$$\Delta f_l = \frac{\partial E}{\partial f_l} = \sum_{m=0}^{61} \frac{\partial E}{\partial o_m} \times \frac{\partial o_m}{\partial f_l} \quad (2.33)$$

$$= \sum_{m=0}^{61} \Delta o_m \times W_{l,m}$$

5. *Reverse flatten* menjadi matriks *pooling* $\Delta S1_{k,x,y}$ dengan persamaan 2.34. Fungsi ini bertujuan membentuk kembali matriks $S1_{k,x,y}$ dari hasil perhitungan error pada *array flatten.* Pada *pooling layer* ke dua ini tidak perlu dilakukan penurunan fungsi, karena hanya membentuk kembali matriks dari *flatten*

$$\{\Delta S1_k\} = FC^{-1}\left(\frac{\partial E}{\partial f_l}\right) \quad (2.34)$$

6. Hitung turunan fungsi *cross entropy* terhadap $C1_{k,x,y}$ dilanjutkan terhadap $n_{k,x,y}$ yang direpresentasikan oleh matriks $\Delta Cn1_{k,x,y}$ berukuran $M \times N$ (8×8) menggunakan persamaan 2.35 berikut.

$$\Delta Cn1_{k,x,y} = \frac{\partial E}{\partial n_{k,x,y}} = \frac{\partial E}{\partial C1_{k,x,y}} \times \frac{\partial C1_{k,x,y}}{\partial n_{k,x,y}} \quad (2.35)$$

$$\frac{\partial E}{\partial C1_{k,x,y}} = \frac{1}{4} \times \Delta S1_{k,trunc\left(\frac{x}{2}\right),trunc\left(\frac{y}{2}\right)}$$

$$\frac{\partial C1_{k,x,y}}{\partial n_{k,x,y}} = \begin{cases} 1, & if\left(n_{k,x,y} > 0\right) \\ \frac{1}{5.5}, & Selain\ dari\ itu \end{cases}$$

$$x, y = 0,1,..,7$$

Keterangan:

$\Delta Cn1_k$ = Turunan fungsi *error* terhadap matriks konvolusi kedua

$\Delta S1_k$ = Matriks hasil *reverse* dari *array flatten*

n_k = Matriks hasil konvolusi kedua tanpa fungsi aktivasi

7. Hitung turunan fungsi *cross entropy* terhadap fitur $F1_{j,k,x,y}$ menggunakan persamaan 2.36 berikut dengan $u, v = 0,1,..,7$ berdasarkan ukuran matriks $S0_{j,x,y}$.

$$\Delta F1_{j,k,x,y} = \frac{\partial E}{\partial F1_{j,k,x,y}} = \sum_{u=0}^{7}\sum_{v=0}^{7} \frac{\partial E}{\partial n_{k,u,v}} \times \frac{\partial n_{k,u,v}}{\partial F1_{j,k,x,y}} \tag{2.36}$$

$$\frac{\partial n_{k,u,v}}{\partial F1_{j,k,x,y}} = S0_{j,u+x,v+y}$$

$$\Delta F1_{j,k,x,y} = \sum_{u=0}^{7}\sum_{v=0}^{7} \Delta Cn1_{k,u,v} \times S0_{j,u+x,v+y}$$

$$j = 0,1,2,3,4,5, \qquad k = 0,1,..,11$$

$$x, y = 0,1,2,3,4$$

Keterangan:

$\Delta F1_{j,k}$	=	Turunan fungsi *cross entropy* terhadap fitur $F1_{j,k,x,y}$
$\Delta Cn1_k$	=	Turunan fungsi *error* terhadap matriks konvolusi kedua
$S0_j$	=	Matriks *pooling* pertama
n_k	=	Matriks hasil konvolusi kedua tanpa fungsi aktivasi
j	=	Jumlah *neuron* pada lapisan *pooling* pertama
k	=	Jumlah *neuron* pada lapisan konvolusi kedua

8. Hitung turunan bias $b1_k$ menggunakan persamaan 2.37 berikut dengan $x, y = 0,1,..,7$ berdasarkan ukuran matriks $\Delta Cn1_{k,x,y}$.

$$\Delta b1_k = \frac{\partial E}{\partial b1_k} = \frac{\partial E}{\partial C1_{k,x,y}} \times \frac{\partial C1_{k,x,y}}{\partial b1_k} \tag{2.37}$$

$$= \sum_{x=0}^{7}\sum_{y=0}^{7} \frac{1}{4} \times \Delta S1_{k,trunc\left(\frac{x}{2}\right),trunc\left(\frac{y}{2}\right)} \times \begin{cases} 1, & if\left(n_{k,x,y} > 0\right) \\ \frac{1}{5.5}, & Selain\ dari\ itu \end{cases}$$

$$= \sum_{x=0}^{7}\sum_{y=0}^{7} \Delta Cn1_{k,x,y}$$

9. Hitung turunan fungsi *error* terhadap matriks $S0_{J,x,y}$ menggunakan persamaan 2.38 dengan k berdasarkan jumlah *neuron* pada lapisan konvolusi kedua.

$$\Delta S0_{j,x,y} = \frac{\partial E}{S0_{j,x,y}} \tag{2.38}$$

$$= \sum_{k=0}^{11}\sum_{u=0}^{4}\sum_{v=0}^{4} \frac{\partial E}{\partial C1_{k,u-p+x,v-p+y}} \times \frac{\partial C1_{k,u-p+x,v-p+y}}{\partial n_{k,u-p+x,v-p+y}} \times \frac{\partial n_{k,u-p+x,v-p+y}}{\partial S0_{j,x,y}}$$

$$= \sum_{k=0}^{11}\sum_{u=0}^{4}\sum_{v=0}^{4} \Delta Cn1_{k,u-p+x,v-p+y} \times F1_{j,k,u,v}$$

$$p = length\left(F1_{j,k}\right) - 1 = 5 - 1 = 4$$

Rotasi matriks $F1_{j,k,u,v}$ pada persamaan 2.38 sebesar 180°, lalu beri kondisi untuk membatasi proses perhitungan hanya pada koordinat matriks yang tersedia. Sehingga persamaan tersebut menjadi sebagai berikut.

$$\Delta S0_{j,x,y}$$

$$= \sum_{k=0}^{11}\sum_{u=0}^{4}\sum_{v=0}^{4} \begin{cases} \Delta Cn1_{k,u-p+x,v-p+y} \times rF1_{j,k,u,v} \ , & if \begin{pmatrix} (u-p+x) > 0 \ and \\ (v-p+y) > 0 \end{pmatrix} \\ 0 \ , & Lainnya \end{cases}$$

$$p = length\left(F1_{j,k}\right) - 1 = 5 - 1 = 4$$

(2.39)

Keterangan:

$\Delta S0_j$ = Turunan fungsi *cross entropy* terhadap *pooling* pertama

$\Delta Cn1_k$ = Turunan fungsi *error* terhadap matriks konvolusi kedua

$F1_{j,k}$ = Matriks bobot pada lapisan konvolusi kedua

p = Ukuran baris matriks bobot dikurang satu untuk penentuan koordinat pada turunan konvolusi kedua

10. Hitung turunan fungsi *cross entropy* terhadap $C0_{j,x,y}$ dilanjutkan terhadap $m_{j,x,y}$ yang direpresentasikan menjadi matriks $\Delta Cm0_{j,x,y}$ menggunakan persamaan 2.40 berikut.

$$\Delta Cm0_{j,x,y} = \frac{\partial E}{\partial C0_{j,x,y}} \times \frac{\partial C0_{j,x,y}}{\partial m_{j,x,y}} \tag{2.40}$$

$$\frac{\partial E}{\partial C0_{j,x,y}} = \frac{1}{4} \times \Delta S0_{j,trunc\left(\frac{x}{2}\right),trunc\left(\frac{y}{2}\right)}$$

$$\frac{\partial C0_{j,x,y}}{\partial m_{j,x,y}} = \begin{cases} 1, & if\left(m_{j,x,y} > 0\right) \\ \frac{1}{5.5}, & Selain\ dari\ itu \end{cases}$$

$$x, y = 0,1,..,11$$

Keterangan:

$\Delta Cm0_j$ = Turunan fungsi *cross entropy* terhadap konvolusi pertama

$\Delta S0_j$ = Turunan fungsi *error* terhadap matriks *pooling* pertama

m_j = Matriks konvolusi pertama tanpa fungsi aktivasi

11. Hitung turunan fungsi *cross entropy* terhadap fitur $F0_{0,j}$ menggunakan persamaan 2.41 berikut.

$$\Delta F0_{0,j,x,y} = \frac{\partial E}{\partial F0_{0,j,x,y}} = \sum_{u=0}^{23} \sum_{v=0}^{23} \frac{\partial E}{\partial m_{j,x,y}} \times \frac{\partial m_{j,x,y}}{\partial F0_{0,j,x,y}} \tag{2.41}$$

$$\frac{\partial m_{j,x,y}}{\partial F0_{0,j,x,y}} = I_{u+x,v+y}$$

$$\Delta F0_{0,j,x,y} = \sum_{u=0}^{23} \sum_{v=0}^{23} \Delta Cm0_{j,u,v} \times I_{u+x,v+y}$$

$$x, y = 0,1,2,3,4$$

Keterangan:

$\Delta F0_{0,j}$ = Turunan fungsi *cross entropy* terhadap fitur pada konvolusi pertama

$\Delta Cm0_j$ = Turunan fungsi *error* terhadap matriks konvolusi pertama

I = Matriks citra masukan

12. Hitung turunan fungsi bias $b0_j$ menggunakan persamaan 2.42 berikut.

$$\Delta b0_j = \frac{\partial E}{\partial b0_j} = \frac{\partial E}{\partial C0_{j,x,y}} \times \frac{\partial C0_{j,x,y}}{\partial b0_j} \tag{2.42}$$

$$= \sum_{x=0}^{23}\sum_{y=0}^{23} \frac{1}{4} \times \Delta S0_{j,trunc\left(\frac{x}{2}\right),trunc\left(\frac{y}{2}\right)} \times \begin{cases} 1, & if\left(m_{k,x,y} > 0\right) \\ \frac{1}{5.5}, & Selain\ dari\ itu \end{cases}$$

$$= \sum_{x=0}^{23}\sum_{y=0}^{23} \Delta Cm0_{j,x,y}$$

13. Tahap berikutnya adalah memperbaharui nilai dari setiap bobot dan bias menggunakan *stochastic gradient descent*. Parameter yang digunakan pada tahap ini berupa *learning rate* $(a = 0.001)$ dengan perhitungan menggunakan persamaan berikut.

$$F0_{0,j,x,y} = F0_{0,j,x,y} - a \times \Delta F0_{0,j,x,y} \tag{2.43}$$

$$F1_{j,k,x,y} = F1_{j,k,x,y} - a \times \Delta F1_{j,k,x,y}$$

$$W_{l,m} = W_{l,m} - a \times \Delta W_{l,m}$$

$$b0_j = b0_j - a \times \Delta b0_j$$

$$b1_k = b1_k - a \times \Delta b1_k$$

$$b2_m = b2_m - a \times \Delta b2_m$$

2.4 Perhitungan Akurasi

Perhitungan akurasi bertujuan untuk mengetahui seberapa baik sistem dalam mengenali data yang diberikan, dalam penelitian ini dilakukan perhitungan akurasi untuk setiap karakter menggunakan persamaan 2.44.

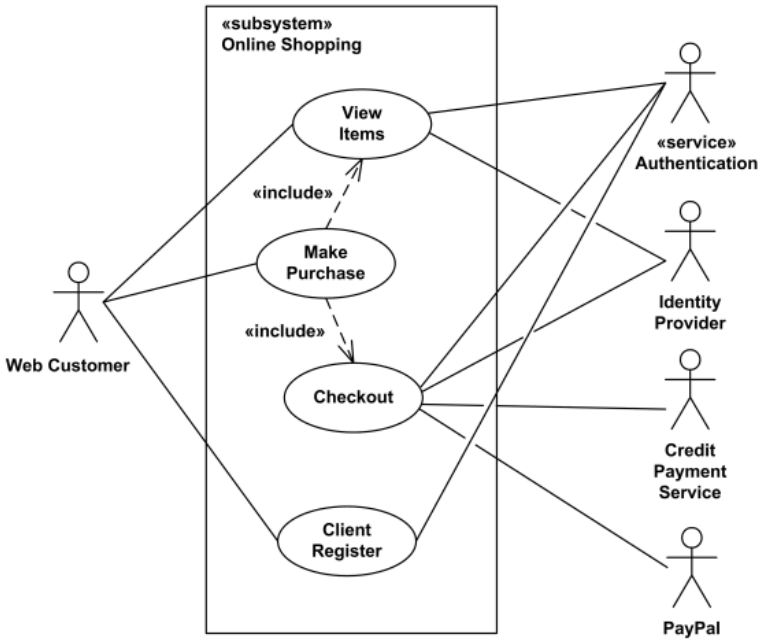
$$Akurasi = \frac{Jumlah\ data\ karakter\ sesuai}{Jumlah\ keseluruhan\ data\ uji} \times 100\% \tag{2.44}$$

2.5 Unified Modeling Language

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 [20]. Adapun yang menjadi pegangan dalam pembuatan UML ini berdasarkan *Ebook* oleh Rules Miles [21].

2.5.1 Diagram Use-case (*Use-case diagram*)

Use-case diagram digunakan untuk mendeskripsikan interaksi antara sistem dengan penggunanya, serta apa yang seharusnya dilakukan oleh sistem dan menyediakan cara mendeskripsikan pandangan eksternal terhadap sistem dan interaksinya dengan dunia luar atau *user* [21]. Berikut adalah contoh dari *use-case* diagram pada Gambar 2.9.



Gambar 2.9 Contoh *Use-case Diagram*

Use-case pada Gambar 2.9 digunakan untuk mendeskripsikan kemampuan dari sebuah sistem *online shopping* beserta hak akses dari penggunanya. Terdapat lima pengguna yang dapat mengakses sistem, yaitu *web customer*, *authentication*,

identity provider, *credit payment service*, dan *paypal*. Berikut adalah detail hak akses dari setiap pengguna.

Tabel 2.9 Detail Hak Akses Pengguna Pada Sistem *Online Shopping*

Hak Akses	Pengguna
View Items	1. *Web Customer* 2. *Authentication* 3. *Identity Provider*
Make Purchase	1. *Web Customer*
Checkout	1. *Identity Provider* 2. *Credit Payment Service* 3. *Paypal*
Client Register	1. *Web Customer* 2. *Authentication*

2.5.2 Use-case Scenario

Use-case scenario merupakan penjelasan secara tekstual dari sekumpulan skenario interaksi. Setiap skenario yang ada akan mendeskripsikan urutan aksi/langkah yang dilakukan aktor ketika berinteraksi dengan sistem, baik dalam kondisi berhasil atau gagal [20].

Use-case scenario dijelaskan secara tekstual dalam beberapa format tergantung kebutuhannya, yaitu singkat (*brief*), *informal* (*casual*), atau lengkap (*fully dressed*) (Larman, 2005), yang dapat dijelaskan dalam satu atau dua kolom (Cockburn, 2000) [20].

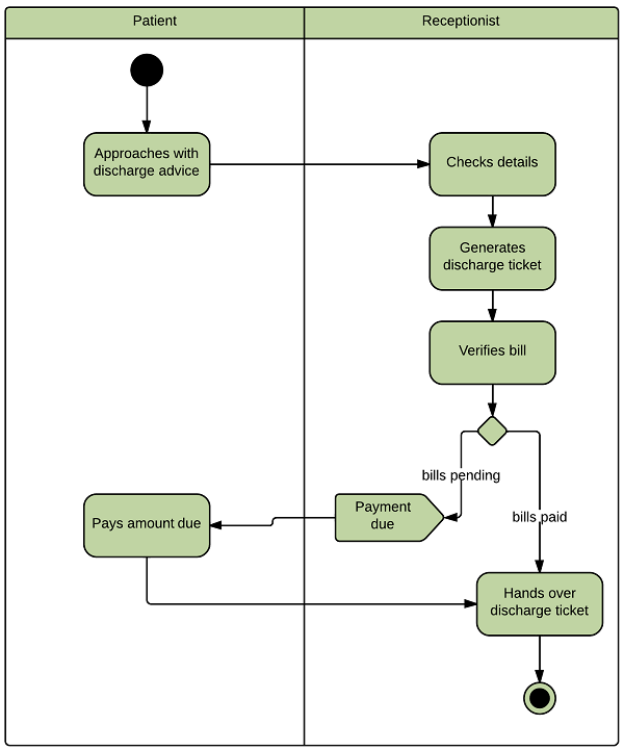
Pada penelitian ini digunakan format yang diperkenalkan oleh (Rules Miles, 2006) [21] dengan bagian yang terlibat sebagai berikut.

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.

2.5.3 Diagram Aktivitas (*Activity diagram*)

Activity diagram adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain berdasarkan *use case diagram*. Berikut adalah contoh dari *activity diagram* pada Gambar 2.10.



Gambar 2.10 Contoh *Acitivity Diagram*

2.5.4 Diagram Kelas (*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 ia berperilaku dalam lingkungannya [21].

2.5.5 Diagram Sequence (*Sequence Diagram*)

Squence 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 [21].

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 [21].

2.6 Pengujian *Black Box*

Pengujian *black box* juga dikenal sebagai pengujian perilaku, berfokus pada persyaratan fungsional perangkat lunak. Artinya, teknik pengujian ini bertujuan untuk menemukan kesalahan fungsionalitas, kesalahan antarmuka, kesalahan dalam struktur data, kesalahan perilaku atau kinerja, dan kesalahan inisialisasi dan penghentian [12].

2.7 Database

Penggunaan *database* pada penelitian ini bertujuan untuk mempermudah proses pengolahan data. Terutama dalam menyimpan informasi terkait pengolahan data huruf baik berupa data biner, nama gambar, dan hasil belajar metode CNN.

2.7.1 Entity Relationship Diagram

Entity Relationship Diagram (ERD) adalah pemodelan awal basis data yang dikembangkan berdasarkan teori himpunan dalam bidang matematika untuk pemodelan basis data relasional [22].

2.7.2 Database Management System

Sistem manajemen *database* atau *Database Management System* (DBMS) merupakan suatu perangkat lunak yang digunakan untuk membuat dan mengolah *database*, dengan kata lain merupakan perantara antara basis data dengan user. DBMS yang digunakan sebagai pendukung dalam penelitian ini ialah SQLyog.

2.8 Hypertext Markup Language

HyperText Markup Language (HTML) adalah bahasa yang digunakan untuk menulis halaman web. HTML merupakan pengembangan dari standar pemformatan dokumen teks, yaitu *Standard Generalized Markup Language* (SGML). HTML pada dasarnya merupakan dokumen ASCII atau teks biasa, yang dirancang untuk tidak tergantung pada suatu sistem operasi tertentu [23].

2.9 Cascading Style Sheet

Cascading Style Sheet (CSS) adalah suatu bahasa *stylesheet* yang digunakan untuk mengatur tampilan suatu *website*, baik tata letaknya, jenis huruf, warna, dan semua yang berhubungan dengan tampilan. Pada umumnya CSS digunakan untuk memformat halaman web yang ditulis dengan HTML atau XHTML.

2.10 JavaScript

JavaScript (JS) dapat digunakan untuk mendeteksi dan beraksi terhadap *event – event* yang disebabkan oleh pengguna. JS mampu memperbaiki situs *web* dengan bantuan navigasional, kotak dialog, citra dinamis, dan lainnya. JS juga dapat digunakan untuk mengendalikan tampilan halaman, memvalidasi suatu proses, perhitungan aritmatik, manipulasi tanggal dan waktu, dan berbagai macam pengolahan data lainnya [24].

2.11 Hypertext Preprocessor

Hypertext Preprocessor (PHP) secara mendasar dapat mengerjakan semua yang dapat dikerjakan oleh CGI (*Commong Gateway Interface*), seperti mendapatkan data dari *form*, menghasilkan isi halaman *web* yang dinamik, dan menerima *cookies*. CGI adalah spesifikasi standar modul yang ditambahkan kepada server *web*, agar *server* web dapat memiliki kemampuan untuk dapat memberikan layanan yang interaktif, tidak sekedar melayani permintan dokumen web HTML dan penghubung antara *web* dengan server saja [25].

2.12 Laravel

Pembangunan program pada penelitian ini berbasis *web*, dimana *framework* yang digunakan ialah *laravel*. Menggunakan *laravel* memiliki keuntungan dalam pemanfaatan konsep *Model View Controller* (MVC). Selain daripada itu, penggunaan *laravel* pun mendukung pemrograman berorientasi objek (PBO) dengan bahasa pemrograman PHP.